LinksPlatform's Platform.Data.Doublets Class Library

```
./Converters/AddressToUnaryNumberConverter.cs
                                                                                                        private readonly IConverter<TLink> unaryNumberToAddressConverter;
    using System Collections Generic:
                                                                                              13
                                                                                                        public LinkToItsFrequencyNumberConveter(
    using Platform.Interfaces:
                                                                                              14
                                                                                                           ILinks<TLink> links.
    using Platform.Reflection:
                                                                                              1.5
                                                                                                           ISpecificPropertyOperator<TLink, TLink> frequencyPropertyOperator,
    using Platform. Numbers:
                                                                                              16
                                                                                                           IConverter < TLink > unary Number To Address Converter)
                                                                                              17
    namespace Platform.Data.Doublets.Converters
                                                                                                           : base(links)
                                                                                              18
                                                                                              10
       public class AddressToUnaryNumberConverter<TLink>: LinksOperatorBase<TLink>.
                                                                                                            frequencyPropertyOperator = frequencyPropertyOperator;
                                                                                              20
                                                                                                            \overline{u}nary\overline{N}umber\overline{T}o\overline{A}dd\overline{d}ress\overline{C}onverter\overline{e} = \overline{u}nary\overline{N}umber\overline{T}o\overline{A}dd\overline{d}ress\overline{C}onverter;
           IConverter<TLink>
                                                                                              21
                                                                                              22
          private static readonly EqualityComparer<TLink> equalityComparer =
                                                                                              23
                                                                                                         public TLink Convert(Doublet<TLink> doublet)
          → EqualityComparer<TLink>.Default:
                                                                                              24
                                                                                              25
          private readonly IConverter<int, TLink> powerOf2ToUnaryNumberConverter;
                                                                                                           var link = Links.SearchOrDefault(doublet.Source, doublet.Target);
                                                                                                           if (equalityComparer.Equals(link, Links.Constants.Null))
13
                                                                                              27
          public AddressToUnaryNumberConverter(ILinks<TLink> links, IConverter<int,
14
                                                                                              28
              TLink> powerOf2ToUnaryNumberConverter): base(links) =>
                                                                                                              throw new ArgumentException($\sigma"Link with {doublet.Source} source and
                                                                                              29
              powerOf2ToUnaryNumberConverter = powerOf2ToUnaryNumberConverter:
                                                                                                              15
                                                                                              30
          public TLink Convert(TLink sourceAddress)
16
                                                                                                           var frequency = frequency Property Operator. Get (link):
                                                                                              31
17
                                                                                                               equalityComparer.Equals(frequency, default))
                                                                                              32
             var number = sourceAddress:
                                                                                              33
             var target = Links.Constants.Null:
                                                                                                              return default:
             for (int i = 0; i < CachedTypeInfo<TLink>.BitsLength; i++)
21
                                                                                                           var frequencyNumber = Links.GetSource(frequency);
               if (equalityComparer.Equals(ArithmeticHelpers.And(number,
                                                                                                           var number = unaryNumberToAddressConverter.Convert(frequencyNumber);
                   Integer<TLink>.One), Integer<TLink>.One))
                                                                                                           return number:
                                                                                              38
                                                                                              39
                  target = equalityComparer.Equals(target, Links.Constants.Null)
                                                                                              40
                        powerOf2ToUnaryNumberConverter.Convert(i)
                                                                                              41
                     : Links.GetOrCreate( powerOf2ToUnaryNumberConverter.Convert(i),
                                                                                              . /Converters /PowerOf2ToUnaryNumberConverter.cs
                     \hookrightarrow target);
                                                                                                  using System:
                                                                                                  using System.Collections.Generic;
               number = (Integer<TLink>)((ulong)(Integer<TLink>)number >> 1); //
                                                                                                  using Platform.Interfaces;
                   Should be BitwiseHelpers.ShiftRight(number, 1):
               if (equalityComparer.Equals(number, default))
                                                                                                  namespace Platform.Data.Doublets.Converters
                  break:
3.1
                                                                                                     public class PowerOf2ToUnaryNumberConverter<TLink>: LinksOperatorBase<TLink>,
                                                                                                         IConverter<int, TLink>
             return target:
                                                                                                        private static readonly EqualityComparer<TLink> equalityComparer =
35
                                                                                                         → EqualityComparer<TLink>.Default;
36
                                                                                                        private readonly TLink[] unaryNumberPowersOf2;
                                                                                              11
                                                                                              12
./Converters/LinkToItsFrequencyNumberConveter.cs
                                                                                                        public PowerOf2ToUnaryNumberConverter(ILinks<TLink> links, TLink one):
                                                                                              13
    using System:
                                                                                                            base(links)
    using System.Collections.Generic;
                                                                                              14
    using Platform.Interfaces;
                                                                                                             unaryNumberPowersOf2 = new TLink[64];
                                                                                              15
                                                                                                            unaryNumberPowersOf2[0] = one;
                                                                                              16
    namespace Platform.Data.Doublets.Converters
                                                                                              17
                                                                                              18
       public class LinkToItsFrequencyNumberConveter<TLink>:
                                                                                                        public TLink Convert(int power)
                                                                                              19
           LinksOperatorBase<TLink>, IConverter<Doublet<TLink>, TLink>
                                                                                              20
                                                                                                           if (power < 0 \mid power >= unaryNumberPowersOf2.Length)
                                                                                              21
          private static readonly EqualityComparer<TLink> equalityComparer =
                                                                                              22
          → EqualityComparer<TLink>.Default;
                                                                                                              throw new ArgumentOutOfRangeException(nameof(power));
                                                                                              23
          private readonly ISpecificPropertyOperator<TLink, TLink>
                                                                                              24
                                                                                                           if (! equalityComparer.Equals( unaryNumberPowersOf2[power], default))
```

```
return unaryNumberPowersOf2[power];
                                                                                                                                                                                    return unaryToUInt64[unaryNumber];
                                                                                                                                                           49
                                                                                                                                                           50
28
                     var previousPowerOf2 = Convert(power - 1);
                                                                                                                                                                                else
                                                                                                                                                           51
                     var powerOf2 = Links.GetOrCreate(previousPowerOf2, previousPowerOf2);
                                                                                                                                                           52
                       unarvNumberPowersOf2[power] = powerOf2:
                                                                                                                                                                                    var result = unaryToUInt64[source];
                                                                                                                                                           53
3.1
                                                                                                                                                                                    TLink lastValue:
                     return powerOf2;
32
                                                                                                                                                           54
                                                                                                                                                                                    while (! unary ToUInt 64. Try Get Value (target, out last Value))
                                                                                                                                                           5.5
34
                                                                                                                                                           56
                                                                                                                                                                                         source = Links.GetSource(target);
                                                                                                                                                           57
                                                                                                                                                                                         result = ArithmeticHelpers.Add(result, unaryToUInt64[source]);
                                                                                                                                                           58
                                                                                                                                                                                         target = Links.GetTarget(target):
                                                                                                                                                           59
./Converters/UnaryNumberToAddressAddOperationConverter.cs
                                                                                                                                                           60
       using System. Collections. Generic:
                                                                                                                                                                                     result = ArithmeticHelpers.Add(result, lastValue);
       using Platform.Interfaces:
                                                                                                                                                                                    return result:
                                                                                                                                                           62
       using Platform. Numbers:
       namespace Platform.Data.Doublets.Converters
                                                                                                                                                           64
                                                                                                                                                           65
            public class UnaryNumberToAddressAddOperationConverter<TLink>:
                                                                                                                                                           66
                  LinksOperatorBase<TLink>, IConverter<TLink>
                private static readonly EqualityComparer<TLink> equalityComparer =
                                                                                                                                                           ./Converters/UnaryNumberToAddressOrOperationConverter.cs
                 → EqualityComparer<TLink>.Default:
                                                                                                                                                                  using System.Collections.Generic:
                 private Dictionary TLink, TLink unary ToUInt 64:
                                                                                                                                                                 using Platform. Interfaces;
11
                                                                                                                                                                 using Platform Reflection:
                private readonly TLink unaryOne;
19
                                                                                                                                                                  using Platform. Numbers:
13
                 public UnaryNumberToAddressAddOperationConverter(ILinks<TLink> links, TLink
                                                                                                                                                                  namespace Platform. Data. Doublets. Converters
                      unaryOne)
                     : base(links)
                                                                                                                                                                      public class UnaryNumberToAddressOrOperationConverter<TLink>:
                                                                                                                                                                            LinksOperatorBase<TLink>, IConverter<TLink>
                        unarvOne = unarvOne:
17
                     InitUnaryToUInt64();
                                                                                                                                                                           private static readonly EqualityComparer<TLink> equalityComparer =
                                                                                                                                                                            → EqualityComparer<TLink>.Default;
20
                 private void InitUnaryToUInt64()
                                                                                                                                                           1.1
21
                                                                                                                                                                           private readonly IDictionary < TLink, int > unary Number Power Of 2 Indicies;
22
                                                                                                                                                           13
                       unaryToUInt64 = new Dictionary<TLink, TLink>
                                                                                                                                                                           public UnaryNumberToAddressOrOperationConverter(ILinks<TLink> links.
                                                                                                                                                           14
24
                                                                                                                                                                                  IConverter<int, TLink> powerOf2ToUnaryNumberConverter)
                             unaryOne, Integer<TLink>.One }
25
                                                                                                                                                                                  base(links)
26
                     var unary = unaryOne;
                                                                                                                                                           16
27
                                                                                                                                                                                  unaryNumberPowerOf2Indicies = new Dictionary<TLink, int>();
                     var number = Integer < TLink>. One;
                                                                                                                                                           17
                                                                                                                                                                                \overline{\text{for}} (int i = 0; i < CachedTypeInfo<TLink>.BitsLength; i++)
                                                                                                                                                           18
                     for (var i = 1: i < 64: i++)
29
                                                                                                                                                           19
                                                                                                                                                                                      unary Number Power Of 2 Indicies. Add (power Of 2 ToUnary Number Converter. Contract Contra
                           unaryToUInt64.Add(unary = Links.GetOrCreate(unary, unary), number =
                                                                                                                                                           20
31
                                                                                                                                                                                           vert(i)
                                (Integer<TLink>)((Integer<TLink>)number * 2UL));
                                                                                                                                                                                           i);
32
33
                                                                                                                                                           ^{21}
34
                                                                                                                                                           22
                 public TLink Convert (TLink unary Number)
35
                                                                                                                                                           23
                                                                                                                                                                            public TLink Convert(TLink sourceNumber)
                                                                                                                                                           ^{24}
                     if ( equalityComparer.Equals(unaryNumber, default))
                                                                                                                                                           25
37
                                                                                                                                                                                var source = sourceNumber;
                                                                                                                                                           26
                          return default;
                                                                                                                                                                                var target = Links.Constants.Null:
39
                                                                                                                                                           27
                                                                                                                                                                                while (! equalityComparer.Equals(source, Links.Constants.Null))
                                                                                                                                                           28
                           equalityComparer.Equals(unaryNumber, unaryOne))
                                                                                                                                                           29
41
                                                                                                                                                                                     if (unaryNumberPowerOf2Indicies.TryGetValue(source, out int
                                                                                                                                                           30
                          return Integer<TLink>.One;
                                                                                                                                                                                           powerOf2Index))
                                                                                                                                                           31
                     var source = Links.GetSource(unaryNumber);
                                                                                                                                                                                         source = Links.Constants.Null;
                                                                                                                                                           32
                     var target = Links.GetTarget(unaryNumber)
                     if (equalityComparer.Equals(source, target))
                                                                                                                                                                                     else
                                                                                                                                                           34
```

```
private static readonly EqualityComparer<TLink> equalityComparer =
                   powerOf2Index = unaryNumberPowerOf2Indicies[Links.GetSource(source)]:
                   source = Links.GetTarget(source);
                                                                                                               EqualityComparer<TLink>.Default;
                                                                                                 1.0
                                                                                                           public LinksCascadeUniquenessAndDependenciesResolver(ILinks<TLink> links) :
                target = (Integer<TLink>)((Integer<TLink>)target | 1UL << powerOf2Index):
                                                                                                 11
                    // MathHelpers.Or(target, MathHelpers.ShiftLeft(One, powerOf2Index))
                                                                                                            \rightarrow base(links) { }
                                                                                                 12
                                                                                                           protected override TLink ResolveAddressChangeConflict(TLink oldLinkAddress,
                                                                                                 13
             return target:
                                                                                                               TLink newLinkAddress)
42
                                                                                                 14
                                                                                                                 TODO: Very similar to Merge (logic should be reused)
                                                                                                 1.5
44
                                                                                                              ulong referencesAsSourceCount = (Integer < TLink > )Links.Count(Constants.Any
                                                                                                              → oldLinkAddress. Constants.Anv):
./Decorators/LinksCascadeDependenciesResolver.cs
                                                                                                              ulong referencesAsTargetCount = (Integer<TLink>)Links.Count(Constants.Anv
    using System.Collections.Generic:
                                                                                                 17
                                                                                                                  Constants. Anv. oldLinkAddress):
    using Platform. Collections. Arrays;
    using Platform. Numbers:
                                                                                                              var references = ArrayPool.Allocate<TLink>((long)(referencesAsSourceCount +
                                                                                                 18
                                                                                                                  referencesAsTargetCount)):
    namespace Platform.Data.Doublets.Decorators
                                                                                                              var referencesFiller = new ArrayFiller < TLink, TLink > (references,
                                                                                                                   Constants.Continue);
        public class LinksCascadeDependenciesResolver<TLink> : LinksDecoratorBase<TLink>
                                                                                                              Links. Each (references Filler. AddFirst AndReturn Constant. Constants. Any
                                                                                                 20
                                                                                                                  oldLinkAddress, Constants, Anv);
          private static readonly EqualityComparer<TLink> equalityComparer =
                                                                                                              Links. Each (references Filler. Add First And Return Constant, Constants. Any,
                                                                                                 21
          → EqualityComparer<TLink>.Default;
                                                                                                              → Constants.Anv, oldLinkAddress);
          public LinksCascadeDependenciesResolver(ILinks<TLink> links) : base(links) { }
                                                                                                              for (ulong i = 0; i < referencesAsSourceCount; <math>i++)
                                                                                                 22
                                                                                                 23
          public override void Delete(TLink link)
                                                                                                                 var reference = references[i];
                                                                                                 ^{24}
                                                                                                                 if (! equalityComparer.Equals(reference, oldLinkAddress))
                                                                                                 25
             EnsureNoDependenciesOnDelete(link);
                                                                                                 26
             base.Delete(link);
                                                                                                                    Links.Update(reference, newLinkAddress, Links.GetTarget(reference));
                                                                                                 27
                                                                                                 28
                                                                                                 29
          public void EnsureNoDependenciesOnDelete(TLink link)
                                                                                                              for (var i = (long)referencesAsSourceCount; i < references.Length; i++)
                                                                                                 30
                                                                                                 31
             ulong referencesCount = (Integer<TLink>)Links.Count(Constants.Any, link);
2.1
                                                                                                                 var reference = references[i];
                                                                                                 32
             var references = ArrayPool.Allocate<TLink>((long)referencesCount):
                                                                                                                 if (! equalityComparer.Equals(reference, oldLinkAddress))
                                                                                                 33
             var referencesFiller = new ArrayFiller < TLink, TLink > (references,
23
                                                                                                 34
             → Constants.Continue):
                                                                                                                    Links. Update (reference, Links. Get Source (reference), newLink Address);
                                                                                                 35
             Links. Each (references Filler. AddFirst AndReturn Constant, Constants. Any, link);
                                                                                                 36
             //references.Sort() // TODO: Решить необходимо ли для корректного порядка
                                                                                                 37
                отмены операций в транзакциях
                                                                                                              ArrayPool.Free(references):
                                                                                                 38
             for (var i = (long) references Count - 1; i >= 0; i--)
                                                                                                              return base.ResolveAddressChangeConflict(oldLinkAddress, newLinkAddress);
                                                                                                 39
                                                                                                 40
27
                if (equalityComparer.Equals(references[i], link))
                                                                                                 41
                                                                                                 42
                   continue;
3.1
                Links.Delete(references[i]);
                                                                                                 ./Decorators/LinksDecoratorBase.cs
             ArrayPool.Free(references);
                                                                                                     using System;
35
                                                                                                     using System.Collections.Generic;
36
                                                                                                     using Platform.Data.Constants;
37
                                                                                                      namespace Platform.Data.Doublets.Decorators
./Decorators/LinksCascadeUniquenessAndDependenciesResolver.cs
                                                                                                        public abstract class LinksDecoratorBase<T>: ILinks<T>
    using System Collections Generic:
    using Platform.Collections.Arrays;
                                                                                                           public LinksCombinedConstants<T, T, int> Constants { get; }
    using Platform. Numbers;
                                                                                                           public readonly ILinks<T> Links;
    namespace Platform.Data.Doublets.Decorators
                                                                                                 12
                                                                                                           protected LinksDecoratorBase(ILinks<T> links)
       public class LinksCascadeUniquenessAndDependenciesResolver<TLink>:
                                                                                                 13
                                                                                                 14
        → LinksUniquenessResolver<TLink>
```

```
Links = links;
                                                                                                23
             Constants = links.Constants:
                                                                                                           public virtual T Create() => Links.Create():
                                                                                                24
17
                                                                                                25
                                                                                                           public virtual T Update(IList<T> restrictions) => Links.Update(restrictions);
                                                                                                26
          public virtual T Count(IList<T> restriction) => Links.Count(restriction);
                                                                                                27
19
                                                                                                           public virtual void Delete(T link) => Links.Delete(link);
20
                                                                                                28
          public virtual T Each(Func<IList<T>, T> handler, IList<T> restrictions) =>
21
                                                                                                20
                                                                                                           protected override bool AllowMultipleDisposeCalls => true;
                                                                                                30
          → Links.Each(handler, restrictions);
                                                                                                31
                                                                                                           protected override void DisposeCore(bool manual, bool wasDisposed) =>
                                                                                                32
          public virtual T Create() => Links.Create();
23
                                                                                                           → Disposable.TryDispose(Links);
24
          public virtual T Update(IList<T> restrictions) => Links.Update(restrictions);
                                                                                                33
                                                                                                34
          public virtual void Delete(T link) => Links.Delete(link):
27
                                                                                                 ./Decorators/LinksInnerReferenceValidator.cs
29
                                                                                                     using System:
                                                                                                     using System. Collections. Generic;
./Decorators/LinksDependenciesValidator.cs
    using System.Collections.Generic:
                                                                                                     namespace Platform.Data.Doublets.Decorators
    namespace Platform.Data.Doublets.Decorators
                                                                                                        // TODO: Make LinksExternalReferenceValidator. A layer that checks each link to exist
                                                                                                        → or to be external (hybrid link's raw number).
        public class LinksDependenciesValidator<T>: LinksDecoratorBase<T>
                                                                                                        public class LinksInnerReferenceValidator<T>: LinksDecoratorBase<T>
          public LinksDependenciesValidator(ILinks<T> links) : base(links) { }
                                                                                                           public LinksInnerReferenceValidator(ILinks<T> links): base(links) {
                                                                                                 10
          public override T Update(IList<T> restrictions)
                                                                                                           public override T Each(Func<IList<T>, T> handler, IList<T> restrictions)
                                                                                                11
                                                                                                12
             Links.EnsureNoDependencies(restrictions[Constants.IndexPart]):
                                                                                                              Links.EnsureInnerReferenceExists(restrictions, nameof(restrictions));
                                                                                                 13
             return base. Update(restrictions);
                                                                                                              return base. Each (handler, restrictions);
                                                                                                14
                                                                                                 15
                                                                                                16
          public override void Delete(T link)
                                                                                                           public override T Count(IList<T> restriction)
                                                                                                17
                                                                                                18
             Links.EnsureNoDependencies(link);
                                                                                                              Links.EnsureInnerReferenceExists(restriction, nameof(restriction));
                                                                                                19
             base.Delete(link):
                                                                                                              return base.Count(restriction);
                                                                                                20
                                                                                                21
                                                                                                22
21
                                                                                                           public override T Update(IList<T> restrictions)
                                                                                                23
                                                                                                24
./Decorators/LinksDisposableDecoratorBase.cs
                                                                                                              // TODO: Possible values: null, ExistentLink or
                                                                                                25
    using System:
                                                                                                              → NonExistentHvbrid(ExternalReference)
    using System.Collections.Generic;
                                                                                                             Links. EnsureInnerReferenceExists(restrictions, nameof(restrictions));
                                                                                                26
    using Platform. Disposables:
                                                                                                              return base. Update (restrictions);
                                                                                                27
    using Platform. Data. Constants;
                                                                                                28
                                                                                                29
    namespace Platform.Data.Doublets.Decorators
                                                                                                           public override void Delete(T link)
                                                                                                30
                                                                                                31
        public abstract class LinksDisposableDecoratorBase<T>: DisposableBase, ILinks<T>
                                                                                                              // TODO: Решить считать ли такое исключением, или лишь более конкретным
                                                                                                32
                                                                                                              → требованием?
          public LinksCombinedConstants<T, T, int> Constants { get; }
                                                                                                              Links.EnsureLinkExists(link, nameof(link));
                                                                                                33
                                                                                                34
                                                                                                              base.Delete(link):
          public readonly ILinks<T> Links;
                                                                                                35
          protected LinksDisposableDecoratorBase(ILinks<T> links)
                                                                                                36
                                                                                                37
             Links = links:
             Constants = links.Constants;
                                                                                                 ./Decorators/LinksNonExistentReferencesCreator.cs
                                                                                                     using System.Collections.Generic;
          public virtual T Count(IList<T> restriction) => Links.Count(restriction);
20
                                                                                                     namespace Platform.Data.Doublets.Decorators
21
          public virtual T Each(Func<IList<T>, T> handler, IList<T> restrictions) =>
                                                                                                        /// <remarks>
```

```
Not practical if newSource and newTarget are too big.
           To be able to use practical version we should allow to create link at any specific
                                                                                                            public LinksSelfReferenceResolver(ILinks<TLink> links) : base(links) { }
                                                                                                  10
        → location inside ResizableDirectMemoryLinks.
                                                                                                  11
                                                                                                            public override TLink Each(Func<IList<TLink>, TLink> handler, IList<TLink>
       /// This in turn will require to implement not a list of empty links, but a list of ranges to
                                                                                                  12

→ restrictions)

        → store it more efficiently.
        /// </remarks>
                                                                                                  13
                                                                                                               if (! equalityComparer.Equals(Constants.Any, Constants.Itself)
       public class LinksNonExistentReferencesCreator<T>: LinksDecoratorBase<T>
                                                                                                  14
                                                                                                                && (((restrictions.Count > Constants.IndexPart) &&
                                                                                                  15
11
          public LinksNonExistentReferencesCreator(ILinks<T> links) : base(links) { }
                                                                                                                      equalityComparer.Equals(restrictions[Constants.IndexPart],
                                                                                                                     Constants. Itself))
          public override T Update(IList<T> restrictions)
                                                                                                                || ((restrictions.Count > Constants.SourcePart) &&
                                                                                                  16
                                                                                                                      equalityComparer.Equals(restrictions[Constants.SourcePart],
             Links. EnsureCreated(restrictions[Constants.SourcePart],
                                                                                                                     Constants. Itself))
             → restrictions[Constants.TargetPart]):
                                                                                                                || ((restrictions.Count > Constants.TargetPart) &&
                                                                                                  17
             return base. Update(restrictions);
                                                                                                                      equalityComparer.Equals(restrictions[Constants.TargetPart],
                                                                                                                    \overline{C}onstants.Itself))))
                                                                                                                  return Constants.Continue:
                                                                                                  19
                                                                                                  20
./Decorators/LinksNullToSelfReferenceResolver.cs
                                                                                                               return base.Each(handler, restrictions);
                                                                                                  21
    using System.Collections.Generic;
                                                                                                  22
                                                                                                  23
    namespace Platform.Data.Doublets.Decorators
                                                                                                            public override TLink Update(IList<TLink> restrictions)
                                                                                                  ^{24}
                                                                                                  25
        public class LinksNullToSelfReferenceResolver<TLink> : LinksDecoratorBase<TLink>
                                                                                                               restrictions[Constants.SourcePart] =
                                                                                                  26
                                                                                                                     equalityComparer.Equals(restrictions[Constants.SourcePart],
          private static readonly EqualityComparer<TLink> equalityComparer =
                                                                                                                    Constants.Itself) ? restrictions[Constants.IndexPart] :
           → EqualityComparer<TLink>.Default;
                                                                                                                   restrictions[Constants.SourcePart];
                                                                                                               restrictions[Constants.TargetPart] =
          public LinksNullToSelfReferenceResolver(ILinks<TLink> links) : base(links) { }
                                                                                                                     equalityComparer.Equals(restrictions[Constants.TargetPart],
          public override TLink Create()
                                                                                                                    Constants. Itself) ? restrictions [Constants. IndexPart] :
                                                                                                                   restrictions[Constants.TargetPart];
             var link = base.Create();
                                                                                                               return base.Update(restrictions);
                                                                                                  28
             return Links. Update(link, link, link);
                                                                                                  29
                                                                                                  30
                                                                                                  31
          public override TLink Update(IList<TLink> restrictions)
             restrictions[Constants.SourcePart] =
                                                                                                  ./Decorators/LinksUniquenessResolver.cs
                    equalityComparer.Equals(restrictions[Constants.SourcePart],
                                                                                                      using System.Collections.Generic;
                 Constants.Null) ? restrictions[Constants.IndexPart] :
                 restrictions[Constants.SourcePart];
                                                                                                      namespace Platform. Data. Doublets. Decorators
             restrictions[Constants.TargetPart] =
                    equalityComparer.Equals(restrictions[Constants.TargetPart],
                                                                                                         public class LinksUniquenessResolver<TLink>: LinksDecoratorBase<TLink>
                  Constants.Null) ? restrictions[Constants.IndexPart] :
                                                                                                            private static readonly EqualityComparer<TLink> equalityComparer =
                  restrictions[Constants.TargetPart];
                                                                                                             → EqualityComparer<TLink>.Default;
             return base. Update(restrictions);
                                                                                                            public LinksUniquenessResolver(ILinks<TLink> links): base(links) { }
                                                                                                  10
                                                                                                            public override TLink Update(IList<TLink> restrictions)
                                                                                                  11
                                                                                                  12
./Decorators/LinksSelfReferenceResolver.cs
                                                                                                               var newLinkAddress = Links.SearchOrDefault(restrictions|Constants.SourcePart|
                                                                                                  13
    using System;
                                                                                                                   restrictions[Constants.TargetPart]);
    using System.Collections.Generic;
                                                                                                               if (equalityComparer.Equals(newLinkAddress, default))
                                                                                                  14
                                                                                                  15
    namespace Platform.Data.Doublets.Decorators
                                                                                                                  return base. Update(restrictions);
                                                                                                  16
                                                                                                  17
        public class LinksSelfReferenceResolver<TLink> : LinksDecoratorBase<TLink>
                                                                                                               return ResolveAddressChangeConflict(restrictions[Constants.IndexPart],
                                                                                                                   newLinkAddress);
          private static readonly EqualityComparer<TLink> equalityComparer =
           → EqualityComparer<TLink>.Default;
                                                                                                  19
```

```
protected virtual TLink ResolveAddressChangeConflict(TLink oldLinkAddress, TLink
                                                                                                 16
21
              newLinkAddress)
                                                                                                 17
22
               (Links.Exists(oldLinkAddress))
23
                                                                                                 18
                                                                                                 19
24
                Delete(oldLinkAddress);
                                                                                                 21
             return newLinkAddress;
28
                                                                                                 22
29
                                                                                                 23
30
                                                                                                 ^{24}
                                                                                                 25
./Decorators/LinksUniquenessValidator.cs
    using System Collections Generic:
                                                                                                 26
                                                                                                 27
    namespace Platform. Data. Doublets. Decorators
                                                                                                 28
       public class LinksUniquenessValidator<T>: LinksDecoratorBase<T>
                                                                                                 29
                                                                                                 30
          public LinksUniquenessValidator(ILinks<T> links) : base(links) { }
                                                                                                 3.1
                                                                                                 32
          public override T Update(IList<T> restrictions)
                                                                                                 33
             Links. EnsureDoesNotExists(restrictions[Constants.SourcePart],
                                                                                                 34
             → restrictions[Constants.TargetPart]):
                                                                                                 35
             return base. Update(restrictions);
                                                                                                 36
                                                                                                 37
                                                                                                 38
15
                                                                                                 39
                                                                                                 40
./Decorators/NonNullContentsLinkDeletionResolver.cs
                                                                                                 41
                                                                                                 42
    namespace Platform.Data.Doublets.Decorators
                                                                                                 43
                                                                                                 44
       public class NonNullContentsLinkDeletionResolver<T>: LinksDecoratorBase<T>
                                                                                                 45
                                                                                                 46
          public NonNullContentsLinkDeletionResolver(ILinks<T> links) : base(links) { }
                                                                                                 47
          public override void Delete(T link)
                                                                                                 48
                                                                                                 49
             Links.Update(link, Constants.Null, Constants.Null);
                                                                                                 50
             base. Delete(link);
                                                                                                 5.1
                                                                                                 52
                                                                                                 5.3
12
                                                                                                 54
./Decorators/UInt64Links.cs
                                                                                                 5.5
    using System:
    using System.Collections.Generic;
                                                                                                 57
    using Platform.Collections:
    using Platform.Collections.Arrays;
                                                                                                 58
                                                                                                 59
    namespace Platform.Data.Doublets.Decorators
                                                                                                 60
                                                                                                 61
           <summary>
                                                                                                 62
        /// Представляет объект для работы с базой данных (файлом) в формате Links
                                                                                                 63
        → (массива взаимосвязей).
                                                                                                 64
        /// </summary>
                                                                                                 65
           <remarks>
                                                                                                 67
           Возможные оптимизации:
           Объединение в одном поле Source и Target с уменьшением до 32 бит.
              + меньше объём БД
```

```
- меньше производительность
      - больше ограничение на количество связей в БД)
   Ленивое хранение размеров поддеревьев (расчитываемое по мере использования
    БД)
      + меньше объём БД
      - больше сложность
      AVL - высота дерева может позволить точно расчитать размер дерева, нет
    необходимости в SBT.
      AVL дерево можно прошить.
   Текущее теоретическое ограничение на размер связей - long.MaxValue
   Желательно реализовать поддержку переключения между деревьями и битовыми
    индексами (битовыми строками) - вариант матрицы (выстраеваемой лениво).
   Решить отключать ли проверки при компиляции под Release. Т.е. исключения
→ будут выбрасываться только при #if DEBUG
'// < / \text{remarks} >
public class UInt64Links: LinksDisposableDecoratorBase<ulong>
  public UInt64Links(ILinks<ulong> links) : base(links) { }
  public override ulong Each(Func<IList<ulong>, ulong> handler, IList<ulong>

→ restrictions)

     this.EnsureLinkIsAnyOrExists(restrictions);
     return Links. Each (handler, restrictions);
  public override ulong Create() => Links.CreatePoint();
   public override ulong Update(IList<ulong> restrictions)
     if (restrictions.IsNullOrEmpty())
        return Constants.Null:
       TODO: Remove usages of these hacks (these should not be backwards compatible)
     if (restrictions.Count == 2)
        return this.Merge(restrictions[0], restrictions[1]);
     if (restrictions.Count == 4)
        return this.UpdateOrCreateOrGet(restrictions[0], restrictions[1], restrictions[2]
         \hookrightarrow restrictions[3]);
      // TODO: Looks like this is a common type of exceptions linked with restrictions
     if (restrictions.Count != 3)
        throw new NotSupportedException();
     var updatedLink = restrictions[Constants.IndexPart];
     this.EnsureLinkExists(updatedLink, nameof(Constants.IndexPart));
     var newSource = restrictions[Constants.SourcePart];
     this.EnsureLinkIsItselfOrExists(newSource, nameof(Constants.SourcePart));
     var newTarget = restrictions[Constants.TargetPart];
     this.EnsureLinkIsItselfOrExists(newTarget, nameof(Constants.TargetPart));
     var existedLink = Constants.Null;
     if (newSource != Constants.Itself && newTarget != Constants.Itself)
```

```
existedLink = this.SearchOrDefault(newSource, newTarget):
71
                                                                                                     12
                 (existedLink == Constants.Null)
72
                                                                                                     13
                                                                                                             /// <remarks>
                 var before = Links.GetLink(updatedLink):
                                                                                                     14
74
                 if (before Constants. Source Part | != newSource || before Constants. Target Part | !=
                                                                                                     15
                                                                                                                 Everything?

→ newTarget)

                    Links.Update(updatedLink, newSource == Constants.Itself? updatedLink:
                                                                                                                  to nothing).
                     \rightarrow newSource.
                                      newTarget == Constants.Itself? updatedLink: newTarget);
                                                                                                     17
                                                                                                     18
                 return updatedLink;
              else
                                                                                                                 </remarks>
                                                                                                     19
                                                                                                     20
                  // Replace one link with another (replaced link is deleted, children are updated
                                                                                                     21
                  → or deleted), it is actually merge operation
                                                                                                     22
                 return this.Merge(updatedLink, existedLink);
                                                                                                     23
                                                                                                     24
                                                                                                     25
               <summary>Удаляет связь с указанным индексом.</summary>
                                                                                                     26
                                                                                                     27
                <param name="link">Индекс удаляемой связи </param>
                                                                                                     28
           public override void Delete(ulong link)
                                                                                                     29
92
                                                                                                     30
              this. EnsureLinkExists(link);
                                                                                                     31
              Links. Update(link, Constants. Null, Constants. Null);
94
                                                                                                     32
              var referencesCount = Links.Count(Constants.Any, link);
                                                                                                     33
              if (referencesCount > 0)
                                                                                                     34
                                                                                                     35
                 var references = new ulong[referencesCount];
                                                                                                     36
                 var referencesFiller = new ArrayFiller < ulong , ulong > (references,
                                                                                                     37
                  → Constants.Continue):
                                                                                                     38
                 Links, Each (references Filler, Add First And Return Constant, Constants, Any, link);
                  //references.Sort(); // TODO: Решить необходимо ли для корректного
                                                                                                     39
                  → порядка отмены операций в транзакциях
                 for (var i = (long)referencesCount - 1; i \ge 0; i--)
102
103
                                                                                                     40
                     if (this.Exists(references[i]))
                                                                                                     41
105
                       Delete(references[i]);
                                                                                                     42
107
108
109
                                                                                                     43
                   / TODO: Определить почему здесь есть связи, которых не существует
110
                                                                                                     44
111
                                                                                                     45
              Links.Delete(link);
112
                                                                                                     46
113
                                                                                                     47
114
                                                                                                     48
115
                                                                                                     49
                                                                                                     50
./Decorators/UniLinks.cs
                                                                                                     5.1
     using System:
                                                                                                     52
     using System.Collections.Generic;
                                                                                                     53
     using System.Ling;
     using Platform.Collections:
                                                                                                     54
     using Platform Collections Arrays:
                                                                                                     55
     using Platform.Collections.Lists;
                                                                                                     56
     using Platform. Helpers. Scopes;
                                                                                                     57
     using Platform. Data. Constants;
```

```
using Platform. Data. Universal;
using System.Collections.ObjectModel:
namespace Platform.Data.Doublets.Decorators
   /// What does empty pattern (for condition or substitution) mean? Nothing or
   /// Now we go with nothing. And nothing is something one, but empty, and cannot be
       changed by itself. But can cause creation (update from nothing) or deletion (update
       TODO: Decide to change to IDoubletLinks or not to change. (Better to create
       Default UniLinks Base, that contains logic itself and can be implemented using both
       IDoublet Links and ILinks.)
   internal class UniLinks<TLink>: LinksDecoratorBase<TLink>, IUniLinks<TLink>
      private static readonly EqualityComparer<TLink> equalityComparer =
      → EqualityComparer<TLink>.Default;
      public UniLinks(ILinks<TLink> links) : base(links) { }
      private struct Transition
         public IList<TLink> Before:
         public IList<TLink> After:
         public Transition(IList<TLink> before, IList<TLink> after)
            Before = before;
            After = after:
      public static readonly TLink NullConstant = Use<LinksCombinedConstants<TLink,
          TLink, int>>.Single.Null:
      public static readonly IReadOnlyList<TLink> NullLink = new
          ReadOnlyCollection<TLink>(new List<TLink> { NullConstant, NullConstant,
          NullConstant \}):
      // TODO: Подумать о том, как реализовать древовидный Restriction и
      → Substitution (Links-Expression)
      public TLink Trigger(IList<TLink> restriction, Func<IList<TLink>, IList<TLink>,
          TLink> matchedHandler, IList<TLink> substitution, Func<IList<TLink>,
          IList<TLink>, TLink> substitutedHandler)
            /List<Transition> transitions = null:
             /if (!restriction.IsNullOrEmpty())
                   Есть причина делать проход (чтение)
                if (matchedHandler!= null)
                   if (!substitution.IsNullOrEmpty())
                       // \text{ restriction} => \{ 0, 0, 0 \} | \{ 0 \} // \text{ Create} 
                        / substitution => { itself, 0, 0 } | { itself, itself, itself } // Create
                      // \text{ substitution} => \{ 0, 0, 0 \} | \{ 0 \} // \text{ Delete}
                      transitions = new List < Transition > ();
                      if (Equals(substitution[Constants.IndexPart], Constants.Null))
```

```
// If index is Null, that means we always ignore every other value
(they are also Null by definition)
                                                                                 115
            var matchDecision = matchedHandler(, NullLink):
                                                                                 116
            if (Equals(matchDecision, Constants, Break))
                                                                                 117
               return false:
                                                                                 118
            if (!Equals(matchDecision, Constants.Skip))
                                                                                 119
               transitions. Add(new Transition(matchedLink, newValue));
                                                                                 120
                                                                                 121
         else
                                                                                 122
                                                                                 123
            Func<T. bool> handler:
                                                                                 124
            handler = link =>
                                                                                 125
                                                                                 126
               var matchedLink = Memory.GetLinkValue(link);
                                                                                 127
               var newValue = Memory.GetLinkValue(link);
                                                                                 128
              newValue[Constants.IndexPart] = Constants.Itself;
                                                                                 129
               newValue Constants.SourcePart =
                                                                                 130
                                                                                 131
Equals(substitution[Constants.SourcePart], Constants.Itself)?
                                                                                 132
matchedLink[Constants.IndexPart]: substitution[Constants.SourcePart];
                                                                                 133
              newValue[Constants.TargetPart] =
                                                                                 134
Equals(substitution[Constants.TargetPart], Constants.Itself)?
                                                                                 135
matchedLink[Constants.IndexPart]: substitution[Constants.TargetPart]
                                                                                 136
              var matchDecision = matchedHandler(matchedLink, newValue);
                                                                                137
               if (Equals(matchDecision, Constants, Break))
                                                                                 138
                 return false:
                                                                                 139
               if (!Equals(matchDecision, Constants.Skip))
                                                                                 140
                 transitions.Add(new Transition(matchedLink, newValue));
                                                                                 141
               return true:
                                                                                 142
                                                                                 143
              (!Memory.Each(handler, restriction))
                                                                                 144
               return Constants.Break:
                                                                                 145
     else
                                                                                 146
                                                                                 147
         Func < T, bool > handler = link = >
                                                                                 148
                                                                                 149
            var matchedLink = Memory.GetLinkValue(link);
                                                                                 150
           var matchDecision = matchedHandler(matchedLink, matchedLink);
                                                                                 151
            return !Equals(matchDecision, Constants.Break);
                                                                                 152
                                                                                 153
           (!Memory.Each(handler, restriction))
                                                                                 154
            return Constants.Break;
                                                                                 155
                                                                                 156
                                                                                 157
  else
                                                                                 158
                                                                                 159
     if (substitution != null)
                                                                                 160
                                                                                 161
         transitions = new List < IList < T >> ();
                                                                                 162
         Func < T, bool > handler = link = >
                                                                                 163
                                                                                 164
            var matchedLink = Memory.GetLinkValue(link);
                                                                                 165
            transitions. Add(matchedLink);
                                                                                 166
            return true:
                                                                                 167
                                                                                 168
           (!Memory.Each(handler, restriction))
                                                                                 169
            return Constants.Break:
                                                                                 170
                                                                                 171
     else
                                                                                 172
```

61

62

76

101

102

103

104

105

107

109

110

111

112

```
return Constants.Continue;
   'if (substitution != null)
         Есть причина делать замену (запись)
       if (substitutedHandler! = null)
      else
  //return Constants.Continue:
//if (restriction.IsNullOrEmpty()) // Create
    substitution[Constants.IndexPart] = Memory.AllocateLink();
    Memory.SetLinkValue(substitution);
 else if (substitution.IsNullOrEmpty()) // Delete
    Memory.FreeLink(restriction[Constants.IndexPart]);
 else if (restriction.EqualTo(substitution)) // Read or ("repeat" the state) // Each
       No need to collect links to list
       Skip == Continue
       No need to check substituedHandler
    if (!Memory.Each(link =>
   !Equals(matchedHandler(Memory.GetLinkValue(link)), Constants.Break),
   restriction))
       return Constants.Break;
 /else // Update
     /List<IList<T>> matchedLinks = null:
    if (matchedHandler!= null)
       matchedLinks = new List < IList < T >> ();
       Func < T, bool> handler = link = >
          var matchedLink = Memory.GetLinkValue(link):
          var matchDecision = matchedHandler(matchedLink);
          if (Equals(matchDecision, Constants.Break))
             return false:
          if (!Equals(matchDecision, Constants.Skip))
             matchedLinks. Add(matchedLink);
          return true:
       if (!Memory.Each(handler, restriction))
          return Constants.Break:
    if (!matchedLinks.IsNullOrEmpty())
       var totalMatchedLinks = matchedLinks.Count;
       for (var i = 0; i < totalMatchedLinks; <math>i++)
          var matchedLink = matchedLinks[i];
          if (substitutedHandler!= null)
```

```
var newValue = new List<T>(); // TODO: Prepare value to update 227
       here
                                                                                       228
                 // TODO: Decide is it actually needed to use Before and After
                                                                                       229
       substitution handling.
                                                                                       230
              var substitutedDecision = substitutedHandler(matchedLink, newValue):
                                                                                      231
                if (Equals(substitutedDecision, Constants.Break))
                                                                                       232
                                                                                       233
                   return Constants.Break;
                 if (Equals(substitutedDecision, Constants.Continue))
                                                                                       234
                                                                                       235
                                                                                       236
                    // Actual update here
                                                                                       237
                   Memory.SetLinkValue(newValue);
                                                                                       238
                 if (Equals(substitutedDecision, Constants.Skip))
                                                                                       239
                                                                                       240
                     / Cancel the update. TODO: decide use separate Cancel constant
                                                                                      241
       or Skip is enough?
                                                                                       242
                                                                                       243
                                                                                       244
                                                                                       245
                                                                                       246
  return Constants.Continue:
                                                                                       247
                                                                                       248
                                                                                       249
public TLink Trigger(IList<TLink> patternOrCondition, Func<IList<TLink>,
                                                                                       250
    TLink> matchHandler, IList<TLink> substitution, Func<IList<TLink>,
                                                                                       251
    IList<TLink>, TLink> substitutionHandler)
                                                                                       252
                                                                                       253
    (patternOrCondition.IsNullOrEmpty() && substitution.IsNullOrEmpty())
                                                                                       254
                                                                                       255
     return Constants.Continue;
                                                                                       256
                                                                                       257
  else if (patternOrCondition.EqualTo(substitution)) // Should be Each here TODO:
                                                                                      258
      Check if it is a correct condition
                                                                                       260
                                                                                       261
        Or it only applies to trigger without matchHandler.
                                                                                       262
     throw new NotImplementedException():
                                                                                       263
  else if (!substitution.IsNullOrEmpty()) // Creation
                                                                                       264
                                                                                       265
     var before = ArrayPool<TLink>.Empty;
                                                                                       266
      // Что должно означать False здесь? Остановиться (перестать идти) или
                                                                                       267
                                                                                       268
         пропустить (пройти мимо) или пустить (взять)?
     if (matchHandler! = null && equalityComparer.Equals(matchHandler(before),
                                                                                       269
          Constants.Break))
                                                                                       270
                                                                                       271
        return Constants.Break:
                                                                                       272
                                                                                       273
     var after = (IList<TLink>)substitution.ToArray();
                                                                                       274
         equalityComparer.Equals(after[0], default))
                                                                                       275
                                                                                       276
        var newLink = Links.Create();
                                                                                       ^{277}
        after[0] = newLink;
                                                                                       278
                                                                                       279
        (substitution.Count == 1)
                                                                                       280
                                                                                       281
        after = Links.GetLink(substitution[0])
                                                                                       282
     else if (substitution.Count == 3)
                                                                                       284
                                                                                       285
```

175

176

177

178

179

180

181

182

183

185

186

187

188

189

190

192

193

194

195

196

197

198

199

200

201

202

203

205

206

207

209

210

211

212

213

214

215

216

217

218

220

222

223

224

```
Links. Update(after);
  else
     throw new NotSupportedException():
    (matchHandler!= null)
     return substitutionHandler(before, after);
  return Constants.Continue:
else if (!patternOrCondition.IsNullOrEmpty()) // Deletion
   if (patternOrCondition.Count == 1)
     var linkToDelete = patternOrCondition[0]:
     var before = Links.GetLink(linkToDelete):
     if (matchHandler!= null &&
          equalityComparer.Equals(matchHandler(before), Constants.Break))
        return Constants.Break;
     var after = ArrayPool<TLink>.Empty:
     Links. Update(linkToDelete, Constants, Null, Constants, Null):
     Links.Delete(linkToDelete):
     if (matchHandler!= null)
        return substitutionHandler(before, after);
     return Constants.Continue;
  else
     throw new NotSupportedException();
else // Replace / Update
   if (patternOrCondition.Count == 1) //-V3125
     var linkToUpdate = patternOrCondition[0];
     var before = Links.GetLink(linkToUpdate):
     if (matchHandler!= null &&
          equalityComparer.Equals(matchHandler(before), Constants.Break))
        return Constants.Break:
     var after = (IList<TLink>)substitution.ToArray(); //-V3125
     if (equalityComparer.Equals(after[0], default))
        after[0] = linkToUpdate;
       (substitution.Count == 1)
          (! equalityComparer.Equals(substitution[0], linkToUpdate))
           after = Links.GetLink(substitution[0]);
           Links. Update(linkToUpdate, Constants. Null, Constants. Null);
           Links.Delete(linkToUpdate);
```

```
else if (substitution.Count == 3)
                                                                                                            public static readonly DoubletComparer<T> Default = new DoubletComparer<T>();
                                                                                                 14
                                                                                                 15
                                                                                                            [MethodImpl(MethodImplOptions.AggressiveInlining)]
                      Links. Update(after);
                                                                                                 16
288
                                                                                                            public bool Equals(Doublet < T > x, Doublet < T > y) =>
                                                                                                 17
289
                                                                                                                 equalityComparer.Equals(x.Source, v.Source) &&
290
                                                                                                                equalityComparer.Equals(x.Target, y.Target);
                      throw new NotSupportedException();
                                                                                                 18
292
                                                                                                            [MethodImpl(MethodImplOptions.AggressiveInlining)]
                                                                                                 19
                    if (matchHandler != null)
                                                                                                            public int GetHashCode(Doublet <T > obj) => unchecked(obj.Source.GetHashCode()
                                                                                                 20
294
                                                                                                            → << 15 ^ obj.Target.GetHashCode()):
295
                      return substitutionHandler(before, after);
296
                                                                                                 21
297
                                                                                                 22
                    return Constants.Continue:
298
299
                                                                                                 ./Doublet.cs
300
                                                                                                      using System;
301
                    throw new NotSupportedException();
                                                                                                      using System. Collections. Generic:
302
                                                                                                      namespace Platform.Data.Doublets
304
305
                                                                                                         public struct Doublet<T>: IEquatable<Doublet<T>>
306
               <remarks>
307
                                                                                                            private static readonly EqualityComparer<T> equalityComparer =
               IList[IList[IList[T]]]
                                                                                                            → EqualityComparer<T>.Default;
309
310
                                                                                                 10
                                                                                                            public T Source { get; set;
                         link
311
                                                                                                            public T Target { get; set;
                                                                                                 11
312
                                                                                                 12
                      change
313
                                                                                                            public Doublet(T source, T target)
                                                                                                 13
314
                                                                                                 14
                    changes
                                                                                                               Source = source;
               < /remarks>
316
                                                                                                               Target = target;
                                                                                                 16
           public IList<IList<ILink>>> Trigger(IList<TLink> condition, IList<TLink>
317
                                                                                                 17
               substitution)
                                                                                                 18
                                                                                                            public override string ToString() => \$"\{Source\}->\{Target\}";
318
                                                                                                 19
              var changes = new List<IList<TLink>>>();
319
                                                                                                 20
              Trigger(condition, AlwaysContinue, substitution, (before, after) =>
                                                                                                            public bool Equals(Doublet < T > other) => equalityComparer.Equals(Source,
320
                                                                                                 21
321
                                                                                                               other.Source) && equalityComparer.Equals(Target, other.Target);
                 var change = new[] { before, after };
322
                                                                                                 22
                 changes. Add(change):
323
                                                                                                 23
                 return Constants.Continue;
324
325
                                                                                                  ./Hybrid.cs
              return changes;
326
327
                                                                                                      using System:
328
                                                                                                      using System. Reflection;
           private TLink AlwaysContinue(IList<TLink> linkToMatch) => Constants.Continue;
                                                                                                      using Platform Reflection;
329
                                                                                                      using Platform.Converters;
330
331
                                                                                                      using Platform.Numbers;
./DoubletComparer.cs
                                                                                                      namespace Platform.Data.Doublets
     using System. Collections. Generic:
                                                                                                         public class Hybrid<T>
     using System.Runtime.CompilerServices;
     namespace Platform.Data.Doublets
                                                                                                            public readonly T Value:
                                                                                                            public bool IsNothing => Convert.ToInt64(To.Signed(Value)) == 0:
                                                                                                 12
                                                                                                            public bool IsInternal => Convert. ToInt64(To.Signed(Value)) > 0;
                                                                                                 13
                                                                                                            public bool IsExternal => Convert ToInt64(To.Signed(Value)) < 0:
            TODO: Может стоит попробовать ref во всех методах (IRefEqualityComparer)
                                                                                                 14
                                                                                                            public long AbsoluteValue => Math.Abs(Convert.ToInt64(To.Signed(Value)));
            2x faster with comparer
                                                                                                 15
            </remarks>
                                                                                                 16
                                                                                                            public Hybrid(T value)
        public class DoubletComparer<T>: IEqualityComparer<Doublet<T>>
                                                                                                 17
                                                                                                 18
1.1
                                                                                                               if (CachedTypeInfo<T>.IsSigned)
           private static readonly EqualityComparer<T> equalityComparer =
                                                                                                 19
12
               EqualityComparer<T>.Default;
                                                                                                 20
                                                                                                                  throw new NotSupportedException();
                                                                                                 21
13
```

```
public static explicit operator sbyte(Hybrid<T> hybrid) =>
                                                                                       73
    \hat{V}alue = value:
                                                                                                 → Convert.ToSBvte(hvbrid.AbsoluteValue):
                                                                                       74
                                                                                                 public override string ToString() => IsNothing? default(T) == null? "Nothing":
                                                                                       75
public Hybrid(object value) => Value =
                                                                                                 → default(T).ToString(): IsExternal? $\B\"<{AbsoluteValue}>\": Value.ToString()
    To. Unsigned As < T > (Convert. Change Type (value.)
                                                                                       76
    CachedTypeInfo<T>SignedVersion)):
                                                                                       77
public Hybrid (object value, bool is External)
                                                                                       ./ILinks.cs
                                                                                          using Platform.Data.Constants;
   var signedType = CachedTypeInfo<T>.SignedVersion;
   var signedValue = Convert.ChangeType(value, signedType);
                                                                                           namespace Platform.Data.Doublets
   var abs = typeof(MathHelpers).GetTypeInfo().GetMethod("Abs").MakeGenericM

→ ethod(signedType):

                                                                                              public interface ILinks<TLink>: ILinks<TLink, LinksCombinedConstants<TLink,
   var negate = typeof(MathHelpers).GetTypeInfo().GetMethod("Negate").MakeGen_1
                                                                                                  TLink, int>>
       ericMethod(signedType):
   var absoluteValue = abs.Invoke(null, new [] { signedValue });
   var resultValue = isExternal? negate.Invoke(null, new[] { absoluteValue }):
       absoluteValue:
   Value = To.UnsignedAs < T > (resultValue):
                                                                                       ./ILinksExtensions.cs
                                                                                           using System:
                                                                                           using System.Collections:
public static implicit operator Hybrid<T>(T integer) => new Hybrid<T>(integer);
                                                                                           using System. Collections. Generic;
                                                                                           using System.Ling;
public static explicit operator Hybrid<T>(ulong integer) => new Hybrid<T>(integer);
                                                                                           using System.Runtime.CompilerServices:
                                                                                           using Platform. Ranges;
public static explicit operator Hybrid<T>(long integer) => new Hybrid<T>(integer);
                                                                                           using Platform.Collections.Arrays;
                                                                                           using Platform. Numbers:
public static explicit operator Hybrid<T>(uint integer) => new Hybrid<T>(integer);
                                                                                           using Platform Random;
                                                                                           using Platform. Helpers. Setters:
public static explicit operator Hybrid<T>(int integer) => new Hybrid<T>(integer);
                                                                                           using Platform Data Exceptions:
                                                                                       12
public static explicit operator Hybrid <T > (ushort integer) => new
                                                                                           namespace Platform.Data.Doublets
                                                                                       13
    Hybrid < T > (integer);
                                                                                       14
                                                                                              public static class ILinksExtensions
                                                                                       15
public static explicit operator Hybrid<T>(short integer) => new Hybrid<T>(integer):
                                                                                       16
                                                                                                 public static void RunRandomCreations<TLink>(this ILinks<TLink> links, long
                                                                                       17
public static explicit operator Hybrid<T>(byte integer) => new Hybrid<T>(integer);
                                                                                                     amount Of Creations)
public static explicit operator Hybrid<T>(sbyte integer) => new Hybrid<T>(integer);
                                                                                      18
                                                                                                    for (long i = 0; i < amountOfCreations; <math>i++)
                                                                                       19
public static implicit operator T(Hybrid<T> hybrid) => hybrid.Value;
                                                                                       20
                                                                                                     var linksAddressRange = new Range < ulong > (0, (Integer < TLink > ) links.Count());
                                                                                      21
public static explicit operator ulong(Hybrid<T> hybrid) =>
                                                                                                       Integer<TLink> source =
                                                                                       22
 → Convert.ToUInt64(hybrid.Value);
                                                                                                       → RandomHelpers.Default.NextUInt64(linksAddressRange);
                                                                                                       Integer<TLink> target =
public static explicit operator long(Hybrid<T> hybrid) => hybrid.AbsoluteValue;
                                                                                                          RandomHelpers.Default.NextUInt64(linksAddressRange);
                                                                                                       links.CreateAndUpdate(source, target);
                                                                                       ^{24}
public static explicit operator uint(Hybrid<T> hybrid) =>
                                                                                       25
    Convert. ToUInt 32(hybrid. Value);
                                                                                       26
                                                                                       27
public static explicit operator int(Hybrid<T> hybrid) =>
                                                                                                 public static void RunRandomSearches<TLink>(this ILinks<TLink> links, long
                                                                                       28
    Convert. ToInt32(hybrid. AbsoluteValue);
                                                                                                     amount Of Searches)
public static explicit operator ushort(Hybrid<T> hybrid) =>
                                                                                                    for (long i = 0; i < amountOfSearches; i++)
                                                                                       30
    Convert. ToUInt16(hybrid. Value):
                                                                                       31
                                                                                                      var linkAddressRange = new Range < ulong > (1, (Integer < TLink > ) links.Count());
                                                                                       32
public static explicit operator short(Hybrid<T> hybrid) =>
                                                                                                       Integer<TLink> source =
                                                                                       33
 → Convert.ToInt16(hybrid.AbsoluteValue):
                                                                                                       → RandomHelpers.Default.NextUInt64(linkAddressRange);
                                                                                                       Integer<TLink> target =
                                                                                       34
public static explicit operator byte(Hybrid<T> hybrid) =>
                                                                                                           RandomHelpers.Default.NextUInt64(linkAddressRange);
    Convert. ToByte(hybrid. Value);
                                                                                                       links.SearchOrDefault(source, target);
                                                                                       35
                                                                                       36
```

26

27

31

41

42

43

44

45

47

52

5.5

56

57

58

5.9

61

62

```
9.4
public static void RunRandomDeletions<TLink>(this ILinks<TLink> links, long
                                                                                        95
    amount Of Deletions)
                                                                                        96
                                                                                        97
  var min = (ulong)amountOfDeletions > (Integer<TLink>)links.Count() ? 1:
                                                                                        98
   → (Integer<TLink>)links.Count() - (ulong)amountOfDeletions;
  for (long i = 0: i < amountOfDeletions: <math>i++)
                                                                                        aa
                                                                                       100
     var linksAddressRange = new Range<ulong>(min.
                                                                                       101
                                                                                       102
         (Integer < TLink > ) links. Count()):
                                                                                       103
    Integer < TLink > link = RandomHelpers.Default.NextUInt64(linksAddressRange)
                                                                                       104
     links.Delete(link):
                                                                                       105
     if ((Integer < TLink >) links.Count() < min)
                                                                                       106
         break:
                                                                                       107
                                                                                       108
                                                                                       109
    <remarks>
                                                                                       110
    ТОДО: Возможно есть очень простой способ это сдедать.
    (Например просто удалить файл, или изменить его размер таким образом,
                                                                                       111
    чтобы удалился весь контент)
                                                                                       112
   Hапример через header->AllocatedLinks в ResizableDirectMemoryLinks
                                                                                       113
    </remarks>
                                                                                       114
public static void DeleteAll<TLink>(this ILinks<TLink> links)
                                                                                       115
                                                                                       116
   var equalityComparer = EqualityComparer < TLink > . Default:
                                                                                       117
  var comparer = Comparer < TLink > . Default;
                                                                                       118
  for (var i = links.Count(); comparer.Compare(i, default) > 0; i =
                                                                                       119
       ArithmeticHelpers.Decrement(i))
                                                                                       120
                                                                                       121
     links.Delete(i):
                                                                                       122
     if (!equalityComparer.Equals(links.Count(), ArithmeticHelpers.Decrement(i)))
                                                                                       123
                                                                                       124
         i = links.Count():
                                                                                       125
                                                                                       126
                                                                                       127
                                                                                       128
public static TLink First<TLink>(this ILinks<TLink> links)
                                                                                       129
   TLink firstLink = default;
                                                                                       130
   var equalityComparer = ÉqualityComparer < TLink > . Default:
                                                                                       131
  if (equalityComparer.Equals(links.Count(), default))
                                                                                       132
     throw new Exception("В хранилище нет связей.");
                                                                                       133
   links.Each(links.Constants.Anv. links.Constants.Anv. link =>
                                                                                       134
                                                                                       135
     firstLink = link[links.Constants.IndexPart];
                                                                                       136
     return links.Constants.Break;
                                                                                       137
                                                                                       138
     (equalityComparer.Equals(firstLink, default))
                                                                                       139
                                                                                       140
     throw new Exception("В процессе поиска по хранилищу не было найдено
                                                                                       141
      → связей."):
                                                                                       142
  return firstLink;
                                                                                       143
```

41

44

5.1

52

53

54

55

60

63

71 72

73

74

75

77

8.1

91

```
public static bool IsInnerReference<TLink>(this ILinks<TLink> links, TLink
    reference)
   var constants = links.Constants;
  var comparer = Comparer < TLink > . Default:
  return comparer.Compare(constants.MinPossibleIndex, reference) >= 0 &&
   → comparer.Compare(reference, constants.MaxPossibleIndex) <= 0;
#region Paths
   <remarks>
   ТООО: Как так? Как то что ниже может быть корректно?
   Скорее всего практически не применимо
   Предполагалось, что можно было конвертировать формируемый в проходе
   через SequenceWalker
   Stack в конкретный путь из Source. Target до связи, но это не всегда так.
   TODO: Возможно нужен метод, который именно выбрасывает исключения

→ (EnsurePathExists)

   </remarks>
public static bool CheckPathExistance<TLink>(this ILinks<TLink> links, params
   TLink[] path)
   var current = path[0]:
   //EnsureLinkExists(current, "path");
   if (!links.Exists(current))
     return false:
   var equalityComparer = EqualityComparer < TLink > . Default:
   var constants = links.Constants:
   for (var i = 1: i < path. Length: i++)
     var next = path[i];
     var values = links.GetLink(current);
     var source = values[constants.SourcePart];
     var target = values[constants.TargetPart];
     if (equalityComparer.Equals(source, target) && equalityComparer.Equals(source,
         next))
        //throw new Exception(string.Format("Невозможно выбрать путь, так как
        \rightarrow и Source и Target совпадают с элементом пути \{0\}.", next));
        return false;
      if (!equalityComparer.Equals(next, source) && !equalityComparer.Equals(next,
         target))
        //throw new Exception(string.Format("Невозможно продолжить путь через
        → элемент пути {0}", next));
        return false:
     return true;
   <remarks>
/// Может потребовать дополнительного стека для PathElement's при
   использовании SequenceWalker.
/// </remarks>
```

```
public static TLink GetByKeys<TLink>(this ILinks<TLink> links, TLink root,
144
               params int [] path)
145
              links.EnsureLinkExists(root, "root");
146
                                                                                                 201
              var currentLink = root;
147
                                                                                                 202
              for (var i = 0; i < path.Length; i++)
148
149
                 currentLink = links.GetLink(currentLink)[path[i]];
150
                                                                                                 204
                                                                                                 205
              return currentLink:
152
                                                                                                 206
153
                                                                                                 207
154
                                                                                                 208
           public static TLink GetSquareMatrixSequenceElementByIndex<TLink>(this
155
                                                                                                 209
              ILinks<TLink> links, TLink root, ulong size, ulong index)
                                                                                                 210
156
                                                                                                 211
              var constants = links.Constants;
157
              var source = constants.SourcePart:
158
                                                                                                 212
              var target = constants. Target Part:
159
                                                                                                 213
              if (!MathHelpers.IsPowerOfTwo(size))
160
                                                                                                 214
161
                                                                                                 215
                 throw new ArgumentOutOfRangeException(nameof(size), "Sequences with sizes
162
                 → other than powers of two are not supported."):
                                                                                                 217
163
              var path = new BitArray(BitConverter.GetBytes(index)):
164
                                                                                                 218
              var length = BitwiseHelpers.GetLowestBitPosition(size):
                                                                                                 219
              links.EnsureLinkExists(root, "root");
166
                                                                                                 220
              var currentLink = root;
167
              for (var i = length - 1; i >= 0; i--)
168
                                                                                                 221
                                                                                                 222
                 currentLink = links.GetLink(currentLink)[path[i] ? target : source];
170
                                                                                                 223
              return currentLink:
172
                                                                                                 224
173
                                                                                                 225
174
                                                                                                 226
           #endregion
175
                                                                                                 227
176
              summary>
177
               Возвращает индекс указанной связи.
179
               <param name="links">Хранилище связей </param>
                                                                                                 228
           /// <param name="link">Связь представленная списком, состоящим из её адреса
181
              и содержимого.</param>
                                                                                                 229
               <returns>Индекс начальной связи для указанной связи.</returns>
                                                                                                 230
182
           [MethodImpl(MethodImplOptions.AggressiveInlining)]
183
           public static TLink GetIndex<TLink>(this ILinks<TLink> links, IList<TLink> link)
184
           \rightarrow => link[links.Constants.IndexPart];
                                                                                                 232
               <summary>
                                                                                                 233
186
               Возвращает индекс начальной (Source) связи для указанной связи.
187
                                                                                                 234
               </summary>
               <param name="links">Хранилище связей.</param>
189
               <param name="link">Индекс связи.</param>
190
                                                                                                 236
               <returns>Индекс начальной связи для указанной связи </returns>
191
                                                                                                 237
           [MethodImpl(MethodImplOptions.AggressiveInlining)]
192
           public static TLink GetSource<TLink>(this ILinks<TLink> links, TLink link) =>
193
           → links.GetLink(link)[links.Constants.SourcePart];
                                                                                                 238
194
               <summary>
195
               Возвращает индекс начальной (Source) связи для указанной связи.
196
                                                                                                 239
197
                                                                                                 240
               <param name="links">Хранилище связей.</param>
198
```

```
/// <param name="link">Связь представленная списком, состоящим из её адреса
   и содержимого.</param>
  / <returns>Индекс начальной связи для указанной связи.</returns>
[MethodImpl(MethodImplOptions, AggressiveInlining)]
public static TLink GetSource<TLink>(this ILinks<TLink> links, IList<TLink>
⇒ link) => linkllinks.Constants.SourcePartl:
   <summary>
   Возвращает индекс конечной (Target) связи для указанной связи.
   </summary>
   <param name="links">Хранилише связей.</param>
   <param name="link">Индекс связи.</param>
   <returns>Индекс конечной связи для указанной связи.</returns>
[MethodImpl(MethodImplOptions.AggressiveInlining)]
public static TLink GetTarget < TLink > (this ILinks < TLink > links, TLink link) =>
→ links.GetLink(link)[links.Constants.TargetPart];
  / <summary>
   Возвращает индекс конечной (Target) связи для указанной связи.
   </summary>
   <param name="links">Хранилише связей.</param>
 // <param name="link">Связь представленная списком, состоящим из её адреса
→ и содержимого.</param>
^{\prime}//<returns>Индекс конечной связи для указанной связи.</returns>
[MethodImpl(MethodImplOptions.AggressiveInlining)]
public static TLink GetTarget<TLink>(this ILinks<TLink> links, IList<TLink>
\rightarrow link) => link[links.Constants.TargetPart]:
/// <summary>
   Выполняет проход по всем связям, соответствующим шаблону, вызывая
→ обработчик (handler) для каждой подходящей связи.
   </summary>
   <param name="links">Хранилище связей.</param>
   <param name="handler">Обработчик каждой подходящей связи </param>
 // <param name="restrictions">Ограничения на содержимое связей. Каждое
    ограничение может иметь значения: Constants.Null - 0-я связь, обозначающая
   ссылку на пустоту. Any - отсутствие ограничения. 1..∞ конкретный адрес
   связи.</param>
/// <returns>True, в случае если проход по связям не был прерван и False в
→ обратном случае.</returns>
[MethodImpl(MethodImplOptions.AggressiveInlining)]
public static bool Each<TLink>(this ILinks<TLink> links. Func<IList<TLink>.
→ TLink> handler, params TLink[] restrictions)
  => EqualityComparer<TLink>.Default.Equals(links.Each(handler, restrictions),
   → links.Constants.Continue);
   Выполняет проход по всем связям, соответствующим шаблону, вызывая
   обработчик (handler) для каждой подходящей связи.
   </summary>
   <param name="links">Хранилище связей.</param>
  / <param name="source">Значение, определяющее соответствующие шаблону
    связи. (Constants.Null - 0-я связь, обозначающая ссылку на пустоту в качестве
    начала, Constants. Any - любое начало, 1..\infty конкретное начало) < /param>
   <param name="target">Значение, определяющее соответствующие шаблону
    связи. (Constants.Null - 0-я связь, обозначающая ссылку на пустоту в качестве
    конца. Constants.Anv - любой конец. 1..\infty конкретный конец) 
   <param name="handler">Обработчик каждой подходящей связи </param>
/// <returns>True, в случае если проход по связям не был прерван и False в
→ обратном случае.</returns>
```

```
[MethodImpl(MethodImplOptions.AggressiveInlining)]
241
                                                                                               287
           public static bool Each TLink (this ILinks TLink) links, TLink source, TLink
                                                                                               288
242

    target, Func<TLink, bool> handler)

    target, Func<TLink, bool> handler)
                                                                                               289
                                                                                               290
243
             var constants = links.Constants:
                                                                                               291
244
             return links. Each(link => handler(link|constants.IndexPart|)? constants.Continue:
245
              202
                                                                                               293
                                                                                               294
247
              <summary>
248
              Выполняет проход по всем связям, соответствующим шаблону, вызывая
249
              обработчик (handler) для каждой подходящей связи.
                                                                                               296
               </summary>
                                                                                               297
250
               <param name="links">Хранилище связей.</param>
251
              <pаram name="source">Значение, определяющее соответствующие шаблону
                                                                                               299
252
                                                                                              300
               связи. (Constants.Null - 0-я связь, обозначающая ссылку на пустоту в качестве
               начала, Constants. Any - любое начало, 1..\infty конкретное начало) 
               <param name="target">Значение, определяющее соответствующие шаблону
                                                                                               301
253
                                                                                               302
               связи. (Constants.Null - 0-я связь, обозначающая ссылку на пустоту в качестве
                                                                                               303
               конца. Constants. Anv - любой конец. 1... конкретный конец) 
                                                                                               304
               <param name="handler">Обработчик каждой подходящей связи.</param>
254
                                                                                               305
              <returns>True, в случае если проход по связям не был прерван и False в
255
                                                                                               306
              обратном случае.</returns>
                                                                                               307
           [MethodImpl(MethodImplOptions.AggressiveInlining)]
                                                                                               308
           public static bool Each<TLink>(this ILinks<TLink> links, TLink source, TLink
257
                                                                                               309
              target, Func<IList<TLink>, TLink> handler)
258
                                                                                               310
             var constants = links.Constants:
259
                                                                                               311
             return links. Each (handler, constants. Any, source, target);
260
                                                                                               312
261
                                                                                               313
262
                                                                                               314
           [MethodImpl(MethodImplOptions.AggressiveInlining)]
263
                                                                                               315
           public static IList<IList<TLink>> All<TLink>(this ILinks<TLink> links, params
264
                                                                                               316
           → TLink[] restrictions)
                                                                                               317
265
                                                                                               318
             var constants = links.Constants:
266
             int listSize = (Integer<TLink>)links.Count(restrictions);
267
                                                                                               319
             var list = new IList < TLink > [listSize];
268
                                                                                               320
             if (listSize > 0)
                                                                                               321
270
                                                                                               322
                var filler = new ArrayFiller < IList < TLink >, TLink > (list,
271
                                                                                               323
                 → links.Constants.Continue):
                links.Each(filler.AddAndReturnConstant, restrictions):
272
                                                                                               324
273
                                                                                               325
             return list:
274
                                                                                               326
275
                                                                                               327
276
                                                                                               328
              <summary>
277
              Возвращает значение, определяющее существует ли связь с указанными
278
                                                                                               329
           ↔ началом и концом в хранилище связей.
                                                                                               330
               </summary>
279
                                                                                               331
               <param name="links">Хранилище связей.</param>
280
                                                                                               332
               <param name="source">Начало связи.</param>
281
                                                                                               333
               <param name="target">Конец связи.</param>
282
              <returns>Значение, определяющее существует ли связь.</returns>
283
                                                                                               334
           [MethodImpl(MethodImplOptions.AggressiveInlining)]
                                                                                               335
           public static bool Exists TLink > (this ILinks TLink > links, TLink source, TLink
                                                                                               336
               target) = >
                                                                                               337
               Comparer<TLink>.Default.Compare(links.Count(links.Constants.Any, source,
                                                                                               338
               target), default) > 0:
286
```

```
#region Ensure
 // TODO: May be move to EnsureExtensions or make it both there and here
[MethodImpl(MethodImplOptions, AggressiveInlining)]
public static void EnsureInnerReferenceExists<TLink>(this ILinks<TLink> links,
→ TLink reference, string argumentName)
   if (links.IsInnerReference(reference) && !links.Exists(reference))
     throw new ArgumentLinkDoesNotExistsException<TLink>(reference,

→ argument Name):

[MethodImpl(MethodImplOptions.AggressiveInlining)]
public static void EnsureInnerReferenceExists<TLink>(this ILinks<TLink> links,
→ IList<TLink> restrictions, string argumentName)
   for (int i = 0; i < restrictions.Count; i++)
     links.EnsureInnerReferenceExists(restrictions[i], argumentName);
[MethodImpl(MethodImplOptions, AggressiveInlining)]
public static void EnsureLinkIsAnyOrExists<TLink>(this ILinks<TLink> links,
→ IList<TLink> restrictions)
   for (int i = 0; i < restrictions.Count; i++)
     links.EnsureLinkIsAnyOrExists(restrictions[i], nameof(restrictions));
[MethodImpl(MethodImplOptions.AggressiveInlining)]
public static void EnsureLinkIsAnyOrExists<TLink>(this ILinks<TLink> links,
→ TLink link, string argumentName)
   var equalityComparer = EqualityComparer < TLink > . Default;
   if (!equalityComparer.Equals(link, links.Constants.Any) && !links.Exists(link))
     throw new ArgumentLinkDoesNotExistsException<TLink>(link,
      \rightarrow argument Name);
[MethodImpl(MethodImplOptions.AggressiveInlining)]
public static void EnsureLinkIsItselfOrExists<TLink>(this ILinks<TLink> links,
→ TLink link, string argumentName)
   var equalityComparer = EqualityComparer < TLink > . Default:
   if (!equalityComparer.Equals(link, links.Constants.Itself) && !links.Exists(link))
     throw new ArgumentLinkDoesNotExistsException<TLink>(link,

→ argument Name);

/// <param_name="links">Хранилише связей.</param>
[MethodImpl(MethodImplOptions.AggressiveInlining)]
```

```
public static void EnsureDoesNotExists<TLink>(this ILinks<TLink> links, TLink
                                                                                                 394
339
               source, TLink target)
                                                                                                 395
                                                                                                 396
340
                (links.Exists(source, target))
341
                                                                                                 397
342
                 throw new LinkWithSameValueAlreadyExistsException():
                                                                                                  398
343
344
                                                                                                  399
                                                                                                  400
345
346
                                                                                                  401
           /// <param name="links">Хранилище связей.</param>
347
                                                                                                  402
           public static void EnsureNoDependencies<TLink>(this ILinks<TLink> links, TLink
348
349
                                                                                                  404
                (links.DependenciesExist(link))
350
                                                                                                  405
351
                                                                                                  406
                 throw new ArgumentLinkHasDependenciesException<TLink>(link);
                                                                                                  407
352
353
                                                                                                  408
                                                                                                  409
354
                                                                                                  410
355
           /// <param name="links">Хранилище связей.</param>
356
                                                                                                 411
           public static void EnsureCreated<TLink>(this ILinks<TLink> links, params TLink)
                                                                                                 412
               addresses) => links.EnsureCreated(links.Create, addresses);
                                                                                                 413
358
           /// <param name="links">Хранилише связей.</param>
359
                                                                                                 414
           public static void EnsurePointsCreated<TLink>(this ILinks<TLink> links, params
                                                                                                 415
           → TLink[] addresses) => links.EnsureCreated(links.CreatePoint, addresses);
                                                                                                 416
361
           /// <param_name="links">Хранилище связей.</param>
362
                                                                                                 417
           public static void EnsureCreated<TLink>(this ILinks<TLink> links, Func<TLink>
363
                                                                                                  418
           419
                                                                                                  420
364
              var constants = links.Constants;
                                                                                                  421
              var nonExistentAddresses = new HashSet < ulong > (addresses.Where(x = >
366
                  !links.Exists(x)).Select(x => (ulong)(Integer<TLink>)x)):
                                                                                                  422
              if (nonExistentAddresses.Count > 0)
                                                                                                  423
367
                                                                                                  424
368
                                                                                                  425
                 var max = nonExistentAddresses.Max();
369
                                                                                                  426
                 // TODO: Эту верхнюю границу нужно разрешить переопределять
                                                                                                  427
                    (проверить применяется ли эта логика)
                                                                                                  428
                 max = Math.Min(max, (Integer<TLink>)constants.MaxPossibleIndex);
371
                var createdLinks = new List < TLink > ():
                 var equalityComparer = EqualityComparer < TLink > . Default;
                                                                                                  429
373
                 TLink createdLink = creator():
374
                 while (!equalityComparer.Equals(createdLink, (Integer<TLink>)max))
                                                                                                  430
375
                    createdLinks.Add(createdLink);
377
                                                                                                  431
378
                                                                                                  432
                 for (\text{var i} = 0; \text{ i} < \text{createdLinks.Count}; \text{ i}++)
379
                                                                                                  433
                    if (!nonExistentAddresses.Contains((Integer<TLink>)createdLinks[i]))
381
                                                                                                  434
                                                                                                  435
                       links.Delete(createdLinks[i]);
384
                                                                                                  436
385
                                                                                                  437
386
                                                                                                  438
387
                                                                                                  439
388
                                                                                                  440
           #endregion
389
                                                                                                  441
390
                                                                                                 442
           /// <param name="links">Хранилише связей.</param>
391
          public static ulong DependenciesCount<TLink>(this ILinks<TLink> links, TLink link)
392
393
```

```
var constants = links.Constants;
  var values = links.GetLink(link):
   ulong referencesAsSource = (Integer<TLink>)links.Count(constants.Any, link,

→ constants.Anv):

   var equalityComparer = EqualityComparer < TLink > . Default:
   if (equalityComparer.Equals(values[constants.SourcePart], link))
     referencesAsSource--:
   ulong referencesAsTarget = (Integer<TLink>)links.Count(constants.Any,
   if (equalityComparer.Equals(values[constants.TargetPart], link))
     referencesAsTarget--;
   return referencesAsSource + referencesAsTarget;
 /// <param name="links">Хранилище связей.</param>
[MethodImpl(MethodImplOptions.AggressiveInlining)]
public static bool DependenciesExist<TLink>(this ILinks<TLink> links, TLink link)
\Rightarrow => links.DependenciesCount(link) > 0:
/// <param name="links">Хранилище связей.</param>
[MethodImpl(MethodImplOptions.AggressiveInlining)]
public static bool Equals<TLink>(this ILinks<TLink> links, TLink link, TLink
→ source, TLink target)
   var constants = links.Constants;
   var values = links.GetLink(link)
   var equalityComparer = EqualityComparer < TLink > . Default;
  return equalityComparer.Equals(values[constants.SourcePart], source) &&

→ equalityComparer.Equals(values[constants.TargetPart], target);

   <summary>
   Выполняет поиск связи с указанными Source (началом) и Target (концом).
   </summary>
   <param name="links">Хранилище связей.</param>
   <param name="source">Индекс связи, которая является началом для искомой
   связи.</param>
   срагат name="target">Индекс связи, которая является концом для искомой
    связи.</param>
/// <returns>Индекс искомой связи с указанными Source (началом) и Target
    (концом).</returns>
[MethodImpl(MethodImplOptions.AggressiveInlining)]
public static TLink SearchOrDefault < TLink > (this ILinks < TLink > links, TLink
→ source, TLink target)
   var contants = links.Constants;
  var setter = new Setter < TLink, TLink > (contants.Continue, contants.Break,
  links. Each (setter. Set First And Return False, contants. Any, source, target);
   return setter Result:
/// <param name="links">Хранилище связей.</param>
[MethodImpl(MethodImplOptions.AggressiveInlining)]
public static TLink CreatePoint<TLink>(this ILinks<TLink> links)
  var link = links.Create();
```

```
return links. Update(link, link, link);
                                                                                               493
446
447
              <param name="links">Хранилише связей.</param>
448
           [MethodImpl(MethodImplOptions.AggressiveInlining)]
449
                                                                                               495
           public static TLink CreateAndUpdate<TLink>(this ILinks<TLink> links, TLink
450

→ source, TLink target) => links.Update(links.Create(), source, target);

451
                                                                                               497
              <summarv>
452
                                                                                               498
              Обновляет связь с указанными началом (Source) и концом (Target)
453
                                                                                               499
              на связь с указанными началом (NewSource) и концом (NewTarget).
                                                                                               500
              </summary>
455
                                                                                               501
               <param name="links">Хранилище связей.</param>
456
                                                                                               502
               <param name="link">Индекс обновляемой связи.</param>
                                                                                               503
            // <param name="newSource">Индекс связи, которая является началом связи.
458
                                                                                               504
              на которую выполняется обновление.</param>
              <pагат name="newTarget">Индекс связи, которая является концом связи, на
              которую выполняется обновление.
                                                                                               507
              <returns>Индекс обновлённой связи.</returns>
460
                                                                                               508
           [MethodImpl(MethodImplOptions.AggressiveInlining)]
                                                                                               509
           public static TLink Update<TLink>(this ILinks<TLink> links, TLink link, TLink
462
                                                                                               510
               newSource, TLink newTarget) => links.Update(new[] { link, newSource,
                                                                                              511
               newTarget }):
463
                                                                                               512
              <summary>
              Обновляет связь с указанными началом (Source) и концом (Target)
465
                                                                                               513
              на связь с указанными началом (NewSource) и концом (NewTarget).
              </summary>
467
                                                                                              514
              <param name="links">Хранилише связей.</param>
468
              <param name="restrictions">Ограничения на содержимое связей. Каждое
                                                                                               515
               ограничение может иметь значения: Constants.Null - 0-я связь, обозначающая
                                                                                               516
               ссылку на пустоту, Itself - требование установить ссылку на себя, 1..\infty
                                                                                              517
               конкретный адрес другой связи. </param>
               <returns>Индекс обновлённой связи.</returns>
470
                                                                                               518
           [MethodImpl(MethodImplOptions.AggressiveInlining)]
471
                                                                                               519
           public static TLink Update < TLink > (this ILinks < TLink > links, params TLink |
472
                                                                                               520
               restrictions)
                                                                                               521
                                                                                               522
             if (restrictions.Length == 2)
474
                                                                                               523
475
                                                                                               524
                return links.Merge(restrictions[0], restrictions[1]);
476
                                                                                               525
477
               (restrictions.Length == 4)
                                                                                               526
479
                                                                                               527
                return links.UpdateOrCreateOrGet(restrictions[0], restrictions[1], restrictions[2]
                 \rightarrow restrictions[3]);
                                                                                               529
481
                                                                                               530
             else
482
                                                                                               531
                return links. Update(restrictions);
485
                                                                                               533
486
                                                                                               534
487
488
              Создаёт связь (если она не существовала), либо возвращает индекс
               существующей связи с указанными Source (началом) и Target (концом).
                                                                                               536
                                                                                               537
               <param name="links">Хранилище связей.</param>
491
              <param name="source">Индекс связи, которая является началом на
492
                                                                                               538
           → создаваемой связи.</param>
                                                                                               539
                                                                                               540
```

```
/// <param name="target">Индекс связи, которая является концом для
    создаваемой связи.</param>
/// <returns>Индекс связи, с указанным Source (началом) и Target
   (концом)</returns>
[MethodImpl(MethodImplOptions.AggressiveInlining)]
public static TLink GetOrCreate<TLink>(this ILinks<TLink> links, TLink source,
   TLink target)
   var link = links.SearchOrDefault(source, target):
   if (EqualityComparer<TLink>.Default.Equals(link, default))
     link = links.CreateAndUpdate(source, target);
   return link:
   <summary>
   Обновляет связь с указанными началом (Source) и конпом (Target)
   на связь с указанными началом (NewSource) и концом (NewTarget).
   </summary>
   <param name="links">Хранилише связей.</param>
   ¬< param name="source">Индекс связи, которая является началом
   обновляемой связи.</param>
   <param name="target" >Индекс связи, которая является концом обновляемой
    связи.</param>
/// <param name="newSource">Индекс связи, которая является началом связи,
   на которую выполняется обновление. </param>
   <рагат name="newTarget">Индекс связи, которая является концом связи, на
→ которую выполняется обновление.
   <returns>Индекс обновлённой связи.</returns>
[MethodImpl(MethodImplOptions.AggressiveInlining)]
public static TLink UpdateOrCreateOrGet<TLink>(this ILinks<TLink> links, TLink
    source, TLink target, TLink newSource, TLink newTarget)
   var equalityComparer = EqualityComparer < TLink > . Default:
   var link = links.SearchOrDefault(source, target);
  if (equalityComparer.Equals(link, default))
     return links.CreateAndUpdate(newSource, newTarget);
   if (equalityComparer.Equals(newSource, source) &&
      equalityComparer.Equals(newTarget, target))
     return link;
   return links. Update(link, newSource, newTarget);
/// <summary>Удаляет связь с указанными началом (Source) и концом
    (Target).</summary>
   <param name="links">Хранилище связей.</param>
   <param name="source">Йндекс связи, которая является началом удаляемой
   связи.</param>
/// <param name="target">Индекс связи, которая является концом удаляемой
↔ связи.</param>
[MethodImpl(MethodImplOptions, AggressiveInlining)]
public static TLink DeleteIfExists<TLink>(this ILinks<TLink> links, TLink source,
   TLink target)
   var link = links.SearchOrDefault(source, target);
  if (!EqualityComparer<TLink> Default.Equals(link, default))
```

```
if (equalityComparer.Equals(reference, linkIndex))
                                                                                                 594
541
                 links.Delete(link):
542
                                                                                                 595
543
                 return link:
                                                                                                                          continue:
                                                                                                 596
544
                                                                                                 597
              return default;
                                                                                                 598
545
                                                                                                                       links.Update(reference, links.GetSource(reference), newLink);
5.46
                                                                                                 600
547
               <summary>Удаляет несколько связей.</summary>
                                                                                                                     ArrayPool.Free(references);
548
                                                                                                 601
               <param name="links">Хранилище связей </param>
549
                                                                                                 602
               <param name="deletedLinks">Список адресов связей к удалению
550
                                                                                                 603
           [MethodImpl(MethodImplOptions.AggressiveInlining)]
                                                                                                               links.Delete(linkIndex);
551
                                                                                                 604
           public static void DeleteMany<TLink>(this ILinks<TLink> links. IList<TLink>
                                                                                                               return newLink:
                                                                                                 605
552
               deletedLinks)
                                                                                                 606
                                                                                                 607
553
              for (int i = 0; i < deletedLinks.Count; i++)
                                                                                                 608
554
555
                 links.Delete(deletedLinks[i]);
556
                                                                                                  ./Incrementers/FrequencyIncrementer.cs
                                                                                                      using System. Collections. Generic:
558
                                                                                                      using Platform.Interfaces:
559
           // Replace one link with another (replaced link is deleted, children are updated or
560
                                                                                                      namespace Platform.Data.Doublets.Incrementers
           public static TLink Merge<TLink>(this ILinks<TLink> links, TLink linkIndex,
561
                                                                                                         public class FrequencyIncrementer<TLink>: LinksOperatorBase<TLink>,
               TLink newLink)
                                                                                                             IIncrementer<TLink>
562
              var equalityComparer = EqualityComparer < TLink > . Default:
563
                                                                                                            private static readonly EqualityComparer<TLink> equalityComparer =
              if (equalityComparer.Equals(linkIndex, newLink))
564
                                                                                                                EqualityComparer<TLink>.Default;
565
                 return newLink;
                                                                                                            private readonly TLink frequency Marker;
                                                                                                  10
567
                                                                                                            private readonly TLink unaryOne;
                                                                                                  11
              var constants = links.Constants:
                                                                                                            private readonly IIncrementer<TLink> unaryNumberIncrementer;
                                                                                                  12
              ulong referencesAsSourceCount = (Integer < TLink > ) links.Count(constants.Any,
569
                                                                                                  13
                 linkIndex, constants, Anv):
                                                                                                            public FrequencyIncrementer(ILinks<TLink> links, TLink frequencyMarker, TLink
                                                                                                  14
              ulong referencesAsTargetCount = (Integer<TLink>)links.Count(constants.Any,
                                                                                                            → unaryOne, IIncrementer<TLink> unaryNumberIncrementer)
              → constants.Anv, linkIndex);
                                                                                                               : base(links)
                                                                                                  15
              var isStandalonePoint = Point < TLink > .IsFullPoint (links.GetLink (linkIndex)) &&
571
                                                                                                  16
                                                                                                                frequency Marker = frequency Marker:
                  referencesAsSourceCount == 1 \&\& referencesAsTargetCount == 1;
                                                                                                  17
                                                                                                                unarvOne = unarvOne:
              if (!isStandalonePoint)
                                                                                                  18
                                                                                                                unary Number Incrementer = unary Number Incrementer;
                                                                                                  19
573
                 var totalReferences = referencesAsSourceCount + referencesAsTargetCount;
                                                                                                  20
574
                                                                                                  21
                if (totalReferences > 0)
575
                                                                                                            public TLink Increment(TLink frequency)
                                                                                                  22
                    var references = ArrayPool.Allocate<TLink>((long)totalReferences);
                                                                                                  23
577
                                                                                                               if (equalityComparer.Equals(frequency, default))
                    var referencesFiller = new ArrayFiller < TLink, TLink > (references,
                                                                                                  ^{24}
                                                                                                  25
                       links.Constants.Continue);
                                                                                                                  return Links.GetOrCreate( unaryOne, frequencyMarker);
                                                                                                  26
                    links. Each (references Filler. Add First And Return Constant, constants. Any,
579
                                                                                                  27
                    var source = Links.GetSource(frequency);
                                                                                                  28
                    links.Each(referencesFiller.AddFirstAndReturnConstant, constants.Any,
                                                                                                               var incrementedSource = unaryNumberIncrementer.Increment(source);
                                                                                                  29
                    → constants.Anv. linkIndex):
                                                                                                               return Links.GetOrCreate(incrementedSource, frequencyMarker);
                                                                                                  30
                    for (ulong i = 0; i < referencesAsSourceCount; <math>i++)
                                                                                                 31
582
                                                                                                  32
                      var reference = references[i];
583
                                                                                                  33
                      if (equalityComparer,Equals(reference, linkIndex))
584
                         continue;
                                                                                                  ./Incrementers/LinkFrequencyIncrementer.cs
586
                                                                                                      using System.Collections.Generic;
588
                                                                                                      using Platform.Interfaces;
                      links.Update(reference, newLink, links.GetTarget(reference));
                                                                                                      namespace Platform. Data. Doublets. Incrementers
                    for (var i = (long)referencesAsSourceCount; i < references.Length; i++)
591
592
                                                                                                         public class LinkFrequencyIncrementer<TLink>: LinksOperatorBase<TLink>,
                      var reference = references[i];
593
                                                                                                         → IIncrementer<IList<TLink>>
```

```
private readonly ISpecificPropertyOperator<TLink, TLink>
                                                                                                          else
                                                                                             26
                frequencyPropertyOperator:
          private readonly Incrementer < TLink > frequency Incrementer;
                                                                                                             return Links.GetOrCreate(source, Increment(target));
                                                                                             29
          public LinkFrequencyIncrementer(ILinks<TLink> links,
11
                                                                                             30
             ISpecificPropertyOperator<TLink, TLink> frequencyPropertyOperator,
                                                                                             31
             IIncrementer < TLink > frequency Incrementer)
                                                                                             32
            : base(links)
                                                                                             ./ISynchronizedLinks.cs
              frequencyPropertyOperator = frequencyPropertyOperator;
14
              frequencyIncrementer = frequencyIncrementer;
                                                                                                 using Platform.Data.Constants;
                                                                                                 namespace Platform.Data.Doublets
          /// <remarks>Sequence itseft is not changed, only frequency of its doublets is
                                                                                                    public interface ISynchronizedLinks<TLink>: ISynchronizedLinks<TLink,
              incremented.</remarks>
                                                                                                        ILinks<TLink>, LinksCombinedConstants<TLink, TLink, int>>, ILinks<TLink>
          public IList<TLink> Increment(IList<TLink> sequence) // TODO: May be move to
              ILinksExtensions or make SequenceDoubletsFrequencyIncrementer
            for (var i = 1; i < sequence.Count; i++)
21
22
               Increment(Links.GetOrCreate(sequence[i - 1], sequence[i]));
                                                                                             ./Link.cs
24
                                                                                                 using System:
            return sequence;
25
                                                                                                 using System.Collections:
                                                                                                 using System.Collections.Generic:
                                                                                                 using Platform. Exceptions:
          public void Increment(TLink link)
28
                                                                                                 using Platform Ranges:
                                                                                                 using Platform Helpers Singletons;
            var previousFrequency = frequencyPropertyOperator.Get(link);
                                                                                                 using Platform.Data.Constants:
            var frequency = frequency Incrementer. Increment (previous Frequency);
              frequency Property Operator. Set(link, frequency);
                                                                                                 namespace Platform.Data.Doublets
33
34
                                                                                                     /// <summary>
                                                                                             11
35
                                                                                                        Структура описывающая уникальную связь.
                                                                                             13
./Incrementers/UnaryNumberIncrementer.cs
                                                                                                    public struct Link<TLink>: IEquatable<Link<TLink>>, IReadOnlyList<TLink>,
    using System.Collections.Generic:
                                                                                                        IList<TLink>
    using Platform.Interfaces;
                                                                                             15
                                                                                                       public static readonly Link<TLink> Null = new Link<TLink>();
                                                                                             16
    namespace Platform.Data.Doublets.Incrementers
                                                                                             17
                                                                                                       private static readonly LinksCombinedConstants<br/>
bool, TLink, int> constants =
       public class UnaryNumberIncrementer<TLink>: LinksOperatorBase<TLink>,
                                                                                                        → Default<LinksCombinedConstants<br/>
bool, TLink, int>>.Instance;
           IIncrementer<TLink>
                                                                                                       private static readonly EqualityComparer<TLink> equalityComparer =
                                                                                             19
                                                                                                        → EqualityComparer<TLink>.Default;
          private static readonly EqualityComparer<TLink> equalityComparer =
                                                                                             20
          → EqualityComparer<TLink>.Default;
                                                                                                       private const int Length = 3:
                                                                                             21
                                                                                             22
          private readonly TLink unaryOne;
                                                                                                       public readonly TLink Index:
                                                                                             23
                                                                                                       public readonly TLink Source:
11
                                                                                             24
          public UnaryNumberIncrementer(ILinks<TLink> links, TLink unaryOne): base(links)
                                                                                                       public readonly TLink Target;
                                                                                             25
          \rightarrow => unaryOne = unaryOne;
                                                                                             26
                                                                                                       public Link(params TLink[] values)
                                                                                             27
          public TLink Increment(TLink unaryNumber)
                                                                                             28
                                                                                                          Index = values.Length > constants.IndexPart ? values[ constants.IndexPart] :
                                                                                             29
              ( equalityComparer.Equals(unaryNumber, unaryOne))
                                                                                                          Source = values.Length > constants.SourcePart ? values constants.SourcePart :
                                                                                             30
               return Links.GetOrCreate( unaryOne, unaryOne);
                                                                                                          Target = values.Length > constants.TargetPart? values[ constants.TargetPart]:
            var source = Links.GetSource(unaryNumber);
                                                                                                              constants.Null;
            var target = Links.GetTarget(unaryNumber);
                                                                                             32
            if (equalityComparer.Equals(source, target))
                                                                                             33
                                                                                                       public Link(IList<TLink> values)
                                                                                             ^{34}
               return Links.GetOrCreate(unaryNumber, unaryOne);
                                                                                             35
```

```
Index = values.Count > constants.IndexPart? values[ constants.IndexPart]:
                                                                                      91
        constants. Null;
                                                                                       92
   Source = values.Count > constants.SourcePart? values[ constants.SourcePart]:
                                                                                      93
                                                                                       94
         constants. Null:
                                                                                       95
   Target = values.Count > constants.TargetPart ? values[ constants.TargetPart]:
                                                                                       96
       constants.Null;
                                                                                       97
public Link(TLink index, TLink source, TLink target)
                                                                                      98
                                                                                       99
   Index = index:
                                                                                      100
   Source = source:
                                                                                      101
   Target = target;
                                                                                      102
                                                                                      103
                                                                                      104
public Link(TLink source, TLink target)
                                                                                      105
   : this( constants.Null, source, target)
                                                                                      106
                                                                                      107
   Source = source;
                                                                                      108
   Target = target:
                                                                                      109
                                                                                      110
public static Link<TLink> Create(TLink source, TLink target) => new
                                                                                      111
112
                                                                                      113
public override int GetHashCode() => (Index, Source, Target).GetHashCode();
                                                                                      114
                                                                                      115
public bool IsNull() => equalityComparer.Equals(Index, constants.Null)
                                                                                      116
               && equality Comparer. Equals (Source, constants. Null)
                                                                                      117
               && equalityComparer.Equals(Target, constants.Null);
                                                                                      118
                                                                                      119
public override bool Equals(object other) => other is Link<TLink> &&
                                                                                      120
    Equals((Link<TLink>)other):
                                                                                      121
                                                                                      122
public bool Equals(Link<TLink> other) => equalityComparer.Equals(Index,
                                                                                      123
\rightarrow other.Index)
                                                                                      124
                            && equalityComparer.Equals(Source, other.Source)
                                                                                      125
                           && equalityComparer.Equals(Target, other.Target);
                                                                                      126
                                                                                      127
public static string ToString(TLink index, TLink source, TLink target) =>
                                                                                      128
                                                                                      129
    \"({index}: {source}->{target})":
                                                                                      130
public static string ToString(TLink source, TLink target) => $\"({source}->{\target})\";
                                                                                      132
public static implicit operator TLink (Link < TLink > link) => link.ToArray();
                                                                                      133
public static implicit operator Link<TLink>(TLink[| linkArray) => new
                                                                                      134
→ Link<TLink>(linkArray);
                                                                                      135
                                                                                      136
public TLink[] ToArray()
                                                                                      137
                                                                                      138
   var array = new TLink[Length];
                                                                                      139
   CopyTo(array, 0);
                                                                                      140
   return array;
                                                                                      141
                                                                                      142
public override string ToString() => equalityComparer.Equals(Index,
                                                                                      143

→ constants.Null) ? ToString(Source, Target) : ToString(Index, Source, Target);

                                                                                      144
                                                                                      145
#region IList
                                                                                      146
                                                                                      147
public int Count => Length;
                                                                                      148
public bool IsReadOnly = > true;
```

41

42

43

45

46

47

48

40

51

52

53

54

55

56

57

59

6.1

62

63

65

71

72

73 74

75

77

78

81

82

83

84

87

```
public TLink this[int index]
      Ensure. Always. Argument In Range (index, new Range < int > (0, Length - 1),
         nameof(index));
      if (index == constants.IndexPart)
         return Index:
       if (index == constants.SourcePart)
         return Source:
        (index == constants.TargetPart)
         return Target:
      throw new NotSupportedException(); // Impossible path due to
       → Ensure. ArgumentInRange
   set => throw new NotSupportedException();
IEnumerator IEnumerable.GetEnumerator() => GetEnumerator():
public IEnumerator<TLink> GetEnumerator()
   vield return Index:
   vield return Source:
   yield return Target;
public void Add(TLink item) => throw new NotSupportedException();
public void Clear() => throw new NotSupportedException();
public bool Contains(TLink item) => IndexOf(item) >= 0;
public void CopyTo(TLink[] array, int arrayIndex)
   Ensure. Always. Argument Not Null (array, name of (array));
   Ensure. Always. Argument In Range (array Index, new Range < int > (0, array. Length -
   \rightarrow 1), nameof(arrayIndex));
   if (\operatorname{arrayIndex} + \operatorname{Length}) > \operatorname{array.Length})
      throw new InvalidOperationException();
   \operatorname{array}[\operatorname{array}\operatorname{Index}++] = \operatorname{Index};
   \operatorname{array} | \operatorname{arrayIndex} + + | = \operatorname{Source};
   array[arrayIndex] = Target;
public bool Remove(TLink item) =>
    Throw.A.NotSupportedExceptionAndReturn<br/>bool>();
public int IndexOf(TLink item)
   if (equalityComparer.Equals(Index, item))
      return constants.IndexPart;
```

```
( equalityComparer.Equals(Source, item))
                                                                                                   using System.Ling;
150
151
                return constants.SourcePart;
152
153
                 equalityComparer.Equals(Target, item))
154
155
                return constants. TargetPart;
156
             return -1:
158
159
160
           public void Insert(int index, TLink item) => throw new NotSupportedException();
161
                                                                                               10
                                                                                               1.1
           public void RemoveAt(int index) => throw new NotSupportedException();
163
                                                                                               12
164
                                                                                               13
           #endregion
165
                                                                                               14
166
                                                                                               15
167
                                                                                               16
                                                                                               17
./LinkExtensions.cs
                                                                                               19
     namespace Platform.Data.Doublets
                                                                                               20
                                                                                               21
        public static class LinkExtensions
                                                                                               22
                                                                                               23
           public static bool IsFullPoint<TLink>(this Link<TLink> link) =>
           → Point < TLink > .IsFullPoint(link):
                                                                                               25
           public static bool IsPartialPoint<TLink>(this Link<TLink> link) =>
                                                                                               26
              Point < TLink > .IsPartialPoint(link):
                                                                                               27
                                                                                               28
                                                                                               29
./LinksOperatorBase.cs
                                                                                               31
                                                                                               32
     namespace Platform.Data.Doublets
                                                                                               33
                                                                                               34
        public abstract class LinksOperatorBase<TLink>
           protected readonly ILinks<TLink> Links;
                                                                                               35
           protected LinksOperatorBase(ILinks<TLink> links) => Links = links;
                                                                                               36
                                                                                               37
                                                                                               38
./obj/Debug/netstandard2.0/Platform.Data.Doublets.AssemblyInfo.cs
       /_____
        <auto-generated>
          Generated by the MSBuild WriteCodeFragment class.
        </auto-generated>
     using System:
     using System Reflection;
     [assembly: System.Reflection.AssemblyConfigurationAttribute("Debug")]
     assembly: System.Reflection.AssemblyCopyrightAttribute("Konstantin Diachenko")
11
                                                                                               10
     [assembly: System.Reflection.AssemblyDescriptionAttribute("LinksPlatform\'s
                                                                                               11
     → Platform.Data.Doublets Class Library")
                                                                                               12
     [assembly: System.Reflection.AssemblyFileVersionAttribute("0.0.1.0")
                                                                                               13
     [assembly: System.Reflection.AssemblyInformationalVersionAttribute("0.0.1")]
14
     [assembly: System.Reflection.AssemblyTitleAttribute("Platform.Data.Doublets")]
                                                                                               14
     [assembly: System.Reflection.AssemblyVersionAttribute("0.0.1.0")]
                                                                                               15
                                                                                               16
```

```
./PropertyOperators/DefaultLinkPropertyOperator.cs
    using System.Collections.Generic:
    using Platform.Interfaces:
    namespace Platform.Data.Doublets.PropertyOperators
       public class DefaultLinkPropertyOperator<TLink>: LinksOperatorBase<TLink>,
       → IPropertyOperator<TLink, TLink, TLink>
         private static readonly EqualityComparer<TLink> equalityComparer =
          → EqualityComparer<TLink>.Default;
          public DefaultLinkPropertyOperator(ILinks<TLink> links): base(links)
          public TLink GetValue(TLink @object, TLink property)
             var objectProperty = Links.SearchOrDefault(@object, property);
             if (equality Comparer. Equals (object Property, default))
               return default:
            var valueLink = Links.All(Links.Constants.Any, objectProperty).SingleOrDefault();
             if (valueLink == null)
               return default:
             var value = Links.GetTarget(valueLink[Links.Constants.IndexPart]);
             return value:
          public void SetValue(TLink @object, TLink property, TLink value)
             var objectProperty = Links.GetOrCreate(@object, property);
            Links. Delete Many (Links. All (Links. Constants. Any, object Property). Select (link =>
             → link[Links.Constants.IndexPart]).ToList()):
             Links.GetOrCreate(objectProperty, value);
./PropertyOperators/FrequencyPropertyOperator.cs
    using System Collections Generic:
    using Platform.Interfaces;
    namespace Platform.Data.Doublets.PropertyOperators
       public class FrequencyPropertyOperator<TLink>: LinksOperatorBase<TLink>,
           ISpecificPropertyOperator<TLink, TLink>
         private static readonly EqualityComparer<TLink> equalityComparer =
          → EqualityComparer<TLink>.Default;
         private readonly TLink _frequencyPropertyMarker;
         private readonly TLink frequency Marker;
         public FrequencyPropertyOperator(ILinks<TLink> links, TLink
              frequencyPropertyMarker, TLink frequencyMarker): base(links)
              frequency Property Marker = frequency Property Marker;
             frequencyMarker = frequencyMarker;
```

```
using Platform.Data.Constants;
                                                                                                       using static Platform. Numbers. ArithmeticHelpers:
18
          public TLink Get(TLink link)
19
                                                                                                       #pragma warning disable 0649
20
                                                                                                       #pragma warning disable 169
             var property = Links.SearchOrDefault(link, frequencyPropertyMarker);
                                                                                                  16
21
                                                                                                       #pragma warning disable 618
             var container = GetContainer(property);
                                                                                                  17
22
             var frequency = GetFrequency(container);
                                                                                                       // ReSharper disable StaticMemberInGenericType
             return frequency;
24
                                                                                                         ReSharper disable BuiltInTypeReferenceStyle
                                                                                                  20
25
                                                                                                         ReSharper disable MemberCanBePrivate.Local
                                                                                                  21
26
          private TLink GetContainer(TLink property)
                                                                                                        ReSharper disable UnusedMember.Local
                                                                                                  22
27
                                                                                                  23
28
                                                                                                       namespace Platform.Data.Doublets.ResizableDirectMemory
                                                                                                  24
             var frequencyContainer = default(TLink);
29
                                                                                                  25
             if (equalityComparer.Equals(property, default))
                                                                                                          public partial class ResizableDirectMemoryLinks<TLink>: DisposableBase,
                                                                                                  26
3.1
                                                                                                          → ILinks<TLink>
                return frequencyContainer;
32
                                                                                                  27
33
                                                                                                            private static readonly EqualityComparer<TLink> equalityComparer =
                                                                                                  28
             Links.Each(candidate =>
34
                                                                                                             → EqualityComparer<TLink>.Default;
35
                                                                                                            private static readonly Comparer<TLink> comparer = Comparer<TLink>.Default;
                                                                                                  29
                var candidateTarget = Links.GetTarget(candidate);
                                                                                                  30
                var frequency Target = Links. Get Target (candidate Target);
37
                                                                                                             /// <summary>Возвращает размер одной связи в байтах.</summary>
                                                                                                  31
                if (equalityComparer.Equals(frequencyTarget, frequencyMarker))
                                                                                                            public static readonly int LinkSizeInBytes = StructureHelpers.SizeOf<Link>();
                                                                                                  32
30
                                                                                                  33
                   frequencyContainer = Links.GetIndex(candidate);
                                                                                                            public static readonly int LinkHeaderSizeInBytes =
                                                                                                  34
                   return Links.Constants.Break:
41
                                                                                                             → StructureHelpers.SizeOf<LinksHeader>();
                                                                                                  35
                return Links.Constants.Continue:
                                                                                                            public static readonly long DefaultLinksSizeStep = LinkSizeInBytes * 1024 * 1024;
                                                                                                  36
              }, Links.Constants.Any, property, Links.Constants.Any);
                                                                                                  37
             return frequencyContainer;
                                                                                                            private struct Link
                                                                                                  38
                                                                                                  39
47
                                                                                                                public static readonly int SourceOffset = Marshal.OffsetOf(typeof(Link),
                                                                                                  40
          private TLink GetFrequency(TLink container) =>
                                                                                                                → nameof(Source)).ToInt32():
                 equalityComparer.Equals(container, default)? default:
                                                                                                                public static readonly int TargetOffset = Marshal.OffsetOf(typeof(Link),
                                                                                                  41
              Links.Get Target (container);
                                                                                                                \rightarrow nameof(Target)).ToInt32():
                                                                                                                public static readonly int Left AsSourceOffset = Marshal.OffsetOf(typeof(Link),
                                                                                                  42
          public void Set(TLink link, TLink frequency)
50
                                                                                                                \rightarrow nameof(LeftAsSource)).ToInt32():
51
                                                                                                                public static readonly int Right AsSourceOffset = Marshal.OffsetOf(typeof(Link),
             var property = Links.GetOrCreate(link, frequencyPropertyMarker);
52
                                                                                                                \rightarrow nameof(RightAsSource)).ToInt32():
             var container = GetContainer(property);
53
                                                                                                                public static readonly int SizeAsSourceOffset = Marshal.OffsetOf(typeof(Link),
             if (equalityComparer.Equals(container, default))
                                                                                                  44
54
                                                                                                                \rightarrow nameof(SizeAsSource)).ToInt32():
55
                                                                                                                public static readonly int LeftAsTargetOffset = Marshal.OffsetOf(typeof(Link),
                Links.GetOrCreate(property, frequency);
                                                                                                                \rightarrow nameof(LeftAsTarget)).ToInt32():
5.7
             else
                                                                                                                public static readonly int RightAsTargetOffset = Marshal.OffsetOf(typeof(Link),
                                                                                                                \rightarrow nameof(RightAsTarget)).ToInt32():
                Links. Update(container, property, frequency);
                                                                                                                public static readonly int SizeAsTargetOffset = Marshal.OffsetOf(typeof(Link))
                                                                                                  47
61
                                                                                                                \rightarrow nameof(SizeAsTarget)).ToInt32();
                                                                                                  48
                                                                                                                public TLink Source:
63
                                                                                                  49
                                                                                                                public TLink Target:
                                                                                                  50
                                                                                                                public TLink Left AsSource:
                                                                                                  51
                                                                                                                public TLink RightAsSource;
                                                                                                  52
./ResizableDirectMemory/ResizableDirectMemoryLinks.cs
                                                                                                                public TLink SizeAsSource;
                                                                                                  53
    using System:
                                                                                                                public TLink LeftAsTarget;
                                                                                                  54
    using System.Collections.Generic:
                                                                                                                public TLink Right AsTarget;
                                                                                                  55
    using System.Runtime.CompilerServices;
                                                                                                                public TLink SizeAsTarget:
                                                                                                  56
    using System.Runtime.InteropServices:
                                                                                                  57
    using Platform. Disposables:
                                                                                                                [MethodImpl(MethodImplOptions.AggressiveInlining)]
                                                                                                  58
    using Platform. Helpers. Singletons;
                                                                                                                public static TLink GetSource(IntPtr pointer) => (pointer +
                                                                                                  59
    using Platform.Collections.Arrays:
                                                                                                                → SourceOffset).GetValue<TLink>():
    using Platform. Numbers:
                                                                                                                [MethodImpl(MethodImplOptions.AggressiveInlining)]
    using Platform. Unsafe;
                                                                                                                public static TLink GetTarget(IntPtr pointer) => (pointer +
    using Platform. Memory;
                                                                                                                → TargetOffset).GetValue<TLink>();
    using Platform. Data. Exceptions;
```

62	[MethodImpl(MethodImplOptions.AggressiveInlining)]	101	public static readonly int LastFreeLinkOffset =
63	$ ext{public static TLink GetLeftAsSource(IntPtr pointer)} => ext{(pointer } +$		→ Marshal.OffsetOf(typeof(LinksHeader), nameof(LastFreeLink)).ToInt32();
	\rightarrow Left AsSourceOffset).GetValue <tlink>();</tlink>	102	All many and a second s
64	[MethodImpl(MethodImplOptions.AggressiveInlining)]	103	public TLink AllocatedLinks;
65	$ m public \ static \ TLink \ GetRightAsSource(IntPtr \ pointer) => (pointer +$	104	public TLink ReservedLinks;
	→ Right AsSourceOffset).GetValue <tlink>();</tlink>	105	public TLink FreeLinks;
66	[MethodImpl(MethodImplOptions.AggressiveInlining)]	106	public TLink FirstFreeLink;
67	public static TLink GetSizeAsSource(IntPtr pointer) => (pointer +	107	public TLink FirstAsSource; public TLink FirstAsTarget;
	\rightarrow SizeAsSourceOffset).GetValue $<$ TLink $>$ ();	108 109	public TLink LastFreeLink;
68	[MethodImpl(MethodImplOptions.AggressiveInlining)]	110	public TLink Reserved8;
69	public static TLink GetLeftAsTarget(IntPtr pointer) => (pointer +	111	public 12lin reconvedo,
	→ LeftAsTargetOffset).GetValue <tlink>();</tlink>	112	[MethodImpl(MethodImplOptions.AggressiveInlining)]
70	[MethodImpl(MethodImplOptions.AggressiveInlining)]	113	public static TLink GetAllocatedLinks(IntPtr pointer) => (pointer +
71	public static TLink GetRightAsTarget(IntPtr pointer) => (pointer +		→ AllocatedLinksOffset).GetValue <tlink>();</tlink>
	→ Right AsTarget Offset). Get Value < TLink > ();	114	[MethodImpl(MethodImplOptions.AggressiveInlining)]
72	[MethodImpl(MethodImplOptions.AggressiveInlining)]	115	public static TLink GetReservedLinks(IntPtr pointer) => (pointer +
73	public static TLink GetSizeAsTarget(IntPtr pointer) => (pointer +		→ ReservedLinksOffset).GetValue <tlink>();</tlink>
	\rightarrow SizeAsTargetOffset).GetValue <tlink>();</tlink>	116	[MethodImpl(MethodImplOptions.AggressiveInlining)]
74	→ bizeristargeromser). Get variae < 1 Emix (),	117	public static TLink GetFreeLinks(IntPtr pointer) => (pointer +
75	[MethodImpl(MethodImplOptions.AggressiveInlining)]		\rightarrow FreeLinksOffset).GetValue <tlink>();</tlink>
76	public static void SetSource(IntPtr pointer, TLink value) => (pointer +	118	[MethodImpl(MethodImplOptions.AggressiveInlining)]
	→ SourceOffset).SetValue(value);	119	public static TLink GetFirstFreeLink(IntPtr pointer) => (pointer +
77	[MethodImpl(MethodImplOptions.AggressiveInlining)]	110	→ FirstFreeLinkOffset).GetValue <tlink>();</tlink>
78	public static void SetTarget(IntPtr pointer, TLink value) => (pointer +	120	[MethodImpl(MethodImplOptions.AggressiveInlining)]
	→ TargetOffset).SetValue(value);	121	public static TLink GetFirstAsSource(IntPtr pointer) => (pointer +
79	[MethodImpl(MethodImplOptions.AggressiveInlining)]		\rightarrow FirstAsSourceOffset).GetValue <tlink>();</tlink>
80	public static void SetLeft AsSource(IntPtr pointer, TLink value) => (pointer +	122	[MethodImpl(MethodImplOptions.AggressiveInlining)]
	→ Left AsSourceOffset) .SetValue(value);	123	public static TLink GetFirstAsTarget(IntPtr pointer) => (pointer +
81	[MethodImpl(MethodImplOptions.AggressiveInlining)]		\rightarrow FirstAsTargetOffset).GetValue <tlink>();</tlink>
82	public static void SetRightAsSource(IntPtr pointer, TLink value) => (pointer +	124	[MethodImpl(MethodImplOptions.AggressiveInlining)]
	→ Right AsSourceOffset).Set Value(value);	125	public static TLink GetLastFreeLink(IntPtr pointer) => (pointer +
83	[MethodImpl(MethodImplOptions.AggressiveInlining)]		\hookrightarrow LastFreeLinkOffset).GetValue <tlink>();</tlink>
84	public static void SetSizeAsSource(IntPtr pointer, TLink value) => (pointer +	126	, 2350110021111011500, 0000 01111115 (),
	→ SizeAsSourceOffset).SetValue(value);	127	[MethodImpl(MethodImplOptions.AggressiveInlining)]
85	[MethodImpl(MethodImplOptions.AggressiveInlining)]	128	public static IntPtr GetFirstAsSourcePointer(IntPtr pointer) => pointer +
86	public static void SetLeftAsTarget(IntPtr pointer, TLink value) => (pointer +		\hookrightarrow FirstAsSourceOffset;
	→ Left As Target Offset). Set Value (value);	129	[MethodImpl(MethodImplOptions.AggressiveInlining)]
87	[MethodImpl(MethodImplOptions.AggressiveInlining)]	130	$public \ static \ IntPtr \ GetFirstAsTargetPointer(IntPtr \ pointer) => pointer \ +$
88	${\rm public\ static\ void\ SetRightAsTarget(IntPtr\ pointer,\ TLink\ value)} => ({\rm pointer}\ +$		\hookrightarrow FirstAsTargetOffset;
	→ Right AsTarget Offset). Set Value(value);	131	
89	[Method Impl(Method Impl Options. Aggressive Inlining)]	132	[MethodImpl(MethodImplOptions.AggressiveInlining)]
90	$public\ static\ void\ SetSizeAsTarget(IntPtr\ pointer,\ TLink\ value) => (pointer\ +$	133	public static void SetAllocatedLinks(IntPtr pointer, TLink value) => (pointer +
	→ SizeAsTargetOffset).SetValue(value);		→ AllocatedLinksOffset).SetValue(value); [Mathed Hand (Mathed Hand) (Options Assume size Individual)]
91	}	134	[MethodImpl(MethodImplOptions.AggressiveInlining)]
92		135	public static void SetReservedLinks(IntPtr pointer, TLink value) => (pointer +
93	private struct LinksHeader		→ ReservedLinksOffset).SetValue(value);
94	public static readonly int AllocatedLinksOffset =	136	[MethodImpl(MethodImplOptions.AggressiveInlining)] public static void SetFreeLinks(IntPtr pointer, TLink value) => (pointer +
95	\rightarrow Marshal.OffsetOf(typeof(LinksHeader), nameof(AllocatedLinks)).ToInt32();	137	
0.6	\rightarrow Marshar. Offset Offcy peof (Efficience 1), name of (Allocated Efficience). For the S2(), public static readonly int Reserved Links Offset =		→ FreeLinksOffset).SetValue(value); [MethodImpl(MethodImplOptions.AggressiveInlining)]
96	7. 1. 1. 0.00 ± 0.00 (138	public static void SetFirstFreeLink(IntPtr pointer, TLink value) => (pointer +
0.7	→ Marshal.OffsetOf(typeof(LinksHeader), nameof(ReservedLinks)).ToInt32(); public static readonly int FreeLinksOffset = Marshal.OffsetOf(typeof(LinksHeader),	139	
97			→ FirstFreeLinkOffset).SetValue(value); [MethodImpl(MethodImplOptions.AggressiveInlining)]
0.8	→ nameof(FreeLinks)).ToInt32(); public static readonly int FirstFreeLinkOffset =	140	public static void SetFirstAsSource(IntPtr pointer, TLink value) => (pointer +
98	\rightarrow Marshal.OffsetOf(typeof(LinksHeader), nameof(FirstFreeLink)).ToInt32();	141	public static void SetFils(AsSource(inter it pointer, 1 Link varie) => (pointer + → FirstAsSourceOffset).SetValue(value);
99	public static readonly int FirstAsSourceOffset =	1.49	[MethodImpl(MethodImplOptions.AggressiveInlining)]
20	\rightarrow Marshal.OffsetOf(typeof(LinksHeader), nameof(FirstAsSource)).ToInt32();	142 143	public static void SetFirstAsTarget(IntPtr pointer, TLink value) => (pointer +
100	public static readonly int FirstAsTargetOffset =	1.20	→ FirstAsTargetOffset).SetValue(value);
	→ Marshal.OffsetOf(typeof(LinksHeader), nameof(FirstAsTarget)).ToInt32();	144	[MethodImpl(MethodImplOptions.AggressiveInlining)]
	([

```
public static void SetLastFreeLink(IntPtr pointer, TLink value) => (pointer +
                                                                                                198
145
                 LastFreeLinkOffset).SetValue(value):
146
147
                                                                                                199
           private readonly long memoryReservationStep;
148
149
                                                                                                200
150
           private readonly IResizableDirectMemory memory:
           private IntPtr header;
151
           private IntPtr —links;
152
153
                                                                                                201
           private LinksTargetsTreeMethods targetsTreeMethods;
154
                                                                                                202
           private LinksSourcesTreeMethods sourcesTreeMethods;
155
                                                                                                203
156
                                                                                                204
           // TODO: Возможно чтобы гарантированно проверять на то, является ли связь
157
                                                                                                205
               удалённой, нужно использовать не список а дерево, так как так можно
                                                                                                206
               быстрее проверить на наличие связи внутри
           private UnusedLinksListMethods unusedLinksListMethods;
158
                                                                                                207
159
                                                                                                208
               <summary>
160
                                                                                                209
               Возвращает общее число связей находящихся в хранилище.
161
                                                                                                210
              </summary>
162
                                                                                                211
           private TLink Total => Subtract(LinksHeader.GetAllocatedLinks(header),
163
                                                                                                212
               LinksHeader.GetFreeLinks( header));
                                                                                                213
164
                                                                                                214
           public LinksCombinedConstants<TLink, TLink, int> Constants { get; }
165
                                                                                                215
166
                                                                                                216
           public ResizableDirectMemoryLinks(string address)
167
                                                                                                217
              : this(address, DefaultLinksSizeStep)
168
                                                                                                218
169
                                                                                                210
170
                                                                                                220
171
                                                                                                221
172
                                                                                                222
              Создаёт экземпляр базы данных Links в файле по указанному адресу, с
173
                                                                                                223
               указанным минимальным шагом расширения базы данных.
                                                                                                224
174
                                                                                                225
               <param name="address">Полный пусть к файлу базы данных.</param>
175
                                                                                                226
           /// <param name="memoryReservationStep">Минимальный шаг расширения базы
176
               данных в байтах. </param>
                                                                                                228
           public ResizableDirectMemoryLinks(string address, long memoryReservationStep)
177
                                                                                                229
              : this(new FileMappedResizableDirectMemory(address, memoryReservationStep).
178
                                                                                                230

→ memoryReservationStep)

179
                                                                                                231
180
                                                                                                232
181
                                                                                                233
           public ResizableDirectMemoryLinks(IResizableDirectMemory memory)
182
                                                                                                234
              : this(memory, DefaultLinksSizeStep)
183
                                                                                                235
184
                                                                                                236
185
                                                                                                237
186
                                                                                                ^{238}
           public ResizableDirectMemoryLinks(IResizableDirectMemory memory, long
187
                                                                                                239
               memoryReservationStep)
                                                                                                240
188
              Constants = Default < LinksCombinedConstants < TLink, TLink, int >> . Instance:
189
                                                                                                242
               memory = memory;
190
                                                                                                243
               memoryReservationStep = memoryReservationStep;
191
                                                                                                244
              if (memory.ReservedCapacity < memoryReservationStep)
192
                                                                                                245
                                                                                                246
                memory.ReservedCapacity = memoryReservationStep;
194
                                                                                                247
195
                                                                                                248
              SetPointers( memory);
196
                                                                                                ^{249}
              // Гарантия корректности memory. UsedCapacity относительно
197
                                                                                                250
                 header->AllocatedLinks
                                                                                                251
                                                                                                252
```

```
memory.UsedCapacity =
      ((long)(Integer<TLink>)LinksHeader.GetAllocatedLinks( header) *
      LinkSizeInBytes) + LinkHeaderSizeInBytes:
  // Гарантия корректности header->ReservedLinks относительно
        memory.ReservedCapacity
  LinksHeader.SetReservedLinks(header,
       (Integer < TLink > )(( memory.Reserved Capacity - Link Header Size In Bytes)
      LinkSizeInBvtes)):
[MethodImpl(MethodImplOptions.AggressiveInlining)]
public TLink Count(IList<TLink> restrictions)
  // Если нет ограничений, тогла возвращаем общее число связей нахолящихся в
   → хранилище.
  if (restrictions.Count == 0)
     return Total:
    (restrictions.Count == 1)
     var index = restrictions[Constants.IndexPart];
     if (equalityComparer.Equals(index, Constants.Any))
        return Total:
     return Exists(index)? Integer<TLink>.One: Integer<TLink>.Zero:
    (restrictions.Count == 2)
     var index = restrictions[Constants.IndexPart]
     var value = restrictions[1];
     if (equalityComparer.Equals(index, Constants.Any))
        if ( equalityComparer,Equals(value, Constants,Any))
           return Total; // Any - как отсутствие ограничения
        return Add( sourcesTreeMethods.CalculateReferences(value)
            targetsTreeMethods.CalculateReferences(value));
     else
        if (!Exists(index))
           return Integer<TLink>.Zero;
           equalityComparer.Equals(value, Constants.Any))
           return Integer<TLink>.One;
        var storedLinkValue = GetLinkUnsafe(index):
        if (equalityComparer.Equals(Link.GetSource(storedLinkValue), value)
            equalityComparer.Equals(Link.GetTarget(storedLinkValue), value))
           return Integer<TLink>.One;
        return Integer<TLink>.Zero;
  if (restrictions.Count == 3)
```

```
var index = restrictions[Constants.IndexPart];
                                                                                                        return Integer<TLink>.One;
                                                                                 310
var source = restrictions[Constants.SourcePart];
                                                                                 311
                                                                                                     return Integer<TLink>.Zero:
var target = restrictions[Constants.TargetPart];
                                                                                 312
                                                                                 313
if (equalityComparer.Equals(index, Constants.Any))
                                                                                 314
                                                                                               throw new NotSupportedException("Другие размеры и способы ограничений не
                                                                                 315
   if (equalityComparer.Equals(source, Constants.Any) &&
                                                                                               → поддерживаются.");
       equalityComparer.Equals(target, Constants.Any))
                                                                                 316
                                                                                 317
      return Total:
                                                                                            [MethodImpl(MethodImplOptions.AggressiveInlining)]
                                                                                 318
                                                                                            public TLink Each (Func < IList < TLink >, TLink > handler, IList < TLink > restrictions)
                                                                                 319
   else if ( equalityComparer.Equals(source, Constants.Any))
                                                                                 320
                                                                                               if (restrictions.Count == 0)
                                                                                 321
      return targetsTreeMethods.CalculateReferences(target);
                                                                                 322
                                                                                                  for (TLink link = Integer < TLink > . One; comparer. Compare(link,
                                                                                 323
   else if ( equalityComparer.Equals(target, Constants.Any))
                                                                                                      (Integer < TLink > )LinksHeader.GetAllocatedLinks(header)) <= 0; link =
                                                                                                      Increment(link))
      return sourcesTreeMethods.CalculateReferences(source):
                                                                                 324
                                                                                                     if (Exists(link) && equalityComparer.Equals(handler(GetLinkStruct(link)),
                                                                                 325
   else //if(source != Any && target != Any)
                                                                                                         Constants.Break))
                                                                                 326
      // Эквивалент Exists(source, target) => Count(Any, source, target) > 0
                                                                                                        return Constants.Break;
                                                                                 327
      var link = sourcesTreeMethods.Search(source, target);
                                                                                 328
      return equalityComparer.Equals(link, Constants.Null)?
                                                                                 329
          Integer<TLink>.Zero: Integer<TLink>.One;
                                                                                 330
                                                                                                  return Constants.Continue;
                                                                                 331
                                                                                 332
else
                                                                                                 (restrictions.Count == 1)
                                                                                 333
                                                                                 334
   if (!Exists(index))
                                                                                                  var index = restrictions[Constants.IndexPart]:
                                                                                 335
                                                                                                  if (equality Comparer. Equals (index, Constants. Any))
                                                                                 336
      return Integer<TLink>.Zero;
                                                                                 337
                                                                                                     return Each(handler, ArrayPool<TLink>.Empty);
                                                                                 338
   if (equalityComparer.Equals(source, Constants.Any) &&
                                                                                 339
         equalityComparer.Equals(target, Constants.Any))
                                                                                                    (!Exists(index))
                                                                                 340
                                                                                 341
      return Integer<TLink>.One;
                                                                                                     return Constants.Continue;
                                                                                 342
                                                                                 343
   var storedLinkValue = GetLinkUnsafe(index);
                                                                                                  return handler(GetLinkStruct(index));
                                                                                 344
   if (! equalityComparer.Equals(source, Constants.Any) &&
                                                                                 345
                                                                                               if (restrictions.Count == 2)
      ! equalityComparer.Equals(target, Constants.Any))
                                                                                 346
                                                                                 347
                                                                                                  var index = restrictions[Constants.IndexPart];
      if (equalityComparer.Equals(Link.GetSource(storedLinkValue), source)
                                                                                348
                                                                                                  var value = restrictions[1]:
                                                                                 349
                                                                                                  if (equalityComparer.Equals(index, Constants.Any))
          equalityComparer.Equals(Link.GetTarget(storedLinkValue), target))
                                                                                350
                                                                                 351
                                                                                                     if (equalityComparer.Equals(value, Constants.Any))
         return Integer<TLink>.One;
                                                                                 352
                                                                                 353
                                                                                                        return Each(handler, ArrayPool<TLink>.Empty);
      return Integer<TLink>.Zero;
                                                                                 354
                                                                                 355
   var value = default(TLink);
                                                                                                         equalityComparer.Equals(Each(handler, new]) { index, value,
                                                                                 356
   if (equalityComparer.Equals(source, Constants.Any))
                                                                                                         Constants.Any \,\), Constants.Break))
                                                                                 357
      value = target;
                                                                                                        return Constants.Break:
                                                                                 358
                                                                                 359
       equalityComparer.Equals(target, Constants.Any))
                                                                                                     return Each(handler, new[] { index, Constants.Any, value });
                                                                                 360
                                                                                 361
      value = source:
                                                                                 362
                                                                                 363
       equalityComparer.Equals(Link.GetSource(storedLinkValue), value)
                                                                                                     if (!Exists(index))
                                                                                 364
       equalityComparer.Equals(Link.GetTarget(storedLinkValue), value))
                                                                                 365
```

```
return Constants.Continue;
                                                                                                           return handler(GetLinkStruct(index)):
                                                                                 422
                                                                                 423
     ( equalityComparer.Equals(value, Constants.Anv))
                                                                                 424
                                                                                                        return Constants.Continue:
                                                                                 425
      return handler(GetLinkStruct(index));
                                                                                                     var value = default(TLink):
                                                                                 426
                                                                                                     if (equalityComparer.Equals(source, Constants.Any))
                                                                                 427
    var storedLinkValue = GetLinkUnsafe(index);
                                                                                 428
                                                                                                        value = target:
       equalityComparer.Equals(Link.GetSource(storedLinkValue), value)
                                                                                 429
        equalityComparer.Equals(Link.GetTarget(storedLinkValue), value))
                                                                                 430
                                                                                                         equalityComparer.Equals(target, Constants.Anv))
                                                                                 431
      return handler(GetLinkStruct(index));
                                                                                 432
                                                                                                        value = source:
                                                                                 433
    return Constants.Continue;
                                                                                 434
                                                                                                         equalityComparer.Equals(Link.GetSource(storedLinkValue), value)
                                                                                 435
                                                                                                          equalityComparer.Equals(Link.GetTarget(storedLinkValue), value))
                                                                                 436
(restrictions.Count == 3)
                                                                                 437
                                                                                                        return handler(GetLinkStruct(index)):
                                                                                 438
var index = restrictions[Constants.IndexPart];
                                                                                 439
                                                                                                     return Constants.Continue;
var source = restrictions[Constants.SourcePart];
                                                                                 440
var target = restrictions[Constants.TargetPart];
                                                                                 441
if (equalityComparer.Equals(index, Constants.Any))
                                                                                 442
                                                                                                throw new NotSupportedException("Другие размеры и способы ограничений не
                                                                                 443
    if (equalityComparer.Equals(source, Constants.Any) &&
                                                                                                equalityComparer.Equals(target, Constants.Anv))
                                                                                 444
                                                                                 445
                                                                                                <remarks>
                                                                                 446
      return Each(handler, ArrayPool<TLink>,Empty);
                                                                                                ТООО: Возможно можно перемещать значения, если указан индекс, но
                                                                                 447
                                                                                                 значение существует в другом месте (но не в менеджере памяти, а в логике
    else if ( equalityComparer.Equals(source, Constants.Any))
                                                                                                 Links)
      return targetsTreeMethods.EachReference(target, handler);
                                                                                             ^{\prime}//</\mathrm{remarks}>
                                                                                 448
                                                                                             [MethodImpl(MethodImplOptions.AggressiveInlining)]
                                                                                 449
                                                                                             public TLink Update(IList<TLink> values)
    else if ( equalityComparer.Equals(target, Constants.Any))
                                                                                 450
                                                                                 451
                                                                                                var linkIndex = values[Constants.IndexPart];
      return sourcesTreeMethods.EachReference(source, handler);
                                                                                 452
                                                                                               var link = GetLinkUnsafe(linkIndex);
                                                                                 453
    else //if(source != Any && target != Any)
                                                                                                // Будет корректно работать только в том случае, если пространство
                                                                                 454
                                                                                                → выделенной связи предварительно заполнено нулями
                                                                                               if (! equalityComparer.Equals(Link.GetSource(link), Constants.Null))
      var link = sourcesTreeMethods.Search(source, target);
                                                                                 455
      return equalityComparer.Equals(link, Constants.Null)?
                                                                                 456
                                                                                                   sources Tree Methods. Detach (Links Header. Get First As Source Pointer (\ header)
          Constants.Continue: handler(GetLinkStruct(link)):
                                                                                 457
                                                                                                   \rightarrow linkIndex):
                                                                                 458
                                                                                                if (! equalityComparer.Equals(Link.GetTarget(link), Constants.Null))
else
                                                                                 459
                                                                                 460
                                                                                                   targetsTreeMethods.Detach(LinksHeader.GetFirstAsTargetPointer(header)
    if (!Exists(index))
                                                                                 461
                                                                                                   \rightarrow linkIndex):
      return Constants.Continue;
                                                                                 462
                                                                                                Link.SetSource(link, values[Constants.SourcePart]);
                                                                                 463
    if (equalityComparer.Equals(source, Constants.Any) &&
                                                                                                Link.SetTarget(link, values[Constants.TargetPart]);
                                                                                 464
        equalityComparer.Equals(target, Constants.Any))
                                                                                                if (! equalityComparer.Equals(Link.GetSource(link), Constants.Null))
                                                                                 465
                                                                                 466
      return handler(GetLinkStruct(index));
                                                                                                   sources Tree Methods. Attach (Links Header. Get First As Source Pointer (\ header)
                                                                                 467
                                                                                                   \rightarrow linkIndex):
    var storedLinkValue = GetLinkUnsafe(index);
                                                                                 468
    if (! equalityComparer.Equals(source, Constants.Any) &&
                                                                                                if (! equalityComparer.Equals(Link.GetTarget(link), Constants.Null))
                                                                                 469
        ! equalityComparer.Equals(target, Constants.Any))
                                                                                 470
                                                                                                   targetsTreeMethods.Attach(LinksHeader.GetFirstAsTargetPointer( header)
                                                                                 471
      if (equalityComparer.Equals(Link.GetSource(storedLinkValue), source)
                                                                                                   \rightarrow linkIndex):
                                                                                 472
           equalityComparer.Equals(Link.GetTarget(storedLinkValue), target))
                                                                                                return linkIndex;
                                                                                 473
                                                                                 474
```

368

370

371

372

373

375

376

377

378

370

380

381

382

383

384

385

386

387

388

389

390

392

393

394

395

396

397

399

401

403

404

405

400

410

411

412

413

415

416

418

419

```
527
[MethodImpl(MethodImplOptions.AggressiveInlining)]
public Link<TLink> GetLinkStruct(TLink linkIndex)
  var link = GetLinkUnsafe(linkIndex):
                                                                                    520
  return new Link<TLink>(linkIndex, Link.GetSource(link), Link.GetTarget(link));
                                                                                    531
[MethodImpl(MethodImplOptions.AggressiveInlining)]
                                                                                    532
private IntPtr GetLinkUnsafe(TLink linkIndex) =>
                                                                                    533
    links.GetElement(LinkSizeInBvtes, linkIndex):
                                                                                    534
                                                                                    535
   <remarks>
                                                                                    536
   TODO: Возможно нужно будет заполнение нулями, если внешнее API ими не
                                                                                    537

⇒ заполняет пространство

 // </ \text{remarks}>
                                                                                    538
public TLink Create()
                                                                                    539
                                                                                    540
  var freeLink = LinksHeader.GetFirstFreeLink( header);
                                                                                    541
  if (! equalityComparer.Equals(freeLink, Constants.Null))
                                                                                    542
                                                                                    543
      unusedLinksListMethods.Detach(freeLink);
                                                                                    544
                                                                                    545
  else
                                                                                    546
                                                                                    547
         comparer.Compare(LinksHeader.GetAllocatedLinks(header),
                                                                                    548
         Constants.MaxPossibleIndex) > 0
                                                                                    549
                                                                                    550
        throw new LinksLimitReachedException((Integer<TLink>)Constants.MaxP
                                                                                    551
                                                                                    552
        \rightarrow ossibleIndex):
                                                                                    553
                                                                                    554
         comparer.Compare(LinksHeader.GetAllocatedLinks(header),
                                                                                    555
         Decrement(LinksHeader.GetReservedLinks(header))) >= 0)
                                                                                    556
                                                                                    557
          memory.ReservedCapacity += memoryReservationStep;
                                                                                    558
        SetPointers( memory):
                                                                                    559
        LinksHeader.SetReservedLinks(header,
                                                                                    560
            (Integer<TLink>)( memory.ReservedCapacity / LinkSizeInBytes)):
                                                                                    561
     LinksHeader.Set AllocatedLinks(header,
                                                                                    563
     → Increment(LinksHeader.GetAllocatedLinks( header)));
                                                                                    564
      memory.UsedCapacity += LinkSizeInBytes;
                                                                                    565
     freeLink = LinksHeader.GetAllocatedLinks( header);
                                                                                    566
                                                                                    567
  return freeLink;
                                                                                    568
                                                                                    569
                                                                                    570
public void Delete(TLink link)
                                                                                    571
                                                                                    572
  if (comparer.Compare(link, LinksHeader.GetAllocatedLinks(header)) < 0)
                                                                                    573
      unusedLinksListMethods.AttachAsFirst(link);
                                                                                    574
  else if ( equalityComparer.Equals(link, LinksHeader.GetAllocatedLinks( header)))
                                                                                    575
                                                                                    576
     LinksHeader.SetAllocatedLinks( header,
                                                                                    577
     → Decrement(LinksHeader.GetAllocatedLinks( header)));
                                                                                    578
      memory.UsedCapacity -= LinkSizeInBytes;
                                                                                    579
     Убираем все связи, находящиеся в списке свободных в конце файла, до
                                                                                    580
     → тех пор, пока не дойдём до первой существующей связи
                                                                                    581
    // Позволяет оптимизировать количество выделенных связей (AllocatedLinks)
                                                                                   582
                                                                                    583
```

476

477

478

479

480

481

482

483

484

485

486

487

489

490

491

492

493

494

496

497

498

500

502

503

505

507

508

509

510

511

512

513

514

515

516

517

518

519

521

522

523

524

525

```
while (( comparer.Compare(LinksHeader.GetAllocatedLinks( header),
          Integer \langle TLink \rangle, Zero \rangle > 0 &&
         IsUnusedLink(LinksHeader.GetAllocatedLinks(header)))
         unusedLinksListMethods.Detach(LinksHeader.GetAllocatedLinks( header));
        LinksHeader.Set Allocated Links (header,
         → Decrement(LinksHeader.GetAllocatedLinks(header))):
         memory.UsedCapacity -= LinkSizeInBytes:
    <remarks>
   TODO: Возможно это должно быть событием, вызываемым из IMemory, в том
    случае, если адрес реально поменялся
   Указатель this.links может быть в том же месте.
   так как 0-я связь не используется и имеет такой же размер как Header,
   поэтому header размещается в том же месте, что и 0-я связь
   </remarks>
private void SetPointers(IDirectMemory memory)
   if (memory == null)
       links = IntPtr.Zero;
       header = links
       \overline{\text{unusedLinksListMethods}} = \text{null};
       targetsTreeMethods = null;
       unusedLinksListMethods = null:
       links = memory.Pointer:
       header = links
       sourcesTreeMethods = new LinksSourcesTreeMethods(this);
       targetsTreeMethods = new LinksTargetsTreeMethods(this);
       unusedLinksListMethods = new UnusedLinksListMethods( links, header);
[MethodImpl(MethodImplOptions.AggressiveInlining)]
private bool Exists(TLink link)
   => ( comparer.Compare(link, Constants.MinPossibleIndex) >= 0)
   && (^{-}comparer.Compare(link, LinksHeader.GetAllocatedLinks(header)) \leq = 0)
   &&!IsUnusedLink(link):
[MethodImpl(MethodImplOptions.AggressiveInlining)]
private bool IsUnusedLink(TLink link)
   => equalityComparer.Equals(LinksHeader.GetFirstFreeLink( header), link)
  (Link), equality Comparer. Equals (Link. Get Size AsSource (Get Link Unsafe (link)),
      Constants. Null)
  &&! equalityComparer.Equals(Link.GetSource(GetLinkUnsafe(link))).

→ Constants.Null));

#region DisposableBase
protected override bool AllowMultipleDisposeCalls => true;
protected override void DisposeCore(bool manual, bool wasDisposed)
   if (!wasDisposed)
      SetPointers(null);
```

```
41
             Disposable. Try Dispose (memory);
585
586
587
                                                                                                ./ResizableDirectMemory/ResizableDirectMemoryLinks.TreeMethods.cs
           #endregion
588
                                                                                                    using System:
589
                                                                                                    using System Text:
590
                                                                                                    using System Collections Generic:
                                                                                                    using System.Runtime.CompilerServices:
./ResizableDirectMemory/ResizableDirectMemoryLinks.ListMethods.cs
                                                                                                    using Platform. Numbers;
     using System:
                                                                                                    using Platform.Unsafe:
     using Platform. Unsafe:
                                                                                                    using Platform. Collections. Methods. Trees;
     using Platform. Collections. Methods. Lists:
                                                                                                    using Platform.Data.Constants;
     namespace Platform.Data.Doublets.ResizableDirectMemory
                                                                                                    namespace Platform.Data.Doublets.ResizableDirectMemory
                                                                                                10
                                                                                                11
        partial class ResizableDirectMemoryLinks<TLink>
                                                                                                       partial class ResizableDirectMemoryLinks<TLink>
                                                                                                12
                                                                                                13
           private class UnusedLinksListMethods: CircularDoublyLinkedListMethods<TLink>
                                                                                                          private abstract class LinksTreeMethodsBase:
                                                                                                14
                                                                                                              SizedAndThreadedAVLBalancedTreeMethods<TLink>
              private readonly IntPtr links;
                                                                                                15
              private readonly IntPtr header;
                                                                                                             private readonly ResizableDirectMemoryLinks<TLink> memory;
                                                                                                16
                                                                                                             private readonly LinksCombinedConstants<TLink, TLink, int> constants;
                                                                                                17
              public UnusedLinksListMethods(IntPtr links, IntPtr header)
                                                                                                             protected readonly IntPtr Links:
                                                                                                18
                                                                                                             protected readonly IntPtr Header:
                                                                                                19
                  links = links:
                                                                                                20
                 -header = header;
17
                                                                                                             protected LinksTreeMethodsBase(ResizableDirectMemoryLinks<TLink> memory)
                                                                                                21
                                                                                                22
                                                                                                                Links = memory. links;
                                                                                                23
             protected override TLink GetFirst() => ( header +
                                                                                                                Header = memory header:
                                                                                                24
              → LinksHeader.FirstFreeLinkOffset).GetValue<TLink>();
                                                                                                                memory = memory;
                                                                                                25
                                                                                                                 \overline{\text{constants}} = \overline{\text{memory.Constants}};
2.1
              protected override TLink GetLast() => ( header +
                                                                                                27
                 LinksHeader.LastFreeLinkOffset).GetValue<TLink>();
                                                                                                28
                                                                                                             [MethodImpl(MethodImplOptions.AggressiveInlining)]
                                                                                                29
             protected override TLink GetPrevious(TLink element) =>
                                                                                                             protected abstract TLink GetTreeRoot():
24
                                                                                                3.0
                    links.GetElement(LinkSizeInBytes, element) +
                                                                                                31
                                                                                                             [MethodImpl(MethodImplOptions.AggressiveInlining)]
                                                                                                32
                 Link.SourceOffset).GetValue<TLink>();
                                                                                                             protected abstract TLink GetBasePartValue(TLink link);
                                                                                                33
                                                                                                34
             protected override TLink GetNext(TLink element) =>
                                                                                                             public TLink this [TLink index]
                                                                                                35
                    links.GetElement(LinkSizeInBytes, element) +
                                                                                                36
                 Link.TargetOffset).GetValue<TLink>():
                                                                                                37
                                                                                                38
             protected override TLink GetSize() => ( header +
                                                                                                                   var root = GetTreeRoot():
                                                                                                39
              → LinksHeader.FreeLinksOffset).GetValue<TLink>();
                                                                                                                   if (GreaterOrEqualThan(index, GetSize(root)))
                                                                                                40
                                                                                                41
             protected override void SetFirst(TLink element) => ( header +
                                                                                                                     return GetZero();
                                                                                                42
                 LinksHeader.FirstFreeLinkOffset).SetValue(element);
                                                                                                43
                                                                                                                   while (!EqualToZero(root))
                                                                                                44
             protected override void SetLast(TLink element) => ( header +
32
                                                                                                45
              var left = GetLeftOrDefault(root);
                                                                                                46
                                                                                                                     var leftSize = GetSizeOrZero(left);
                                                                                                47
              protected override void SetPrevious(TLink element, TLink previous) =>
                                                                                                                     if (LessThan(index, leftSize))
                                                                                                48
                    links.GetElement(LinkSizeInBytes, element) +
                  Link.SourceOffset).SetValue(previous);
                                                                                                                        root = left:
                                                                                                50
                                                                                                                        continue:
                                                                                                5.1
             protected override void SetNext(TLink element, TLink next) =>
                                                                                                52
                    links.GetElement(LinkSizeInBytes, element) +
                                                                                                                        (IsEquals(index, leftSize))
                                                                                                53
                 Link.TargetOffset).SetValue(next);
                                                                                                54
                                                                                                                        return root;
                                                                                                5.5
             protected override void SetSize(TLink size) => ( header +
                                                                                                56
                 LinksHeader.FreeLinksOffset).SetValue(size);
                                                                                                                     root = GetRightOrDefault(root);
                                                                                                5.7
                                                                                                                     index = Subtract(index, Increment(leftSize));
                                                                                                58
40
```

```
return GetZero(): // TODO: Impossible situation exception (only if tree
                                                                                                      else
                                                                                  121
         structure broken)
                                                                                   122
                                                                                                         current = GetRightOrDefault(current);
                                                                                   123
                                                                                   124
                                                                                   125
// TODO: Return indices range instead of references count
                                                                                                     (!EqualToZero(first))
                                                                                   126
public TLink CalculateReferences(TLink link)
                                                                                   127
                                                                                                      current = first:
                                                                                   128
  var root = GetTreeRoot():
                                                                                                      while (true)
                                                                                   129
  var total = GetSize(root);
                                                                                   130
  var totalRightIgnore = GetZero():
                                                                                                        if (IsEquals(handler( memory.GetLinkStruct(current)), constants.Break))
                                                                                   131
  while (!EqualToZero(root))
                                                                                   132
                                                                                                            return constants.Break;
                                                                                   133
     var @base = GetBasePartValue(root);
                                                                                   134
                                                                                                         current = GetNext(current):
     if (LessOrEqualThan(@base, link))
                                                                                   135
                                                                                                         if (EqualToZero(current) || !IsEquals(GetBasePartValue(current), link))
                                                                                  136
        root = GetRightOrDefault(root);
                                                                                   137
                                                                                                            break:
                                                                                   138
     else
                                                                                   130
                                                                                   140
        totalRightIgnore = Add(totalRightIgnore, Increment(GetRightSize(root)));
                                                                                  141
                                                                                                    return constants. Continue;
        root = GetLeftOrDefault(root):
                                                                                   142
                                                                                   143
                                                                                  144
                                                                                                 protected override void PrintNodeValue(TLink node, StringBuilder sb)
                                                                                   145
  root = GetTreeRoot():
                                                                                  146
  var totalLeftIgnore = GetZero():
                                                                                                   sb.Append('');
                                                                                   147
  while (!EqualToZero(root))
                                                                                                   sb. Append((Links, Get Element (LinkSizeInBytes, node) +
                                                                                   148
                                                                                                    var @base = GetBasePartValue(root)
                                                                                                   sb.Append('-'):
     if (GreaterOrEqualThan(@base, link))
                                                                                   149
                                                                                                   \operatorname{sb.Append}('>')
                                                                                   150
                                                                                                   sb. Append((Links.GetElement(LinkSizeInBytes, node) +
        root = GetLeftOrDefault(root);
                                                                                  151
                                                                                                      Link.TargetOffset).GetValue<TLink>());
     else
                                                                                   152
                                                                                   153
        totalLeftIgnore = Add(totalLeftIgnore, Increment(GetLeftSize(root)));
                                                                                   154
                                                                                              private class LinksSourcesTreeMethods: LinksTreeMethodsBase
                                                                                   155
        root = GetRightOrDefault(root);
                                                                                   156
                                                                                                 public LinksSourcesTreeMethods(ResizableDirectMemoryLinks<TLink> memory
                                                                                   157
                                                                                                   : base(memory)
                                                                                  158
  return Subtract(Subtract(total, totalRightIgnore), totalLeftIgnore);
                                                                                  159
                                                                                   160
                                                                                   161
                                                                                                 protected override IntPtr GetLeftPointer(TLink node) =>
public TLink EachReference(TLink link, Func<IList<TLink>, TLink> handler)
                                                                                  162
                                                                                                 → Links.GetElement(LinkSizeInBytes, node) + Link.LeftAsSourceOffset;
  var root = GetTreeRoot();
                                                                                  163
                                                                                                 protected override IntPtr GetRightPointer(TLink node) =>
  if (EqualToZero(root))
                                                                                   164
                                                                                                     Links.GetElement(LinkSizeInBytes, node) + Link.RightAsSourceOffset;
     return constants.Continue;
                                                                                   165
                                                                                                 protected override TLink GetLeftValue(TLink node) =>
                                                                                   166
   TLink first = GetZero(), current = root;
                                                                                                     (Links.GetElement(LinkSizeInBytes, node) +
  while (!EqualToZero(current))
                                                                                                     Link.LeftAsSourceOffset).GetValue<TLink>();
                                                                                  167
                                                                                                 protected override TLink GetRightValue(TLink node) =>
     var @base = GetBasePartValue(current);
                                                                                   168
     if (GreaterOrEqualThan(@base, link))
                                                                                                     (Links.GetElement(LinkSizeInBytes, node) +
                                                                                                     Link.RightAsSourceOffset).GetValue<TLink>();
        if (IsEquals(@base, link))
                                                                                   169
                                                                                                 protected override TLink GetSize(TLink node)
                                                                                   170
           first = current;
                                                                                   171
                                                                                                   var previousValue = (Links.GetElement(LinkSizeInBytes, node) +
                                                                                   172
        current = GetLeftOrDefault(current);
                                                                                                    → Link.SizeAsSourceOffset).GetValue<TLink>();
                                                                                                    return BitwiseHelpers.PartialRead(previousValue, 5, -5);
                                                                                   173
```

65

67

77

92

93

100

101

102

103

104

105

106

107

109

110

111

112

113

116

117

118

119

```
174
                                                                                                  217
175
                                                                                                  218
              protected override void SetLeft(TLink node, TLink left) =>
176
                                                                                                  219
                   (Links.GetElement(LinkSizeInBytes, node) +
                                                                                                  220
                  Link.LeftAsSourceOffset).SetValue(left);
                                                                                                  221
                                                                                                  222
              protected override void SetRight(TLink node, TLink right) =>
178
                   (Links.GetElement(LinkSizeInBytes, node) +
                                                                                                  223
                  Link.RightAsSourceOffset).SetValue(right):
                                                                                                  224
              protected override void SetSize(TLink node, TLink size)
180
                                                                                                  225
181
                 var previousValue = (Links.GetElement(LinkSizeInBytes, node) +
182
                                                                                                  226
                 → Link.SizeAsSourceOffset).GetValue<TLink>():
                                                                                                  227
                 (Links.GetElement(LinkSizeInBytes, node) + Link.SizeAsSourceOffset).SetValu
                                                                                                 228
183
                                                                                                  229
                     e(BitwiseHelpers.PartialWrite(previousValue, size, 5,
                                                                                                  230
                     -5)):
184
                                                                                                  231
185
              protected override bool GetLeftIsChild(TLink node)
                                                                                                  232
187
                 var previousValue = (Links.GetElement(LinkSizeInBytes, node) +
                                                                                                  233
                 → Link.SizeAsSourceOffset).GetValue<TLink>():
                 return (Integer < TLink > ) Bitwise Helpers, Partial Read (previous Value, 4, 1):
189
190
191
              protected override void SetLeftIsChild(TLink node, bool value)
192
                                                                                                  234
                                                                                                  235
                 var previousValue = (Links.GetElement(LinkSizeInBytes, node) +
                                                                                                  236

→ Link.SizeAsSourceOffset).GetValue<TLink>():
                                                                                                  237
                 var modified = BitwiseHelpers.PartialWrite(previousValue,
195
                                                                                                  238
                     (TLink)(Integer<TLink>)value, 4, 1):
                 (Links.GetElement(LinkSizeInBytes, node) +
                                                                                                  239
                 → Link.SizeAsSourceOffset).SetValue(modified);
197
                                                                                                  240
198
                                                                                                  241
              protected override bool GetRightIsChild(TLink node)
199
200
                 var previousValue = (Links.GetElement(LinkSizeInBytes, node) +
201
                 → Link.SizeAsSourceOffset).GetValue<TLink>():
                 return (Integer<TLink>)BitwiseHelpers.PartialRead(previousValue, 3, 1):
202
                                                                                                  242
203
                                                                                                  243
204
                                                                                                  244
              protected override void SetRightIsChild(TLink node, bool value)
205
206
                                                                                                  245
                 var previousValue = (Links.GetElement(LinkSizeInBytes, node) +
                                                                                                  246
                 → Link.SizeAsSourceOffset).GetValue<TLink>():
                 var modified = BitwiseHelpers.PartialWrite(previousValue.
                     (TLink)(Integer<TLink>)value, 3, 1):
                                                                                                  247
                 (Links.GetElement(LinkSizeInBytes, node) +
                                                                                                  248
                 → Link.SizeAsSourceOffset).SetValue(modified);
                                                                                                  249
210
211
                                                                                                  250
              protected override sbyte GetBalance(TLink node)
212
                                                                                                  251
213
                                                                                                  252
                 var previousValue = (Links.GetElement(LinkSizeInBytes, node) +
                 → Link.SizeAsSourceOffset).GetValue<TLink>():
                 var value = (ulong)(Integer<TLink>)BitwiseHelpers.PartialRead(previousValue,
215
                 var unpackedValue = (sbyte)((value & 4) > 0? ((value & 4) << 5) | value & 3 |
                     124 : value & 3);
```

```
return unpackedValue;
protected override void SetBalance(TLink node, sbyte value)
   var previousValue = (Links.GetElement(LinkSizeInBytes, node) +

→ Link.SizeAsSourceOffset).GetValue<TLink>():
   var packagedValue = (TLink)(Integer<TLink>)((((byte)value >> 5) & 4)
   \rightarrow value & 3):
  var modified = BitwiseHelpers.PartialWrite(previousValue, packagedValue, 0, 3);
   (Links.GetElement(LinkSizeInBytes, node) +
   → Link.SizeAsSourceOffset).SetValue(modified);
protected override bool FirstIsToTheLeftOfSecond(TLink first, TLink second)
   var firstSource = (Links.GetElement(LinkSizeInBytes, first) +
   → Link.SourceOffset).GetValue<TLink>():
   var secondSource = (Links.GetElement(LinkSizeInBytes, second) +
   → Link.SourceOffset).GetValue<TLink>();
   return LessThan(firstSource, secondSource)
        (IsEquals(firstSource, secondSource) &&
            LessThan((Links.GetElement(LinkSizeInBytes, first) +
            Link.TargetOffset).GetValue<TLink>(),
            (Links.GetElement(LinkSizeInBytes, second) +
            Link.TargetOffset).GetValue<TLink>())):
protected override bool FirstIsToTheRightOfSecond(TLink first, TLink second)
   var firstSource = (Links.GetElement(LinkSizeInBvtes. first) +
   → Link.SourceOffset).GetValue<TLink>();
   var secondSource = (Links.GetElement(LinkSizeInBytes, second) +
   → Link.SourceOffset).GetValue<TLink>():
   return GreaterThan(firstSource, secondSource) ||
        (IsEquals(firstSource, secondSource) &&
            GreaterThan((Links.GetElement(LinkSizeInBytes, first) +
            Link.TargetOffset).GetValue<TLink>().
            (Links.GetElement(LinkSizeInBytes, second) +
            Link.TargetOffset).GetValue<TLink>())):
protected override TLink GetTreeRoot() => (Header +
    LinksHeader.FirstAsSourceOffset).GetValue<TLink>();
protected override TLink GetBasePartValue(TLink link) =>
    (Links.GetElement(LinkSizeInBytes, link) +
    Link.SourceOffset).GetValue<TLink>();
    <summary>
   Выполняет поиск и возвращает индекс связи с указанными Source
    (началом) и Target (концом)
   по дереву (индексу) связей, отсортированному по Source, а затем по Target.
    </summary>
 ///<param name="source">Индекс связи, которая является началом на
    искомой связи.</param>
/// <param name="target">Индекс связи, которая является концом на искомой
   связи.</param>
   <returns>Индекс искомой связи.</returns>
```

```
public TLink Search (TLink source, TLink target)
                                                                                     300
                                                                                     301
     var root = GetTreeRoot():
                                                                                     302
     while (!EqualToZero(root))
                                                                                     303
        var rootSource = (Links.GetElement(LinkSizeInBvtes, root) +
                                                                                     304
        → Link.SourceOffset).GetValue<TLink>():
                                                                                     305
        var rootTarget = (Links.GetElement(LinkSizeInBytes, root) +
                                                                                     306
            Link.TargetOffset).GetValue<TLink>():
        if (FirstIsToTheLeftOfSecond(source, target, rootSource, rootTarget)) //
                                                                                     307
            node.Kev < root.Kev
                                                                                     308
           root = GetLeftOrDefault(root):
        else if (FirstIsToTheRightOfSecond(source, target, rootSource, rootTarget))
                                                                                     309
               node.Key > root.Key
                                                                                     310
                                                                                     311
                                                                                     312
           root = GetRightOrDefault(root):
        else // node.Kev == root.Kev
                                                                                     313
           return root:
                                                                                     314
                                                                                     315
     return GetZero():
                                                                                     316
                                                                                     317
                                                                                     318
  [MethodImpl(MethodImplOptions.AggressiveInlining)]
  private bool FirstIsToTheLeftOfSecond(TLink firstSource, TLink firstTarget, TLink
       secondSource. TLink secondTarget) => LessThan(firstSource, secondSource) ||
                                                                                    320
       (IsEquals(firstSource, secondSource) && LessThan(firstTarget,
                                                                                     321
       secondTarget)):
                                                                                     322
                                                                                     323
  [MethodImpl(MethodImplOptions.AggressiveInlining)]
                                                                                     324
  private bool FirstIsToTheRightOfSecond(TLink firstSource, TLink firstTarget,
       TLink secondSource, TLink secondTarget) => GreaterThan(firstSource,
                                                                                     325
       secondSource) || (IsEquals(firstSource, secondSource) &&
      GreaterThan(firstTarget, secondTarget)):
                                                                                     326
                                                                                     327
private class LinksTargetsTreeMethods: LinksTreeMethodsBase
                                                                                     328
                                                                                     329
  public LinksTargetsTreeMethods(ResizableDirectMemoryLinks<TLink> memory
                                                                                     330
     : base(memory)
                                                                                     331
                                                                                     332
                                                                                     333
  protected override IntPtr GetLeftPointer(TLink node) =>
                                                                                     334
      Links.GetElement(LinkSizeInBytes, node) + Link.LeftAsTargetOffset;
                                                                                     335
                                                                                     336
  protected override IntPtr GetRightPointer(TLink node) =>
                                                                                     337
      Links.GetElement(LinkSizeInBytes, node) + Link.RightAsTargetOffset;
                                                                                     338
  protected override TLink GetLeftValue(TLink node) =>
       (Links.GetElement(LinkSizeInBytes, node) +
                                                                                     339
      Link.LeftAsTargetOffset).GetValue<TLink>();
                                                                                     340
  protected override TLink GetRightValue(TLink node) =>
                                                                                     341
       (Links.GetElement(LinkSizeInBytes, node) +
                                                                                     342
      Link.RightAsTargetOffset).GetValue<TLink>();
                                                                                     343
```

256

257

258

259

260

262

263

264

266

268

270

271

273

274

275

276

277

278

279

280

281

282

283 284

285

286

287

288

289

290

291

292

293

295

296

297

```
protected override TLink GetSize(TLink node)
  var previousValue = (Links.GetElement(LinkSizeInBytes. node) +
   → Link.SizeAsTargetOffset).GetValue<TLink>():
  return BitwiseHelpers.PartialRead(previousValue, 5, -5);
protected override void SetLeft(TLink node, TLink left) =>
    (Links.GetElement(LinkSizeInBytes, node) +
    Link.LeftAsTargetOffset).SetValue(left);
protected override void SetRight(TLink node, TLink right) =>
    (Links.GetElement(LinkSizeInBytes, node) +
    Link.RightAsTargetOffset).SetValue(right):
protected override void SetSize(TLink node, TLink size)
  var previousValue = (Links.GetElement(LinkSizeInBytes, node) +
  → Link.SizeAsTargetOffset).GetValue<TLink>():
  (Links.GetElement(LinkSizeInBytes, node) + Link.SizeAsTargetOffset).SetValu
      e(BitwiseHelpers.PartialWrite(previousValue, size, 5,
      -5));
protected override bool GetLeftIsChild(TLink node)
  var previousValue = (Links.GetElement(LinkSizeInBytes, node) +
      Link.SizeAsTargetOffset).GetValue<TLink>():
  return (Integer<TLink>)BitwiseHelpers.PartialRead(previousValue, 4, 1);
protected override void SetLeftIsChild(TLink node, bool value)
  var previousValue = (Links.GetElement(LinkSizeInBytes, node) +
   → Link.SizeAsTargetOffset).GetValue<TLink>():
  var modified = BitwiseHelpers.PartialWrite(previousValue,
      (TLink)(Integer<TLink>)value, 4, 1);
  (Links.GetElement(LinkSizeInBytes, node) +
   → Link.SizeAsTargetOffset).SetValue(modified):
protected override bool GetRightIsChild(TLink node)
  var previousValue = (Links.GetElement(LinkSizeInBytes, node) +
   → Link.SizeAsTargetOffset).GetValue<TLink>();
  return (Integer<TLink>)BitwiseHelpers.PartialRead(previousValue, 3, 1);
protected override void SetRightIsChild(TLink node, bool value)
  var previousValue = (Links.GetElement(LinkSizeInBytes, node) +
   → Link.SizeAsTargetOffset).GetValue<TLink>():
  var modified = BitwiseHelpers.PartialWrite(previousValue,
      (TLink)(Integer<TLink>)value, 3, 1);
  (Links.GetElement(LinkSizeInBytes, node) +
   → Link.SizeAsTargetOffset).SetValue(modified);
protected override sbyte GetBalance(TLink node)
```

```
var previousValue = (Links.GetElement(LinkSizeInBytes, node) +
344
                    Link.SizeAsTargetOffset).GetValue<TLink>():
                var value = (ulong)(Integer<TLink>)BitwiseHelpers.PartialRead(previousValue,
                var unpackedValue = (sbyte)((value & 4) > 0? ((value & 4) << 5) | value & 3 |
                    124 : value & 3):
                return unpackedValue;
3/18
349
              protected override void SetBalance(TLink node, sbyte value)
350
351
                var previousValue = (Links.GetElement(LinkSizeInBytes, node) +
                 → Link.SizeAsTargetOffset).GetValue<TLink>():
                var packagedValue = (TLink)(Integer<TLink>)((((byte)value >> 5) & 4)
353
                 \rightarrow value & 3):
                var modified = BitwiseHelpers.PartialWrite(previousValue, packagedValue, 0, 3);
354
                (Links.GetElement(LinkSizeInBytes, node) +
355
                    Link.SizeAsTargetOffset).SetValue(modified);
356
357
              protected override bool FirstIsToTheLeftOfSecond(TLink first, TLink second)
358
359
                var firstTarget = (Links.GetElement(LinkSizeInBytes, first) +
360
                 → Link.TargetOffset).GetValue<TLink>():
                var secondTarget = (Links.GetElement(LinkSizeInBytes, second) +
361
                 → Link.TargetOffset).GetValue<TLink>():
                return LessThan(firstTarget, secondTarget)
                      (IsEquals(first Target, second Target) &&
363
                          LessThan((Links.GetElement(LinkSizeInBytes, first) +
                          Link.SourceOffset).GetValue<TLink>().
                          (Links.GetElement(LinkSizeInBytes, second) +
                          Link.SourceOffset).GetValue<TLink>()));
364
365
              protected override bool FirstIsToTheRightOfSecond(TLink first, TLink second)
366
367
                var firstTarget = (Links.GetElement(LinkSizeInBytes, first) +
                 → Link.TargetOffset).GetValue<TLink>();
                var secondTarget = (Links.GetElement(LinkSizeInBytes, second) +
                 → Link.TargetOffset).GetValue<TLink>():
                return GreaterThan(firstTarget, secondTarget) ||
370
                      (IsEquals(firstTarget, secondTarget) &&
371
                          GreaterThan((Links.GetElement(LinkSizeInBytes, first) +
                          Link.SourceOffset).GetValue<TLink>(),
                          (Links.GetElement(LinkSizeInBytes, second) +
                          Link.SourceOffset).GetValue<TLink>()));
372
373
              protected override TLink GetTreeRoot() => (Header +
374
                 LinksHeader.FirstAsTargetOffset).GetValue<TLink>();
375
              protected override TLink GetBasePartValue(TLink link) =>
376
                  (Links.GetElement(LinkSizeInBytes, link) +
                 Link.TargetOffset).GetValue<TLink>();
378
379
```

./ResizableDirectMemory/UInt64ResizableDirectMemoryLinks.cs

1.0

11 12

13

14

15

17

18 19

20

21

22

23

24

25

26

27

28

29

30

3.1

32

33

34

35

36

37

38

39

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57 58

59

60

```
using System:
using System. Collections. Generic;
using System.Runtime.CompilerServices;
using Platform. Disposables;
using Platform. Collections. Arrays:
using Platform. Helpers. Singletons;
using Platform. Memory:
using Platform. Data. Exceptions:
using Platform.Data.Constants;
//#define ENABLE TREE AUTO DEBUG AND VALIDATION
#pragma warning disable 0649
#pragma warning disable 169
// ReSharper disable BuiltInTypeReferenceStyle
namespace Platform.Data.Doublets.ResizableDirectMemory
   using id = UInt64:
   public unsafe partial class UInt64ResizableDirectMemoryLinks: DisposableBase,
       ILinks<id>
         <summary>Возвращает размер одной связи в байтах.</summary>
          Используется только во вне класса, не рекомедуется использовать внутри.
          Так как во вне не обязательно будет доступен unsafe C#.
          </remarks>
      public static readonly int LinkSizeInBytes = sizeof(Link);
      public static readonly long DefaultLinksSizeStep = LinkSizeInBytes * 1024 * 1024:
      private struct Link
         public id Source:
         public id Target;
         public id Left AsSource;
         public id RightAsSource:
         public id SizeAsSource;
         public id LeftAsTarget:
         public id RightAsTarget;
         public id SizeAsTarget:
      private struct LinksHeader
         public id AllocatedLinks;
         public id ReservedLinks:
         public id FreeLinks;
         public id FirstFreeLink:
         public id FirstAsSource:
         public id FirstAsTarget:
         public id LastFreeLink;
         public id Reserved8;
      private readonly long memoryReservationStep;
      private readonly IResizableDirectMemory memory;
      private LinksHeader* header;
      private Link* links;
      private LinksTargetsTreeMethods targetsTreeMethods;
```

```
private LinksSourcesTreeMethods sourcesTreeMethods;
                                                                                  112
                                                                                  113
// TODO: Возможно чтобы гарантированно проверять на то, является ли связь
                                                                                  114
    удалённой, нужно использовать не список а дерево, так как так можно
                                                                                  115
    быстрее проверить на наличие связи внутри
                                                                                  116
private UnusedLinksListMethods unusedLinksListMethods;
                                                                                  117
                                                                                  118
   <summary>
   Возвращает общее число связей находящихся в хранилище.
                                                                                  120
                                                                                  121
private id Total => header->AllocatedLinks - header->FreeLinks;
                                                                                  122
                                                                                  123
// TODO: Дать возможность переопределять в конструкторе
                                                                                  124
public LinksCombinedConstants<id, id, int> Constants { get; }
                                                                                  125
                                                                                  126
public UInt64ResizableDirectMemoryLinks(string address): this(address,
                                                                                  127
    DefaultLinksSizeStep) { }
                                                                                  128
                                                                                  129
   <summary>
                                                                                  130
   Создаёт экземпляр базы данных Links в файле по указанному адресу, с
                                                                                  131
   указанным минимальным шагом расширения базы данных.
                                                                                  132
                                                                                  133
   cparam name="address">Полный пусть к файлу базы данных.
/// <param name="memoryReservationStep">Минимальный шаг расширения базы
                                                                                  135
   данных в байтах.</param>
                                                                                  136
public UInt64ResizableDirectMemoryLinks(string address, long
                                                                                  137
    memoryReservationStep): this(new FileMappedResizableDirectMemory(address,
                                                                                  138
    memoryReservationStep), memoryReservationStep) { }
                                                                                  139
                                                                                  140
public UInt64ResizableDirectMemoryLinks(IResizableDirectMemory memory):
                                                                                  141
   this(memory, DefaultLinksSizeStep) { }
                                                                                  142
                                                                                  143
public UInt64ResizableDirectMemoryLinks(IResizableDirectMemory memory, long
                                                                                  144
    memoryReservationStep)
                                                                                  145
                                                                                  146
  Constants = Default < LinksCombinedConstants < id, id, int >>.Instance;
                                                                                  147
   memory = memory;
                                                                                  148
    memoryReservationStep = memoryReservationStep;
                                                                                  149
    (memory Reserved Capacity < memory Reservation Step)
                                                                                  150
                                                                                  151
     memory.ReservedCapacity = memoryReservationStep;
                                                                                  152
                                                                                  153
  Set Pointers (memory);
                                                                                  154
     Гарантия корректности memory. UsedCapacity относительно
                                                                                  155
        header->AllocatedLinks
                                                                                  156
   memory.UsedCapacity = ((long) header->AllocatedLinks * sizeof(Link)) +
                                                                                  157
  \rightarrow sizeof(LinksHeader):
                                                                                  158
   // Гарантия корректности header->ReservedLinks относительно
                                                                                  159
        memory.ReservedCapacity
                                                                                  160
   header->ReservedLinks = (id)((memory.ReservedCapacity -
                                                                                  161

→ sizeof(LinksHeader)) / sizeof(Link));

                                                                                  162
                                                                                  163
                                                                                  164
[MethodImpl(MethodImplOptions.AggressiveInlining)]
                                                                                  165
public id Count(IList<id>restrictions)
                                                                                  166
                                                                                  167
   // Если нет ограничений, тогда возвращаем общее число связей находящихся в 168
   → хранилише.
                                                                                  169
  if (restrictions.Count == 0)
                                                                                  170
                                                                                  171
     return Total;
                                                                                  172
                                                                                  173
```

66

67

68

69

70

71

72

73

74

77

82

91

92

100

102

103

104

105

106

107

109

110

```
if (restrictions.Count == 1)
  var index = restrictions[Constants.IndexPart];
  if (index == Constants.Any)
     return Total:
  return Exists(index) ? 1UL : 0UL;
if (restrictions.Count == 2)
  var index = restrictions[Constants.IndexPart]
  var value = restrictions[1];
  if (index == Constants.Any)
     if (value == Constants.Anv)
        return Total; // Any - как отсутствие ограничения
     return sourcesTreeMethods.CalculateReferences(value)
         + targetsTreeMethods.CalculateReferences(value);
  else
     if (!Exists(index))
        return 0:
       (value == Constants.Anv)
        return 1;
     var storedLinkValue = GetLinkUnsafe(index):
     if (storedLinkValue->Source == value ||
        storedLinkValue->Target == value)
        return 1:
     return 0:
if (restrictions.Count == 3)
  var index = restrictions[Constants.IndexPart];
  var source = restrictions[Constants.SourcePart];
  var target = restrictions[Constants.TargetPart];
  if (index == Constants.Anv)
     if (source == Constants.Any && target == Constants.Any)
        return Total:
     else if (source == Constants.Any)
        return targetsTreeMethods.CalculateReferences(target);
     else if (target == Constants.Anv)
        return sourcesTreeMethods.CalculateReferences(source);
     else //if(source != Any && target != Any)
```

```
// Эквивалент Exists(source, target) => Count(Any, source, target) > 0
                                                                                                      if (restrictions.Count == 1)
           var link = sourcesTreeMethods.Search(source, target):
                                                                                       237
           return link \equiv Constants.Null? 0UL: 1UL:
                                                                                                         var index = restrictions[Constants.IndexPart];
                                                                                       238
                                                                                                         if (index == Constants.Any)
                                                                                       239
                                                                                       240
     else
                                                                                                            return Each(handler, ArrayPool<ulong>.Empty):
                                                                                       241
                                                                                       242
        if (!Exists(index))
                                                                                                         if (!Exists(index))
                                                                                       243
                                                                                       244
           return 0:
                                                                                                            return Constants.Continue:
                                                                                       245
                                                                                       246
         if (source == Constants. Any && target == Constants. Any)
                                                                                                         return handler(GetLinkStruct(index));
                                                                                       247
                                                                                       248
           return 1:
                                                                                                        (restrictions.Count == 2)
                                                                                       249
                                                                                       250
        var storedLinkValue = GetLinkUnsafe(index);
                                                                                                         var index = restrictions[Constants.IndexPart]
                                                                                       251
        if (source != Constants.Any && target != Constants.Any)
                                                                                                         var value = restrictions[1];
                                                                                       252
                                                                                                         if (index == Constants.Anv)
                                                                                       253
           if (storedLinkValue->Source == source &&
                                                                                       254
              storedLinkValue->Target == target)
                                                                                                            if (value == Constants.Anv)
                                                                                       255
                                                                                       256
              return 1:
                                                                                                               return Each(handler, ArrayPool<ulong>.Empty);
                                                                                       257
                                                                                       258
           return 0;
                                                                                                             (Each(handler, new] { index, value, Constants.Any }) == Constants.Break)
                                                                                       259
                                                                                       260
        var value = default(id):
                                                                                                               return Constants.Break:
                                                                                       261
        if (source == Constants.Any)
                                                                                       262
                                                                                                            return Each(handler, new[] { index, Constants.Any, value });
                                                                                       263
           value = target;
                                                                                       264
                                                                                       265
         if (target == Constants.Anv)
                                                                                                            if (!Exists(index))
                                                                                       267
           value = source:
                                                                                       268
                                                                                                               return Constants.Continue;
                                                                                       269
         if (storedLinkValue->Source == value |
                                                                                       270
           storedLinkValue->Target == value)
                                                                                                            if (value == Constants.Any)
                                                                                       271
                                                                                       272
           return 1:
                                                                                                               return handler(GetLinkStruct(index));
                                                                                       273
                                                                                       274
        return 0:
                                                                                                            var storedLinkValue = GetLinkUnsafe(index);
                                                                                       275
                                                                                                            if (storedLinkValue->Source == value |
                                                                                       276
                                                                                                               storedLinkValue->Target == value)
                                                                                       277
  throw new NotSupportedException ("Другие размеры и способы ограничений не
                                                                                       278
      поддерживаются.");
                                                                                                               return handler(GetLinkStruct(index));
                                                                                       279
                                                                                       280
                                                                                                            return Constants.Continue;
                                                                                       281
[MethodImpl(MethodImplOptions.AggressiveInlining)]
                                                                                       282
public id Each(Func<IList<id>, id> handler, IList<id> restrictions)
                                                                                       283
                                                                                                      if (restrictions.Count == 3)
                                                                                       284
    (restrictions.Count == 0)
                                                                                       285
                                                                                                         var index = restrictions[Constants.IndexPart];
                                                                                       286
      for (id link = 1; link \leq header->AllocatedLinks; link++)
                                                                                                         var source = restrictions[Constants.SourcePart]
                                                                                       287
                                                                                                         var target = restrictions[Constants.TargetPart];
                                                                                       288
        if (Exists(link))
                                                                                                         if (index == Constants.Any)
                                                                                       289
                                                                                       290
           if (handler(GetLinkStruct(link)) == Constants.Break)
                                                                                       291
                                                                                                            if (source == Constants.Any && target == Constants.Any)
                                                                                       292
              return Constants.Break;
                                                                                                               return Each(handler, ArrayPool<ulong>.Empty);
                                                                                       293
                                                                                       294
                                                                                                            else if (source == Constants.Any)
                                                                                       295
                                                                                       296
     return Constants.Continue;
```

 $\frac{234}{235}$

```
return targetsTreeMethods.EachReference(target, handler);
                                                                                                     var linkIndex = values[Constants.IndexPart];
                                                                                       355
                                                                                                     var link = GetLinkUnsafe(linkIndex):
                                                                                       356
        else if (target == Constants.Anv)
                                                                                                     // Будет корректно работать только в том случае, если пространство
                                                                                       357
                                                                                                     → выделенной связи предварительно заполнено нулями
           return sourcesTreeMethods.EachReference(source, handler);
                                                                                                     if (link->Source != Constants.Null)
                                                                                       358
                                                                                       359
        else //if(source!= Any && target!= Any)
                                                                                                        sourcesTreeMethods.Detach(new IntPtr(& header->FirstAsSource), linkIndex);
                                                                                       360
                                                                                       361
           var link = sourcesTreeMethods.Search(source, target);
                                                                                                     if (link->Target != Constants.Null)
                                                                                       362
           return link == Constants. Null? Constants. Continue:
                                                                                       363
            → handler(GetLinkStruct(link));
                                                                                                        targetsTreeMethods.Detach(new IntPtr(& header->FirstAsTarget), linkIndex);
                                                                                       364
                                                                                       365
                                                                                            #if ENABLE TREE AUTO DEBUG AND VALIDATION
                                                                                       366
                                                                                                     var leftTreeSize = sourcesTreeMethods.GetSize(new
     else
                                                                                       367
                                                                                                     \rightarrow IntPtr(& header->FirstAsSource)):
        if (!Exists(index))
                                                                                                     var rightTreeSize = targetsTreeMethods.GetSize(new
                                                                                                     \rightarrow IntPtr(& header->FirstAsTarget)):
           return Constants.Continue;
                                                                                                     if (leftTreeSize! = rightTreeSize)
                                                                                       369
                                                                                       370
          (source == Constants.Any && target == Constants.Any)
                                                                                                        throw new Exception("One of the trees is broken.");
                                                                                       371
                                                                                      372
           return handler(GetLinkStruct(index));
                                                                                            #endif
                                                                                      373
                                                                                                     link->Source = values[Constants.SourcePart];
                                                                                       374
        var storedLinkValue = GetLinkUnsafe(index):
                                                                                                     link-Target = values[Constants.TargetPart]
                                                                                       375
        if (source != Constants.Any && target != Constants.Any)
                                                                                                     if (link->Source != Constants.Null)
                                                                                       376
                                                                                       377
           if (storedLinkValue->Source == source &&
                                                                                                        sourcesTreeMethods.Attach(new IntPtr(& header->FirstAsSource), linkIndex);
                                                                                       378
              storedLinkValue->Target == target)
                                                                                       379
                                                                                       380
                                                                                                     if (link->Target != Constants.Null)
              return handler(GetLinkStruct(index));
                                                                                       381
                                                                                                        targetsTreeMethods.Attach(new IntPtr(& header->FirstAsTarget), linkIndex);
                                                                                       382
           return Constants.Continue:
                                                                                       383
                                                                                            \# {
m if} \; {
m ENABLE} \; \; {
m TREE} \; \; {
m AUTO} \; \; {
m DEBUG} \; \; {
m AND} \; \; {
m VALIDATION}
                                                                                       384
        var value = default(id);
                                                                                                     leftTreeSize = sourcesTreeMethods.GetSize(new
                                                                                       385
        if (source == Constants.Any)
                                                                                                     \rightarrow IntPtr(& header->FirstAsSource));
                                                                                       386
                                                                                                     rightTreeSize = targetsTreeMethods.GetSize(new
           value = target;
                                                                                                     \rightarrow IntPtr(& header->FirstAsTarget)):
                                                                                                     if (leftTreeSize!= rightTreeSize)
        if (target == Constants.Anv)
                                                                                       388
                                                                                                        throw new Exception ("One of the trees is broken.");
                                                                                       389
           value = source;
                                                                                      390
                                                                                            #endif
                                                                                      391
         if (storedLinkValue->Source == value ||
                                                                                                     return linkIndex;
                                                                                       392
           storedLinkValue->Target == value)
                                                                                       393
                                                                                       394
           return handler(GetLinkStruct(index));
                                                                                                  |MethodImpl(MethodImplOptions.AggressiveInlining)|
                                                                                       395
                                                                                                  private IList<id>GetLinkStruct(id linkIndex)
                                                                                       396
        return Constants.Continue;
                                                                                       397
                                                                                                     var link = GetLinkUnsafe(linkIndex);
                                                                                       398
                                                                                                     return new UInt64Link(linkIndex, link->Source, link->Target);
                                                                                       399
  throw new NotSupportedException ("Другие размеры и способы ограничений не
                                                                                       400
   → поддерживаются.");
                                                                                       401
                                                                                                  [MethodImpl(MethodImplOptions, AggressiveInlining)]
                                                                                       402
                                                                                                  private Link* GetLinkUnsafe(id linkIndex) => & links[linkIndex];
                                                                                       403
                                                                                       404
    TODO: Возможно можно перемещать значения, если указан индекс, но
                                                                                                     <remarks>
                                                                                       405
    значение существует в другом месте (но не в менеджере памяти, а в логике
                                                                                                      TODO: Возможно нужно будет заполнение нулями, если внешнее API ими не
                                                                                                   → заполняет пространство
    Links)
                                                                                       407
                                                                                                   ^{\prime}//</\mathrm{remarks}>
// </remarks>
                                                                                                  public id Create()
                                                                                       408
[MethodImpl(MethodImplOptions.AggressiveInlining)]
                                                                                       409
public id Update(IList<id>values)
```

299

300

302

303

304

306

307

308

309

311

313

315

317

320

321

322

323

324

325

326

327

328

329

330

331

332

333

334

335

336

337

338

340

341

342

343

344

345

346

347

348

349

350

351

352

```
var freeLink = header->FirstFreeLink;
                                                                                                      unusedLinksListMethods = null;
                                                                                     470
  if (freeLink != \overline{C}onstants.Null)
                                                                                     471
                                                                                     472
                                                                                                   else
      unusedLinksListMethods.Detach(freeLink):
                                                                                     473
                                                                                                       header = (LinksHeader*)(void*)memory.Pointer:
                                                                                    474
  else
                                                                                                       - links = (Link*)(void*)memory.Pointer;
                                                                                     475
                                                                                                       sourcesTreeMethods = new LinksSourcesTreeMethods(this);
                                                                                    476
     if ( header->AllocatedLinks > Constants.MaxPossibleIndex)
                                                                                                       targetsTreeMethods = new LinksTargetsTreeMethods(this);
                                                                                     477
                                                                                                       unusedLinksListMethods = new UnusedLinksListMethods( links, header);
                                                                                     478
        throw new LinksLimitReachedException(Constants.MaxPossibleIndex);
                                                                                     479
                                                                                     480
       ( header->AllocatedLinks >= header->ReservedLinks - 1)
                                                                                     481
                                                                                                [MethodImpl(MethodImplOptions.AggressiveInlining)]
                                                                                     482
                                                                                                private bool Exists(id link) => link >= Constants.MinPossibleIndex && link <=
         memory.ReservedCapacity += memoryReservationStep;
                                                                                     483
                                                                                                → header->AllocatedLinks && !IsUnusedLink(link);
        SetPointers( memory);
         header->ReservedLinks = (id)( memory.ReservedCapacity / sizeof(Link));
                                                                                    484
                                                                                                [MethodImpl(MethodImplOptions.AggressiveInlining)]
                                                                                     485
       header->AllocatedLinks++:
                                                                                               private bool IsUnusedLink(id link) => header->FirstFreeLink == link
                                                                                     486
                                                                                                                        | ( links[link] SizeAsSource == Constants.Null &&
       memory UsedCapacity += sizeof(Link);
                                                                                     487
     freeLink = header->AllocatedLinks;
                                                                                                                         → links[link].Source != Constants.Null):
  return freeLink;
                                                                                               #region Disposable
                                                                                     489
                                                                                     490
                                                                                               protected override bool AllowMultipleDisposeCalls => true;
                                                                                     491
                                                                                     492
public void Delete(id link)
                                                                                                protected override void DisposeCore(bool manual, bool wasDisposed)
                                                                                     493
                                                                                    494
  if (link < header->AllocatedLinks)
                                                                                                   if (!wasDisposed)
                                                                                     495
      unusedLinksListMethods.AttachAsFirst(link);
                                                                                     496
                                                                                                      Set Pointers (null):
                                                                                     497
                                                                                     498
  else if (link == header->AllocatedLinks)
                                                                                                   Disposable. Try Dispose (memory);
                                                                                     499
      header->AllocatedLinks--:
                                                                                    500
                                                                                    501
      memory UsedCapacity -= sizeof(Link):
                                                                                                #endregion
                                                                                    502
     // Убираем все связи, находящиеся в списке свободных в конце файла, до
                                                                                     503
      → тех пор, пока не дойдём до первой существующей связи
                                                                                    504
     // Позволяет оптимизировать количество выделенных связей (AllocatedLinks)
     while ( header->AllocatedLinks > 0 &&
         IsUnusedLink( header->AllocatedLinks))
                                                                                      ./ResizableDirectMemory/UInt64ResizableDirectMemoryLinks.ListMethods.cs
                                                                                          using Platform.Collections.Methods.Lists;
          unusedLinksListMethods.Detach( header->AllocatedLinks);
         header->AllocatedLinks--:
                                                                                          namespace Platform.Data.Doublets.ResizableDirectMemory
        memory.UsedCapacity -= sizeof(Link);
                                                                                             unsafe partial class UInt64ResizableDirectMemoryLinks
                                                                                                private class UnusedLinksListMethods: CircularDoublyLinkedListMethods<ulong>
                                                                                                   private readonly Link* links;
   TODO: Возможно это должно быть событием, вызываемым из IMemory, в том
                                                                                                   private readonly LinksHeader* header;
    случае, если адрес реально поменялся
                                                                                     1.1
                                                                                                   public UnusedLinksListMethods(Link* links, LinksHeader* header)
                                                                                     12
   Указатель this.links может быть в том же месте,
                                                                                     13
   так как 0-я связь не используется и имеет такой же размер как Header,
                                                                                                       links = links;
                                                                                     14
   поэтому header размешается в том же месте, что и 0-я связь
                                                                                                      \overline{\phantom{a}} header = header;
                                                                                     15
   </remarks>
                                                                                     16
private void SetPointers(IResizableDirectMemory memory)
                                                                                     17
                                                                                                   protected override ulong GetFirst() => header->FirstFreeLink;
                                                                                     18
  if (memory == null)
                                                                                     19
                                                                                                   protected override ulong GetLast() => header->LastFreeLink;
                                                                                     20
       header = null;
                                                                                     21
      -links = null;
                                                                                                   protected override ulong GetPrevious(ulong element) => links[element].Source;
                                                                                     22
       unusedLinksListMethods = null;
                                                                                     23
                                                                                                   protected override ulong GetNext(ulong element) => links[element]. Target;
      \mathsf{TtargetsTreeMethods} = \mathsf{null};
                                                                                     24
```

411

412

413

414

415

416

418

419

420

421

422

423

424

425

426

428

429 430

431

432

433

434

435

436

437

438

439

441

442

443

445

446

447

449

450 451

452 453

454 455

456

457

458

459

460

461

462

463

464

465

466

467

```
43
             protected override ulong GetSize() => header->FreeLinks;
                                                                                                  44
27
                                                                                                  45
             protected override void SetFirst(ulong element) => header->FirstFreeLink =
28
              \rightarrow element:
                                                                                                  47
                                                                                                  48
             protected override void SetLast(ulong element) => header->LastFreeLink =
                                                                                                  49
                element;
                                                                                                  50
                                                                                                  51
             protected override void SetPrevious(ulong element, ulong previous) =>
                                                                                                  52
                  links[element].Source = previous;
                                                                                                  53
                                                                                                  54
             protected override void SetNext(ulong element, ulong next) =>
34
                                                                                                  55
                 links[element].Target = next;
                                                                                                  57
             protected override void SetSize(ulong size) => header->FreeLinks = size;
                                                                                                  58
37
38
                                                                                                  59
39
                                                                                                  60
                                                                                                  61
./ResizableDirectMemory/UInt64ResizableDirectMemoryLinks.TreeMethods.cs
                                                                                                  62
                                                                                                  63
    using System:
    using System.Collections.Generic;
                                                                                                  64
    using System.Runtime.CompilerServices;
                                                                                                  65
    using System. Text:
                                                                                                  66
    using Platform.Collections.Methods.Trees;
                                                                                                  67
    using Platform.Data.Constants;
                                                                                                  68
                                                                                                  69
    namespace Platform.Data.Doublets.ResizableDirectMemory
                                                                                                  70
                                                                                                  71
       unsafe partial class UInt64ResizableDirectMemoryLinks
                                                                                                  72
1.1
                                                                                                  73
          private abstract class LinksTreeMethodsBase:
                                                                                                  74
           → SizedAndThreadedAVLBalancedTreeMethods<ulong>
                                                                                                  75
                                                                                                  76
             private readonly UInt64ResizableDirectMemoryLinks memory:
                                                                                                  77
             private readonly LinksCombinedConstants<ulong, ulong, int> constants;
                                                                                                  78
             protected readonly Link* Links:
                                                                                                  79
             protected readonly LinksHeader* Header;
                                                                                                  80
                                                                                                  81
             protected LinksTreeMethodsBase(UInt64ResizableDirectMemoryLinks memory
19
                                                                                                  82
20
                                                                                                  83
                Links = memory. links;
^{21}
                                                                                                  84
                Header = memory header;
22
                                                                                                  85
                 memory = memory;
23
                                                                                                  86
                -constants = memory.Constants;
24
                                                                                                  87
25
                                                                                                  88
             [MethodImpl(MethodImplOptions.AggressiveInlining)]
                                                                                                  89
27
             protected abstract ulong GetTreeRoot();
28
                                                                                                  91
             [MethodImpl(MethodImplOptions.AggressiveInlining)]
30
                                                                                                  92
             protected abstract ulong GetBasePartValue(ulong link);
31
                                                                                                  93
32
                                                                                                  94
             public ulong this ulong index
33
                                                                                                  95
34
                                                                                                  96
                                                                                                  97
36
                                                                                                  98
                   var root = GetTreeRoot();
                   if (index >= GetSize(root))
                                                                                                 100
                                                                                                 101
                      return 0;
                                                                                                 102
                                                                                                 103
                   while (root != 0)
```

```
var left = GetLeftOrDefault(root);
        var leftSize = GetSizeOrZero(left);
        if (index < leftSize)
           root = left;
           continue:
          (index == leftSize)
           return root:
        root = GetRightOrDefault(root);
        index -= leftSize + 1:
     return 0; // TODO: Impossible situation exception (only if tree structure
      → broken)
// TODO: Return indices range instead of references count
public ulong CalculateReferences(ulong link)
  var root = GetTreeRoot():
  var total = GetSize(root);
  var totalRightIgnore = 0UL:
  while (root != 0)
     var @base = GetBasePartValue(root);
     if (@base \leq = link)
        root = GetRightOrDefault(root):
        totalRightIgnore += GetRightSize(root) + 1;
        root = GetLeftOrDefault(root);
  root = GetTreeRoot();
  var totalLeftIgnore = 0UL;
  while (root != 0)
     var @base = GetBasePartValue(root);
     if (@base >= link)
        root = GetLeftOrDefault(root);
     else
        totalLeftIgnore += GetLeftSize(root) + 1;
        root = GetRightOrDefault(root);
  return total - totalRightIgnore - totalLeftIgnore;
public ulong EachReference(ulong link, Func<IList<ulong>, ulong> handler)
  var root = GetTreeRoot();
  if (root == 0)
```

```
return constants. Continue;
                                                                                                      protected override ulong GetRightValue(ulong node) => Links[node].RightAsSource;
                                                                                        165
                                                                                        166
                                                                                                       protected override ulong GetSize(ulong node)
     ulong first = 0, current = \text{root}:
                                                                                        167
     while (current != 0)
                                                                                        168
                                                                                                          var previousValue = Links[node].SizeAsSource;
                                                                                        169
         var @base = GetBasePartValue(current);
                                                                                        170
                                                                                                          //return MathHelpers.PartialRead(previousValue, 5, -5):
                                                                                                          return (previous Value & 4294967264) >> 5:
         if (@base >= link)
                                                                                        171
                                                                                        172
                                                                                        173
           if (@base == link)
                                                                                                       protected override void SetLeft(ulong node, ulong left) =>
                                                                                        174
                                                                                                           Links[node].LeftAsSource = left;
               first = current;
                                                                                        175
                                                                                                       protected override void SetRight(ulong node, ulong right) =>
                                                                                        176
           current = GetLeftOrDefault(current):
                                                                                                       → Links[node].RightAsSource = right;
                                                                                        177
                                                                                                       protected override void SetSize(ulong node, ulong size)
                                                                                        178
                                                                                        179
           current = GetRightOrDefault(current);
                                                                                                          var previousValue = Links[node].SizeAsSource;
                                                                                        180
                                                                                                          //var modified = MathHelpers.PartialWrite(previousValue, size, 5, -5);
                                                                                        181
                                                                                                          var modified = (previous Value & 31) | ((size & 134217727) << 5);
                                                                                        182
     if (first !=0)
                                                                                                          Links[node].SizeAsSource = modified;
                                                                                        183
        current = first:
                                                                                        184
                                                                                        185
         while (true)
                                                                                                       protected override bool GetLeftIsChild(ulong node)
                                                                                        186
                                                                                        187
           if (handler( memory.GetLinkStruct(current)) == constants.Break)
                                                                                                          var previousValue = Links[node].SizeAsSource;
                                                                                        188
                                                                                                          //return (Integer)MathHelpers.PartialRead(previousValue, 4, 1);
                                                                                        189
              return constants.Break;
                                                                                                          return (previous Value & 16) >> 4 == 1UL:
                                                                                        190
                                                                                        191
           current = GetNext(current):
                                                                                        192
           if (current == 0 || GetBasePartValue(current) != link)
                                                                                                       protected override void SetLeftIsChild(ulong node, bool value)
                                                                                        193
                                                                                        194
               break:
                                                                                                          var previousValue = Links[node].SizeAsSource;
                                                                                        195
                                                                                                          //var modified = MathHelpers.PartialWrite(previousValue,
                                                                                        196
                                                                                                          \rightarrow (ulong)(Integer)value, 4, 1):
                                                                                                          var modified = (previous Value & 4294967279) | ((value ? 1UL : 0UL) <<4);
     return constants.Continue;
                                                                                        197
                                                                                                          Links[node].SizeAsSource = modified;
                                                                                        198
                                                                                        199
  protected override void PrintNodeValue(ulong node, StringBuilder sb)
                                                                                        200
                                                                                                       protected override bool GetRightIsChild(ulong node)
                                                                                        201
     sb.Append('')
                                                                                        202
                                                                                                          var previousValue = Links[node].SizeAsSource:
     sb.Append(Links[node].Source);
                                                                                        203
                                                                                                          //return (Integer)MathHelpers.PartialRead(previousValue, 3, 1);
     sb.Append('-');
                                                                                        204
                                                                                                          return (previous Value & 8) >> 3 == 1UL;
     \operatorname{sb.Append}(')>'):
                                                                                        205
     sb.Append(Links[node].Target);
                                                                                        206
                                                                                        207
                                                                                                       protected override void SetRightIsChild(ulong node, bool value)
                                                                                        208
                                                                                        209
                                                                                                          var previousValue = Links[node].SizeAsSource;
private class LinksSourcesTreeMethods: LinksTreeMethodsBase
                                                                                        210
                                                                                                          //var modified = MathHelpers.PartialWrite(previousValue,
                                                                                        211
  public LinksSourcesTreeMethods(UInt64ResizableDirectMemoryLinks memory)
                                                                                                          \rightarrow (ulong)(Integer)value, 3, 1):
                                                                                                          var modified = (previous Value & 4294967287) | ((value ? 1UL : 0UL) << 3);
     : base(memory)
                                                                                        212
                                                                                                          Links[node] SizeAsSource = modified;
                                                                                        213
                                                                                        214
                                                                                        215
  protected override IntPtr GetLeftPointer(ulong node) => new
                                                                                                       protected override sbyte GetBalance(ulong node)
                                                                                        216
   → IntPtr(&Links[node].LeftAsSource);
                                                                                        217
                                                                                                          var previousValue = Links[node].SizeAsSource;
                                                                                        218
  protected override IntPtr GetRightPointer(ulong node) => new
                                                                                                          //var value = MathHelpers.PartialRead(previousValue, 0, 3);
                                                                                        219
      IntPtr(&Links[node].RightAsSource);
                                                                                        220
                                                                                                          var value = previous Value & 7:
                                                                                                         var unpackedValue = (sbyte)((value & 4) > 0? ((value & 4) << 5) | value & 3 |
                                                                                        221
  protected override ulong GetLeftValue(ulong node) => Links[node].LeftAsSource;
                                                                                                          \rightarrow 124 : value & 3):
```

```
return unpackedValue;
                                                                                   277
protected override void SetBalance(ulong node, sbyte value)
                                                                                   279
  var previousValue = Links[node].SizeAsSource:
                                                                                   280
  var packagedValue = (ulong)((((byte)value >> 5) & 4) | value & 3);
                                                                                   281
  //var modified = MathHelpers.PartialWrite(previousValue, packagedValue, 0, 3);
  var modified = (previous Value & 4294967288) | (packaged Value & 7);
                                                                                   282
  Links[node].SizeAsSource = modified;
                                                                                   283
                                                                                   284
protected override bool FirstIsToTheLeftOfSecond(ulong first, ulong second)
                                                                                   285
  => Links[first].Source < Links[second].Source ||
                                                                                   286
    (Links[first].Source == Links[second].Source && Links[first].Target <
                                                                                   287
    → Links[second]. Target);
                                                                                   288
                                                                                   289
protected override bool FirstIsToTheRightOfSecond(ulong first, ulong second)
                                                                                   290
   => Links[first].Source > Links[second].Source ||
                                                                                   291
    (Links[first].Source == Links[second].Source && Links[first].Target >
                                                                                   292
    \rightarrow Links[second]. Target):
                                                                                   293
                                                                                   294
protected override ulong GetTreeRoot() => Header->FirstAsSource;
                                                                                   295
                                                                                   296
protected override ulong GetBasePartValue(ulong link) => Links[link].Source;
                                                                                   297
                                                                                   298
                                                                                   299
   Выполняет поиск и возвращает индекс связи с указанными Source
                                                                                   300
    (началом) и Target (концом)
                                                                                   301
   по дереву (индексу) связей, отсортированному по Source, а затем по Target. 302
   </summary>
 // <param name="source">Индекс связи, которая является началом на
                                                                                   303
→ искомой связи.</param>
                                                                                   304
/// <param name="target">Индекс связи, которая является концом на искомой
                                                                                   305
→ связи.</param>
                                                                                   307
 /// <returns>Индекс искомой связи.</returns>
                                                                                   308
public ulong Search(ulong source, ulong target)
                                                                                   309
                                                                                   310
  var root = Header->FirstAsSource;
                                                                                   311
  while (root != 0)
                                                                                   312
                                                                                   313
      var rootSource = Links[root].Source;
                                                                                   314
      var rootTarget = Links[root].Target;
                                                                                   315
     if (FirstIsToTheLeftOfSecond(source, target, rootSource, rootTarget)) //
                                                                                   316
      → node.Key < root.Key
                                                                                   317
        root = GetLeftOrDefault(root);
                                                                                   318
                                                                                   319
     else if (FirstIsToTheRightOfSecond(source, target, rootSource, rootTarget))
                                                                                   320
          // node.Key > root.Key
                                                                                   321
        root = GetRightOrDefault(root);
                                                                                   322
                                                                                   323
      else // node.Kev == root.Kev
                                                                                   324
        return root:
                                                                                   325
                                                                                   326
  return 0;
                                                                                   328
[MethodImpl(MethodImplOptions.AggressiveInlining)]
```

 $\frac{223}{224}$

225

226

227

228

229

231

232

233

234

235

236

237

238

239

240

241

 242

243

244

245

246

247

248

249

250

251

252

253

254

255

256

257

258

259

260

261

264

265

266

268

270

271

272

274

275

276

```
private static bool FirstIsToTheLeftOfSecond(ulong firstSource, ulong firstTarget
    ulong secondSource, ulong secondTarget)
   => firstSource < secondSource || (firstSource == secondSource && firstTarget
   \rightarrow < secondTarget);
[MethodImpl(MethodImplOptions, AggressiveInlining)]
private static bool FirstIsToTheRightOfSecond(ulong firstSource, ulong firstTarget,
→ ulong secondSource, ulong secondTarget)
   => firstSource > secondSource || (firstSource == secondSource && firstTarget
   \Rightarrow > secondTarget):
[MethodImpl(MethodImplOptions.AggressiveInlining)]
protected override void ClearNode(ulong node)
   Links[node].LeftAsSource = 0UL:
   Links nodel Right AsSource = 0UL:
   Links[node].SizeAsSource = 0UL;
[MethodImpl(MethodImplOptions.AggressiveInlining)]
protected override ulong GetZero() => 0UL:
[MethodImpl(MethodImplOptions.AggressiveInlining)]
protected override ulong GetOne() => 1UL:
[MethodImpl(MethodImplOptions.AggressiveInlining)]
protected override ulong GetTwo() => 2UL:
[MethodImpl(MethodImplOptions.AggressiveInlining)]
protected override bool ValueEqualToZero(IntPtr pointer) =>
\rightarrow *(ulong*)pointer. ToPointer() == 0UL;
[MethodImpl(MethodImplOptions.AggressiveInlining)]
protected override bool EqualToZero(ulong value) => value == 0UL:
[MethodImpl(MethodImplOptions.AggressiveInlining)]
protected override bool IsEquals(ulong first, ulong second) => first == second;
[MethodImpl(MethodImplOptions.AggressiveInlining)]
protected override bool GreaterThanZero(ulong value) => value > 0UL;
[MethodImpl(MethodImplOptions.AggressiveInlining)]
protected override bool GreaterThan(ulong first, ulong second) => first > second;
[MethodImpl(MethodImplOptions, AggressiveInlining)]
protected override bool GreaterOrEqualThan(ulong first, ulong second) => first
\Rightarrow >= second:
[MethodImpl(MethodImplOptions, AggressiveInlining)]
protected override bool GreaterOrEqualThanZero(ulong value) => true; // value
\Rightarrow >= 0 is always true for ulong
[MethodImpl(MethodImplOptions.AggressiveInlining)]
protected override bool LessOrEqualThanZero(ulong value) => value == 0; //
\rightarrow value is always >= 0 for ulong
[MethodImpl(MethodImplOptions.AggressiveInlining)]
protected override bool LessOrEqualThan(ulong first, ulong second) => first <=
\rightarrow second:
[MethodImpl(MethodImplOptions.AggressiveInlining)]
protected override bool LessThanZero(ulong value) => false; // value < 0 is always
```

→ false for ulong

```
384
  [MethodImpl(MethodImplOptions.AggressiveInlining)]
                                                                                       385
  protected override bool LessThan(ulong first, ulong second) => first < second:
                                                                                      386
                                                                                      387
  [MethodImpl(MethodImplOptions.AggressiveInlining)]
                                                                                       388
  protected override ulong Increment(ulong value) => ++value:
                                                                                       389
                                                                                       390
  [MethodImpl(MethodImplOptions.AggressiveInlining)]
                                                                                       391
  protected override ulong Decrement(ulong value) => --value;
                                                                                       392
                                                                                       393
  [MethodImpl(MethodImplOptions.AggressiveInlining)]
                                                                                       394
  protected override ulong Add(ulong first, ulong second) => first + second;
                                                                                       395
                                                                                       396
  [MethodImpl(MethodImplOptions.AggressiveInlining)]
                                                                                       397
  protected override ulong Subtract(ulong first, ulong second) => first - second;
                                                                                       398
                                                                                       399
                                                                                       400
private\ class\ Links Targets Tree Methods: Links Tree Methods Base
                                                                                       401
                                                                                       402
  public LinksTargetsTreeMethods(UInt64ResizableDirectMemoryLinks memory
                                                                                       403
     : base(memory)
                                                                                       404
                                                                                       405
                                                                                       406
                                                                                       407
  //protected override IntPtr GetLeft(ulong node) => new
                                                                                       408
   → IntPtr(&Links[node].LeftAsTarget);
   //protected override IntPtr GetRight(ulong node) => new
                                                                                       409
                                                                                       410
   → IntPtr(&Links[node].RightAsTarget);
                                                                                       411
                                                                                       412
  //protected override ulong GetSize(ulong node) => Links[node].SizeAsTarget;
                                                                                       413
  //protected override void SetLeft(ulong node, ulong left) =>
                                                                                       414
                                                                                       415
   → Links[node].LeftAsTarget = left;
                                                                                       416
  //protected override void SetRight(ulong node, ulong right) =>
                                                                                       417
   → Links[node].RightAsTarget = right;
                                                                                       418
                                                                                       419
  //protected override void SetSize(ulong node, ulong size) =>
                                                                                       420
      Links[node].SizeAsTarget = size:
                                                                                       421
                                                                                       422
  protected override IntPtr GetLeftPointer(ulong node) => new
                                                                                       423
                                                                                       424
   → IntPtr(&Links[node].LeftAsTarget);
                                                                                       425
  protected override IntPtr GetRightPointer(ulong node) => new
                                                                                       426
                                                                                       427
   → IntPtr(&Links[node].RightAsTarget);
  protected override ulong GetLeftValue(ulong node) => Links[node].LeftAsTarget:
                                                                                       429
 protected override ulong GetRightValue(ulong node) => Links[node].RightAsTarget;
                                                                                      430
                                                                                       431
  protected override ulong GetSize(ulong node)
                                                                                       432
                                                                                       433
     var previousValue = Links[node].SizeAsTarget;
                                                                                       434
      //return MathHelpers.PartialRead(previousValue, 5, -5);
                                                                                       435
     return (previous Value & 4294967264) >> 5;
                                                                                       436
                                                                                       437
                                                                                       438
  protected override void SetLeft(ulong node, ulong left) =>
      Links[node].LeftAsTarget = left;
                                                                                       439
                                                                                       440
  protected override void SetRight(ulong node, ulong right) =>
                                                                                       441
      Links[node].RightAsTarget = right;
                                                                                       442
```

331

332

333

334

335

336

337

338

339

340

341

342

 343

344

 345

346

347

348

349

350

351

352

353

354

355

356

357

358

359

360

361

362

364

365

367

368

369

370

371

372

373

374

375

376

377

378

379

380

```
protected override void SetSize(ulong node, ulong size)
  var previousValue = Links[node].SizeAsTarget;
   //var modified = MathHelpers.PartialWrite(previousValue, size, 5, -5);
  var modified = (previous Value & 31) | ((size & 134217727) << 5);
   Links[node].SizeAsTarget = modified;
protected override bool GetLeftIsChild(ulong node)
  var previousValue = Links[node].SizeAsTarget:
   //return (Integer)MathHelpers.PartialRead(previousValue, 4, 1);
  return (previous Value & 16) >> 4 == 1UL;
   // TODO: Check if this is possible to use
    /var nodeSize = GetSize(node);
    /var left = GetLeftValue(node)
    /var leftSize = GetSizeOrZero(left);
    //return leftSize > 0 && nodeSize > leftSize;
protected override void SetLeftIsChild(ulong node, bool value)
  var previousValue = Links[node].SizeAsTarget;
   //var modified = MathHelpers.PartialWrite(previousValue.
   \hookrightarrow (ulong)(Integer)value, 4, 1);
  var modified = (previous Value & 4294967279) | ((value ? 1UL : 0UL) <<4);
   Links[node].SizeAsTarget = modified:
protected override bool GetRightIsChild(ulong node)
  var previousValue = Links[node].SizeAsTarget;
   //return (Integer)MathHelpers.PartialRead(previousValue, 3, 1):
   return (previous Value & 8) >> 3 == 1UL:
   // TODO: Check if this is possible to use
    //var nodeSize = GetSize(node);
    /var right = GetRightValue(node)
    /var rightSize = GetSizeOrZero(right):
   //return rightSize > 0 && nodeSize > rightSize:
protected override void SetRightIsChild(ulong node, bool value)
   var previousValue = Links[node].SizeAsTarget;
   //var modified = MathHelpers.PartialWrite(previousValue,
   \rightarrow (ulong)(Integer)value, 3, 1):
  var modified = (previous Value & 4294967287) | ((value ? 1UL : 0UL) << 3);
  Links[node].SizeAsTarget = modified;
protected override sbyte GetBalance(ulong node)
  var previousValue = Links[node].SizeAsTarget;
   //var value = MathHelpers.PartialRead(previousValue, 0, 3):
  var value = previousValue & 7:
  var unpackedValue = (sbyte)((value & 4) > 0? ((value & 4) << 5) | value & 3 |
   \rightarrow 124 : value & 3):
  return unpackedValue;
protected override void SetBalance(ulong node, sbyte value)
```

```
28
                var previousValue = Links[node].SizeAsTarget;
                                                                                                                 Keep creating layer after layer
                                                                                                 29
444
                var packagedValue = (ulong)((((byte)value >> 5) & 4) | value & 3);
                                                                                                               while (length > 2)
445
                                                                                                 3.0
                //var modified = MathHelpers.PartialWrite(previousValue, packagedValue, 0, 3);
446
                                                                                                 31
                 var modified = (previous Value & 4294967288) | (packaged Value & 7);
                                                                                                                  HalveSequence(sequence, sequence, length);
                                                                                                 32
447
                 Links[node].SizeAsTarget = modified:
                                                                                                                 length = (length / 2) + (length \% 2):
                                                                                                 33
448
449
                                                                                                 34
450
                                                                                                               return Links.GetOrCreate(sequence[0], sequence[1]):
                                                                                                 35
              protected override bool FirstIsToTheLeftOfSecond(ulong first, ulong second)
451
                                                                                                 36
                 => Links[first].Target < Links[second].Target |
452
                                                                                                 37
                  (Links[first].Target == Links[second].Target && Links[first].Source <
                                                                                                            private void HalveSequence(IList<TLink> destination, IList<TLink> source, int
453
                                                                                                 38
                  → Links[second].Source);
454
                                                                                                 39
              protected override bool FirstIsToTheRightOfSecond(ulong first, ulong second)
                                                                                                               var loopedLength = length - (length \% 2);
455
                                                                                                 40
                 => Links[first]. Target > Links[second]. Target ||
                                                                                                               for (var i = 0; i < loopedLength; i += 2)
456
                                                                                                 41
                  (Links[first].Target == Links[second].Target && Links[first].Source >
                                                                                                 42
                  → Links[second].Source):
                                                                                                                  destination[i / 2] = Links.GetOrCreate(source[i], source[i + 1]);
                                                                                                 43
                                                                                                 44
              protected override ulong GetTreeRoot() => Header->FirstAsTarget;
459
                                                                                                               if (length > loopedLength)
                                                                                                 45
460
                                                                                                 46
              protected override ulong GetBasePartValue(ulong link) => Links[link]. Target:
461
                                                                                                                  destination[length / 2] = source[length - 1]:
                                                                                                 47
462
                                                                                                 48
              [MethodImpl(MethodImplOptions.AggressiveInlining)]
                                                                                                 49
              protected override void ClearNode(ulong node)
464
                                                                                                 50
                                                                                                 51
                 Links[node].LeftAsTarget = 0UL;
                 Links node Right As Target = 0 UL;
                 Links[node].SizeAsTarget = 0UL;
                                                                                                  ./Sequences/Converters/CompressingConverter.cs
469
                                                                                                     using System:
470
                                                                                                      using System.Collections.Generic;
471
                                                                                                      using System.Runtime.CompilerServices:
                                                                                                      using Platform.Interfaces;
                                                                                                      using Platform. Collections:
./Sequences/Converters/BalancedVariantConverter.cs
                                                                                                      using Platform. Helpers. Singletons;
     using System.Collections.Generic;
                                                                                                      using Platform Numbers:
                                                                                                      using Platform Data Constants;
     namespace Platform.Data.Doublets.Sequences.Converters
                                                                                                      using Platform.Data.Doublets.Sequences.Frequencies.Cache;
                                                                                                 10
        public class BalancedVariantConverter<TLink>:
                                                                                                      namespace Platform.Data.Doublets.Sequences.Converters
            LinksListToSequenceConverterBase<TLink>
                                                                                                 12
                                                                                                 13
           public BalancedVariantConverter(ILinks<TLink> links) : base(links) { }
                                                                                                            TODO: Возможно будет лучше если алгоритм будет выполняться полностью
                                                                                                 14
                                                                                                         → изолированно от Links на этапе сжатия.
           public override TLink Convert(IList<TLink> sequence)
                                                                                                 1.5
                                                                                                               А именно будет создаваться временный список пар необходимых для
                                                                                                             выполнения сжатия, в таком случае тип значения элемента массива может быть
              var length = sequence.Count;
                                                                                                             любым, как char так и ulong.
              if (length < 1)
                                                                                                               Как только список/словарь пар был выявлен можно разом выполнить
                                                                                                            создание всех этих пар, а так же разом выполнить замену
                 return default;
                                                                                                         /// < / \text{remarks} >
                                                                                                 17
                                                                                                         public class CompressingConverter<TLink>:
                (length == 1)
                                                                                                             LinksListToSequenceConverterBase<TLink>
                                                                                                 19
                 return sequence[0];
                                                                                                            private static readonly LinksCombinedConstants<br/>
slood, TLink, long> constants =
                                                                                                 20
                                                                                                            → Default<LinksCombinedConstants<br/>
<br/>bool, TLink, long>>.Instance:
                Make copy of next layer
                                                                                                           private static readonly EqualityComparer<TLink> equalityComparer =
                                                                                                 ^{21}
              if (length > 2)
21
                                                                                                            → EqualityComparer<TLink>.Default:
                                                                                                            private static readonly Comparer<TLink> comparer = Comparer<TLink>.Default;
                 // TODO: Try to use stackalloc (which at the moment is not working with
                                                                                                 23

    generics) but will be possible with Sigil

                                                                                                            private readonly IConverter</List<TLink>, TLink> baseConverter;
                                                                                                 24
                 var halvedSequence = new TLink[(length / 2) + (length \% 2)];
                                                                                                            private readonly LinkFrequenciesCache<TLink> doubletFrequenciesCache;
                                                                                                 25
                 HalveSequence(halvedSequence, sequence, length);
                                                                                                            private readonly TLink minFrequencyToCompress;
                                                                                                 26
                 sequence = halvedSequence;
                                                                                                            private readonly bool doInitialFrequenciesIncrement;
                                                                                                 27
                 length = halvedSequence.Length:
                                                                                                            private Doublet<TLin\overline{k}> maxDoublet;
```

```
private LinkFrequency<TLink> maxDoubletData;
                                                                                                      return sequence;
                                                                                      84
private struct HalfDoublet
                                                                                                   if (sequence.Count == 2)
                                                                                      85
   public TLink Element:
                                                                                                      return new[] { Links.GetOrCreate(sequence[0], sequence[1]) };
                                                                                      87
  public LinkFrequency<TLink> DoubletData:
                                                                                      88
                                                                                                    // TODO: arraypool with min size (to improve cache locality) or stackallow with
   public HalfDoublet(TLink element, LinkFrequency<TLink> doubletData)
                                                                                                   var copy = new HalfDoublet[sequence.Count];
     Element = element:
                                                                                                   Doublet \leq TLink \geq doublet = default:
                                                                                     9.1
     Doublet Data = doublet Data:
                                                                                                   for (var i = 1; i < sequence.Count; i++)
                                                                                      92
                                                                                      93
                                                                                                      doublet.Source = sequence[i - 1]:
                                                                                      94
  public override string ToString() => \$"\{Element\}: (\{DoubletData\})";
                                                                                                      doublet.Target = sequencelil:
                                                                                      95
                                                                                                      LinkFrequency<TLink> data;
                                                                                      96
                                                                                                      if (doInitialFrequenciesIncrement)
                                                                                      97
public CompressingConverter(ILinks<TLink> links, IConverter<IList<TLink>,
    TLink > baseConverter, LinkFrequenciesCache < TLink >
                                                                                      98
                                                                                                        data = doubletFrequenciesCache.IncrementFrequency(ref doublet);
                                                                                      99
    doubletFrequenciesCache)
                                                                                     100
   : this(links, baseConverter, doubletFrequenciesCache, Integer<TLink>.One, true)
                                                                                                      else
                                                                                     101
                                                                                     102
                                                                                                        data = doubletFrequenciesCache.GetFrequency(ref doublet);
                                                                                     103
                                                                                                         if (data = null)
                                                                                     104
public CompressingConverter(ILinks<TLink> links, IConverter<IList<TLink>,
                                                                                     105
    TLink baseConverter, LinkFrequenciesCache<TLink>
                                                                                                          throw new NotSupportedException("If you ask not to increment frequencies.
                                                                                     106
    doubletFrequenciesCache, bool doInitialFrequenciesIncrement)
                                                                                                            → it is expected that all frequencies for the sequence are prepared.");
   : this(links, baseConverter, doubletFrequenciesCache, Integer<TLink>.One,
                                                                                     107

→ doInitialFrequenciesIncrement)

                                                                                     108
                                                                                                      copv[i-1]. Element = sequence[i-1];
                                                                                     109
                                                                                                      copv[i-1].DoubletData = data;
                                                                                     110
                                                                                                      UpdateMaxDoublet(ref doublet, data);
public CompressingConverter(ILinks<TLink> links, IConverter<IList<TLink>,
                                                                                     111
                                                                                     112
    TLink> baseConverter, LinkFrequenciesCache<TLink>
                                                                                                   copy[sequence.Count - 1].Element = sequence[sequence.Count - 1];
    doubletFrequenciesCache, TLink minFrequencyToCompress, bool
                                                                                     113
                                                                                                   copy sequence.Count - 1 DoubletData = new LinkFrequency < TLink > ();
                                                                                     114
    doInitialFrequenciesIncrement)
                                                                                                      comparer.Compare( maxDoubletData.Frequency, default) > 0)
                                                                                     115
   : base(links)
                                                                                     116
                                                                                                      var newLength = ReplaceDoublets(copy);
                                                                                     117
    baseConverter = baseConverter:
                                                                                                      sequence = new TLink[newLength]
    doubletFrequenciesCache = doubletFrequenciesCache;
                                                                                     118
                                                                                                      for (int i = 0: i < \text{newLength}: i++)
    (comparer.Compare(minFrequencyToCompress, Integer<TLink>.One) < 0)
                                                                                     119
                                                                                     120
                                                                                                        sequence[i] = copy[i].Element;
     minFrequencyToCompress = Integer<TLink>.One;
                                                                                     121
                                                                                     122
    minFrequencyToCompress = minFrequencyToCompress;
                                                                                     123
                                                                                                   return sequence;
    doInitialFrequenciesIncrement = doInitialFrequenciesIncrement;
                                                                                     124
   \overline{R}eset MaxDoublet():
                                                                                     125
                                                                                     126
                                                                                                    <remarks>
                                                                                     127
                                                                                                   Original algorithm idea: https://en.wikipedia.org/wiki/Byte pair encoding
public override TLink Convert(IList<TLink> source) =>
                                                                                     128
                                                                                     129
    baseConverter.Convert(Compress(source));
                                                                                                private int ReplaceDoublets(HalfDoublet | copy)
                                                                                     130
                                                                                     131
                                                                                                   var oldLength = copy.Length;
   Original algorithm idea: https://en.wikipedia.org/wiki/Byte_pair_encoding.
                                                                                     132
                                                                                                   var newLength = copy.Length;
                                                                                     133
   Faster version (doublets' frequencies dictionary is not recreated).
                                                                                                   while (comparer.Compare(maxDoubletData.Frequency, default) > 0)
                                                                                     134
   </remarks>
                                                                                     135
private IList<TLink> Compress(IList<TLink> sequence)
                                                                                                      var maxDoubletSource = maxDoublet.Source;
                                                                                     136
                                                                                                      var maxDoubletTarget = maxDoublet.Target;
                                                                                     137
  if (sequence.IsNullOrEmpty())
                                                                                                      if (equalityComparer.Equals(maxDoubletData.Link, constants.Null))
                                                                                     138
                                                                                     139
     return null;
                                                                                                          maxDoubletData.Link = Links.GetOrCreate(maxDoubletSource,
                                                                                     140
                                                                                                         → maxDoubletTarget):
    (sequence.Count == 1)
```

3.1

32

33

34

35

37

38

39

41

42

43

45

47

5.1

5.3

54

55

57

58

63

67

71

72

73

75

77

```
UpdateMaxDoublet(ref doublet, copy[i - 1].DoubletData);
141
                 var maxDoubletReplacementLink = maxDoubletData.Link;
142
                                                                                                 199
                oldLength--:
143
                                                                                                 200
                var oldLengthMinusTwo = oldLength - 1;
144
                                                                                                 201
                  / Substitute all usages
                                                                                                            [MethodImpl(MethodImplOptions.AggressiveInlining)]
                                                                                                 202
145
                                                                                                            private void UpdateMaxDoublet(ref Doublet<TLink> doublet.
                 int w = 0, r = 0; // (r == read, w == write)
146
                                                                                                 203
                 for (r < \text{oldLength}; r++)
                                                                                                             → LinkFrequency<TLink> data)
147
148
                                                                                                 204
                    if (equalityComparer.Equals(copy[r].Element, maxDoubletSource) &&
                                                                                                               var frequencv = data.Frequencv:
149
                                                                                                 205
                         equalityComparer.Equals(copy[r + 1].Element, maxDoubletTarget))
                                                                                                               var maxFrequency = maxDoubletData.Frequency;
                                                                                                 206
                                                                                                                //if (frequency > minFrequency ToCompress && (maxFrequency < frequency |
                                                                                                 207
150
                      if (r > 0)
                                                                                                                    (\max Frequency == frequency \&\& doublet.Source + doublet.Target < /* gives
151
                                                                                                                    better compression string data (and gives collisions quickly) */
152
                         var previous = copv[w - 1].Element;
153
                                                                                                                     maxDoublet.Source + maxDoublet.Target)))
                         copy[w - 1].DoubletData.DecrementFrequency():
154
                                                                                                               if (
                                                                                                                   comparer.Compare(frequency, minFrequencyToCompress) > 0 &&
                                                                                                 208
                         copv[w - 1].DoubletData =
155
                                                                                                                    comparer. Compare (\max Frequency, frequency) < 0 \parallel
                                                                                                 209
                               doubletFrequenciesCache.IncrementFrequency(previous.
                                                                                                                      ( equalityComparer.Equals(maxFrequency, frequency) &&
                              maxDoubletReplacementLink):
                                                                                                                       comparer. Compare (ArithmeticHelpers, Add (doublet. Source,
                                                                                                                      doublet.Target), ArithmeticHelpers.Add( maxDoublet.Source,
                       if (r < oldLengthMinusTwo)
157
                                                                                                                       \max Doublet.Target) > 0))) /* gives better stability and better
                                                                                                                      compression on sequent data and even on rundom numbers data (but gives
                         var next = copv[r + 2]. Element:
                                                                                                                      collisions anyway) */
                         copv[r + 1].DoubletData.DecrementFrequency();
160
                         copy [w]. Doublet Data = doublet Frequencies Cache. Increment Frequenc_
161
                                                                                                                    \max Doublet = doublet:
                                                                                                 211
                              y(maxDoubletReplacementLink,
                                                                                                                   -\max Doublet Data = data;
                                                                                                 212
                              next);
                                                                                                 213
                                                                                                 214
                       copy[w++]. Element = maxDoubletReplacementLink;
163
                                                                                                 215
164
                                                                                                 216
                       newLength--;
165
166
                                                                                                  ./Sequences/Converters/LinksListToSequenceConverterBase.cs
167
                                                                                                      using System.Collections.Generic:
                      \operatorname{copy}[w++] = \operatorname{copy}[r];
169
                                                                                                      using Platform.Interfaces;
                                                                                                      namespace Platform. Data. Doublets. Sequences. Converters
171
                 if (w < newLength)
172
                                                                                                         public abstract class LinksListToSequenceConverterBase<TLink>:
173
                                                                                                             IConverter<IList<TLink>, TLink>
                    copy|w| = copy|r|;
174
175
                                                                                                            protected readonly ILinks<br/>
TLink> Links;
                 oldLength = newLength;
176
                 Reset MaxDoublet():
                                                                                                            public LinksListToSequenceConverterBase(ILinks<TLink> links) => Links = links;
177
                 UpdateMaxDoublet(copy, newLength);
                                                                                                            public abstract TLink Convert(IList<TLink> source);
                                                                                                  11
179
              return newLength;
180
                                                                                                  12
181
182
                                                                                                  ./Sequences/Converters/OptimalVariantConverter.cs
           [MethodImpl(MethodImplOptions.AggressiveInlining)]
183
                                                                                                      using System. Collections. Generic:
           private void ResetMaxDoublet()
184
                                                                                                      using System Ling;
185
                                                                                                      using Platform.Interfaces;
               \max Doublet = new Doublet < TLink > ();
186
187
               \max Doublet Data = new LinkFrequency < TLink > ();
                                                                                                       namespace Platform.Data.Doublets.Sequences.Converters
188
189
                                                                                                         public class OptimalVariantConverter<TLink> :
           [MethodImpl(MethodImplOptions.AggressiveInlining)]
190
                                                                                                             LinksListToSequenceConverterBase<TLink>
           private void UpdateMaxDoublet(HalfDoublet[] copy, int length)
191
192
                                                                                                            private static readonly EqualityComparer<TLink> equalityComparer =
              Doublet < TLink > doublet = default;
193
                                                                                                             → EqualityComparer<TLink>.Default:
              for (var i = 1; i < length; i++)
194
                                                                                                            private static readonly Comparer<TLink> comparer = Comparer<TLink> Default;
                                                                                                  1.0
195
                                                                                                  11
                 doublet Source = copv[i - 1]. Element:
196
                                                                                                            private readonly IConverter<IList<TLink>>
                 doublet.Target = copy[i].Element;
                                                                                                             ⇒ sequenceToItsLocalElementLevelsConverter;
```

```
72
public OptimalVariantConverter(ILinks<TLink> links, IConverter<IList<TLink>>
                                                                                                             else
                                                                                      73

⇒ sequenceToItsLocalElementLevelsConverter) : base(links)

                                                                                      74
                                                                                                               sequence[w] = sequence[i - 1];
  => sequenceToItsLocalElementLevelsConverter =
                                                                                      75
                                                                                                               levels[w] = levels[i - 1]
   → sequenceToItsLocalElementLevelsConverter:
                                                                                      76
                                                                                                               previousLevel = levels[w];
                                                                                      77
                                                                                                               w++:
public override TLink Convert(IList<TLink> sequence)
                                                                                      78
                                                                                      79
                                                                                                               (i == length - 1)
  var length = sequence.Count:
                                                                                      80
  if (length == 1)
                                                                                      81
                                                                                                               sequence[w] = sequence[i];
                                                                                      82
                                                                                                               levels[w] = levels[i];
     return sequence[0];
                                                                                      83
                                                                                      84
  var links = Links;
  if (length == 2)
                                                                                      86
                                                                                      87
                                                                                                       length = w:
                                                                                      88
     return links.GetOrCreate(sequence[0], sequence[1]);
                                                                                      89
                                                                                                    return links.GetOrCreate(sequence[0], sequence[1]);
  sequence = sequence. To Array():
                                                                                      90
  var levels = sequenceToItsLocalElementLevelsConverter.Convert(sequence);
                                                                                      91
                                                                                      92
  while (length > 2)
                                                                                                 private static TLink GetGreatestNeigbourLowerThanCurrentOrCurrent(TLink
                                                                                      93
                                                                                                     previous, TLink current, TLink next)
     var levelRepeat = 1:
     var currentLevel = levels[0]
                                                                                      94
                                                                                                    return comparer.Compare(previous, next) > 0
                                                                                      95
     var previousLevel = levels[0]:
                                                                                                       ? comparer.Compare(previous, current) < 0 ? previous : current
     var skipOnce = false;
                                                                                                       : comparer.Compare(next, current) < 0 ? next : current:
     var \mathbf{w} = 0:
                                                                                      97
     for (var i = 1: i < length: i++)
                                                                                      98
                                                                                      99
                                                                                                 private static TLink GetNextLowerThanCurrentOrCurrent(TLink current, TLink
        if ( equalityComparer.Equals(currentLevel, levels[i]))
                                                                                      100
                                                                                                 \rightarrow next) => comparer.Compare(next, current) < 0 ? next : current;
           levelRepeat++:
                                                                                      101
                                                                                                 private static TLink GetPreviousLowerThanCurrentOrCurrent(TLink previous, TLink
           skipOnce = false:
                                                                                      102
           if (levelRepeat == 2)
                                                                                                 → current) => comparer.Compare(previous, current) < 0 ? previous : current;
                                                                                      103
              sequence[w] = links.GetOrCreate(sequence[i - 1], sequence[i]);
                                                                                      104
              var newLevel = i > = length - 1?
                 GetPreviousLowerThanCurrentOrCurrent(previousLevel,
                 \rightarrow currentLevel):
                                                                                      ./Sequences/Converters/SequenceToltsLocalElementLevelsConverter.cs
                 i < 2?
                                                                                           using System.Collections.Generic:
                 GetNextLowerThanCurrentOrCurrent(currentLevel, levels[i + 1]):
                                                                                           using Platform.Interfaces;
                 GetGreatestNeigbourLowerThanCurrentOrCurrent(previousLevel,
                 \rightarrow currentLevel, levels[i + 1]);
                                                                                           namespace Platform.Data.Doublets.Sequences.Converters
              levels[w] = newLevel:
                                                                                       5
              previousLevel = currentLevel;
                                                                                              public class SequenceToItsLocalElementLevelsConverter<TLink>:
                                                                                                  LinksOperatorBase<TLink>, IConverter<IList<TLink>>
              level Repeat = 0;
              skipOnce = true;
                                                                                                 private static readonly Comparer<TLink> comparer = Comparer<TLink>.Default;
                                                                                                 private readonly IConverter Doublet TLink, TLink
           else if (i == length - 1)
                                                                                                       linkToItsFrequencyToNumberConveter;
                                                                                                 public SequenceToItsLocalElementLevelsConverter(ILinks<TLink> links,
              sequence[w] = sequence[i];
                                                                                                     IConverter < Doublet < TLink >, TLink > linkToItsFrequencyToNumberConveter)
              levels[w] = levels[i];
                                                                                                     : base(links) => linkToItsFrequencyToNumberConveter =
                                                                                                     linkToItsFrequencyToNumberConveter;
                                                                                                 public IList<TLink> Convert(IList<TLink> sequence)
                                                                                      1.1
        else
                                                                                      12
                                                                                                    var levels = new TLink[sequence.Count];
                                                                                      13
           currentLevel = levels[i];
                                                                                                    levels[0] = GetFrequencyNumber(sequence[0], sequence[1]);
                                                                                      14
           levelRepeat = 1;
                                                                                                    for (var i = 1; i < sequence.Count - 1; i++)
                                                                                      15
           if (skipOnce)
                                                                                      16
                                                                                                       var previous = GetFrequencyNumber(sequence[i - 1], sequence[i]);
                                                                                      17
              skipOnce = false;
                                                                                                       var next = GetFrequencyNumber(sequence[i], sequence[i + 1]);
                                                                                      18
```

15

17

19

2.1

23

24

25

27

20

31

32

3.3

34

3.5

3.8

39

41

42

5.2

5.3

54

5.7

```
levels[i] = comparer.Compare(previous, next) > 0? previous : next;
                                                                                                     public class DefaultSequenceAppender<TLink>: LinksOperatorBase<TLink>,
             levels[levels.Length - 1] = GetFrequencyNumber(sequence[sequence.Count - 2],
                                                                                                         ISequenceAppender<TLink>
21
             → sequence|sequence.Count - 1|):
                                                                                                        private static readonly EqualityComparer<TLink> equalityComparer =
             return levels;
                                                                                              10
                                                                                                         → EqualityComparer<TLink>.Default:
23
24
                                                                                              11
                                                                                                        private readonly IStack<TLink> stack;
          public TLink GetFrequencyNumber(TLink source, TLink target) =>
                                                                                              12
                                                                                                        private readonly ISequenceHeightProvider<TLink> heightProvider;
                linkToItsFrequencyToNumberConveter.Convert(new Doublet<TLink>(source,
                                                                                              13
                                                                                              14
              target));
                                                                                                        public DefaultSequenceAppender(ILinks<TLink> links, IStack<TLink> stack,
                                                                                              15
                                                                                                           ISequenceHeightProvider<TLink> heightProvider)
27
                                                                                                           : base(links)
                                                                                              16
                                                                                              17
./Sequences/CreteriaMatchers/DefaultSequenceElementCreteriaMatcher.cs
                                                                                                            stack = stack:
                                                                                              18
    using Platform.Interfaces;
                                                                                                            heightProvider = heightProvider:
                                                                                              19
                                                                                              20
    namespace Platform.Data.Doublets.Sequences.CreteriaMatchers
                                                                                              21
                                                                                                         public TLink Append(TLink sequence, TLink appendant)
                                                                                              22
       public class DefaultSequenceElementCreteriaMatcher<TLink>:
                                                                                              23
           LinksOperatorBase<TLink>, ICreteriaMatcher<TLink>
                                                                                                           var cursor = sequence:
                                                                                              24
                                                                                                           while (! equalityComparer.Equals( heightProvider.Get(cursor), default))
                                                                                              25
         public DefaultSequenceElementCreteriaMatcher(ILinks<TLink> links) : base(links) { }
                                                                                              26
                                                                                                              var source = Links.GetSource(cursor);
          public bool IsMatched(TLink argument) => Links.IsPartialPoint(argument);
                                                                                              27
                                                                                                              var target = Links.GetTarget(cursor):
                                                                                              28
                                                                                                              if (equalityComparer.Equals(heightProvider.Get(source),
                                                                                              29
                                                                                                                  heightProvider.Get(target)))
./Sequences/CreteriaMatchers/MarkedSequenceCreteriaMatcher.cs
                                                                                              30
                                                                                                                break:
                                                                                              31
    using System Collections Generic:
                                                                                              32
    using Platform.Interfaces;
                                                                                                              else
                                                                                              33
    namespace Platform.Data.Doublets.Sequences.CreteriaMatchers
                                                                                                                  stack.Push(source);
                                                                                                                \overline{c}ursor = target;
       public class MarkedSequenceCreteriaMatcher<TLink>: ICreteriaMatcher<TLink>
                                                                                              36
                                                                                              37
          private static readonly EqualityComparer<TLink> equalityComparer =
                                                                                                           var left = cursor:
             EqualityComparer<TLink>.Default;
                                                                                                           var right = appendant:
                                                                                              40
                                                                                                           while (! equalityComparer.Equals(cursor = stack.Pop(), Links.Constants.Null))
          private readonly ILinks<TLink> links;
                                                                                              41
          private readonly TLink sequenceMarkerLink;
                                                                                              42
11
                                                                                                              right = Links.GetOrCreate(left, right);
12
          public MarkedSequenceCreteriaMatcher(ILinks<TLink> links, TLink
                                                                                                              left = cursor:
13
                                                                                              44
              sequenceMarkerLink)
                                                                                              45
                                                                                                           return Links.GetOrCreate(left, right);
                                                                                              46
15
              links = links;
                                                                                              47
              sequenceMarkerLink = sequenceMarkerLink;
                                                                                              48
                                                                                              49
          public bool IsMatched(TLink sequenceCandidate)
             => equalityComparer.Equals( links.GetSource(sequenceCandidate),
20
                                                                                              ./Sequences/DuplicateSegmentsCounter.cs
                 sequenceMarkerLink)
                                                                                                  using System.Collections.Generic;
             !! equalityComparer.Equals( links.SearchOrDefault( sequenceMarkerLink,
                                                                                                  using System.Ling;
                sequenceCandidate), links.Constants.Null);
                                                                                                  using Platform Interfaces;
23
                                                                                                  namespace Platform.Data.Doublets.Sequences
                                                                                                     public class DuplicateSegmentsCounter<TLink>: ICounter<int>
./Sequences/DefaultSequenceAppender.cs
    using System Collections Generic:
                                                                                                        private readonly IProvider<IList<KeyValuePair<IList<TLink>, IList<TLink>>>
    using Platform. Collections. Stacks:
                                                                                                              duplicateFragmentsProvider:
    using Platform.Data.Doublets.Sequences.HeightProviders;
                                                                                                        public DuplicateSegmentsCounter(IProvider<IList<KeyValuePair<IList<TLink>,
    using Platform.Data.Sequences;
                                                                                                            IList<TLink>>>> duplicateFragmentsProvider) =>
    namespace Platform.Data.Doublets.Sequences
                                                                                                             duplicateFragmentsProvider = duplicateFragmentsProvider;
```

```
public int Count() => duplicateFragmentsProvider.Get().Sum(x => x.Value.Count);
12
                                                                                                48
13
                                                                                                49
./Sequences/DuplicateSegmentsProvider.cs
                                                                                                50
    using System:
                                                                                                51
    using System.Ling;
                                                                                                52
    using System. Collections. Generic:
                                                                                                53
    using Platform.Interfaces:
                                                                                                54
    using Platform Collections:
                                                                                                55
    using Platform.Collections.Lists:
                                                                                                56
    using Platform. Collections. Segments;
                                                                                                57
    using Platform.Collections.Segments.Walkers;
    using Platform. Helpers:
                                                                                                58
    using Platform, Helpers, Singletons:
                                                                                                59
    using Platform. Numbers;
    using Platform Data Sequences;
                                                                                                61
                                                                                                62
    namespace Platform.Data.Doublets.Sequences
14
                                                                                                63
15
       public class DuplicateSegmentsProvider<TLink>:
                                                                                                65
           DictionaryBasedDuplicateSegmentsWalkerBase<TLink>.
           IProvider < IList < Key Value Pair < IList < TLink >, IList < TLink >>>>
                                                                                                67
17
          private readonly ILinks<TLink> links:
18
          private readonly ISequences < TLink > sequences:
19
                                                                                                70
          private HashSet<KeyValuePair<IList<TLink>, IList<TLink>>> groups;
20
          private BitString visited;
                                                                                                72
          private class ItemEquilityComparer:
                                                                                                73
23
          → IEqualityComparer<KeyValuePair<IList<TLink>, IList<TLink>>>
                                                                                                74
                                                                                                75
             private readonly IListEqualityComparer<TLink> listComparer;
                                                                                                76
                                                                                                77
             public ItemEquilityComparer() => listComparer =
             → Default < IListEquality Comparer < TLink >> Instance;
                                                                                                79
             public bool Equals(KeyValuePair<IList<TLink>, IList<TLink>> left,
                 KeyValuePair<IList<TLink>, IList<TLink>> right) =>
                                                                                                81
                   listComparer.Equals(left.Key, right.Key) &&
                                                                                                82
                   listComparer.Equals(left.Value, right.Value);
                                                                                                83
             public int GetHashCode(KeyValuePair<IList<TLink>, IList<TLink>> pair) =>
                                                                                                84
                 HashHelpers.Generate( listComparer.GetHashCode(pair.Key),
                                                                                                85
                  listComparer.GetHashCode(pair.Value)):
                                                                                                87
29
30
          private class ItemComparer: IComparer<KeyValuePair<IList<TLink>,
31
              IList<TLink>>>
                                                                                                89
32
                                                                                                90
             private readonly IListComparer<TLink> listComparer;
33
                                                                                                91
                                                                                                92
             public ItemComparer() => listComparer =
35
                                                                                                93
             → Default < IListComparer < TLink >> .Instance;
                                                                                                94
             public int Compare(KevValuePair<IList<TLink>, IList<TLink>> left,
                                                                                                95
                 KeyValuePair<IList<TLink>, IList<TLink>> right)
                                                                                                96
                                                                                                97
                var intermediateResult = listComparer.Compare(left.Key, right.Key);
                                                                                                98
                if (intermediateResult == \overline{0})
                                                                                               100
                   intermediateResult = listComparer.Compare(left.Value, right.Value);
                                                                                               101
                                                                                               102
                return intermediateResult;
                                                                                               103
                                                                                               104
```

```
public DuplicateSegmentsProvider(ILinks<TLink> links, ISequences<TLink>

→ sequences)

         : base(minimumStringSegmentLength: 2)
          links = links:
         sequences = sequences;
      public IList<KevValuePair<IList<TLink>, IList<TLink>>> Get()
         groups = new HashSet < KeyValuePair < IList < TLink > .
         → IList < TLink >>> (Default < Item Equility Comparer > .Instance);
         var count = links.Count():
          visited = \overline{\text{new}} BitString((long)(Integer<TLink>)count + 1):
          \frac{1}{1} links. Each (link =>
           var linkIndex = links.GetIndex(link);
            var linkBitIndex = (long)(Integer < TLink >) linkIndex;
            if (! visited.Get(linkBitIndex))
              var sequenceElements = new List < TLink > ():
                sequences. EachPart(sequenceElements. AddAndReturnTrue. linkIndex);
              \overline{\text{if}} (sequence Elements. Count > 2)
                 WalkAll(sequenceElements);
            return links.Constants.Continue;
         var resultList = groups.ToList();
        var comparer = \overline{Default} < ItemComparer > .Instance:
        resultList.Sort(comparer):
#if DEBUG
         foreach (var item in resultList)
            Print Duplicates (item);
#endif
         return resultList;
     protected override Segment < TLink > Create Segment (IList < TLink > elements, int
      → offset, int length) => new Segment<TLink>(elements, offset, length);
      protected override void OnDublicateFound(Segment < TLink > segment)
         var duplicates = CollectDuplicatesForSegment(segment);
         if (duplicates.Count > 1)
            groups.Add(new KeyValuePair<IList<TLink>,
            → IList<TLink>>(segment.ToArray(), duplicates));
      private List<TLink> CollectDuplicatesForSegment(Segment<TLink> segment)
        var duplicates = new List < TLink > ():
        var readAsElement = new HashSet < TLink > ();
          sequences.Each(sequence =>
           duplicates.Add(sequence);
```

```
readAsElement.Add(sequence);
                                                                                                        public class FrequenciesCacheBasedLinkFrequencyIncrementer<TLink>:
                return true; // Continue
106
                segment):
                                                                                                           IIncrementer<IList<TLink>>
107
                (duplicates.Any(x => visited.Get((Integer<TLink>)x)))
108
                                                                                                          private readonly LinkFrequenciesCache<TLink> cache;
109
                return new List<TLink>();
110
                                                                                                          public FrequenciesCacheBasedLinkFrequencyIncrementer(LinkFrequenciesCache<TL_1
                                                                                                1.0
111
              foreach (var duplicate in duplicates)
                                                                                                              ink > cache =  cache = 
112
                                                                                                               cache:
113
                var duplicateBitIndex = (long)(Integer<TLink>)duplicate;
114
                 visited.Set(duplicateBitIndex);
                                                                                                           /// <remarks>Sequence itseft is not changed, only frequency of its doublets is
                                                                                                12
115
                                                                                                           → incremented.</remarks>
116
                  sequences is Sequences sequences Experiments)
                                                                                                           public IList<TLink> Increment(IList<TLink> sequence)
117
                                                                                                13
118
                                                                                                14
                var partiallyMatched = sequencesExperiments.GetAllPartiallyMatchingSequenc
                                                                                                               cache.IncrementFrequencies(sequence);
119
                                                                                                16
                                                                                                             return sequence;
                     es4((HashSet<ulong>)(object)readAsElement,
                                                                                                17
                     (IList<ulong>)segment):
                                                                                                18
                foreach (var partially Matched Sequence in partially Matched)
120
                                                                                                19
121
                    TLink sequenceIndex = (Integer < TLink >) partially Matched Sequence;
                    duplicates. Add(sequenceIndex):
                                                                                                ./Sequences/Frequencies/Cache/FrequenciesCacheBasedLinkToItsFrequencyNumberConver
124
                                                                                                    using Platform. Interfaces;
125
              duplicates.Sort():
126
                                                                                                     namespace Platform.Data.Doublets.Sequences.Frequencies.Cache
             return duplicates:
127
128
                                                                                                        public class FrequenciesCacheBasedLinkToItsFrequencyNumberConverter<TLink>:
129
                                                                                                           IConverter<Doublet<TLink>, TLink>
           private void PrintDuplicates(KeyValuePair<IList<TLink>, IList<TLink>>
130
               duplicatesItem)
                                                                                                          private readonly LinkFrequenciesCache<TLink> cache;
131
                                                                                                          public FrequenciesCacheBasedLinkToItsFrequencyNumberConverter(LinkFrequencies
              if (!(_links is ILinks<ulong> ulongLinks))
132
                                                                                                               Cache < TLink > cache = 
133
                return:
134
                                                                                                          public TLink Convert(Doublet<TLink> source) => cache.GetFrequency(ref
135
             var duplicatesKey = duplicatesItem.Key;

→ source).Frequency:

136
             var keyString = UnicodeMap.FromLinksToString((IList<ulong>)duplicatesKey);
                                                                                                10
137
             Console WriteLine($\mathbb{S}\) \{\key\String\} (\{\string.Join(\hat{n}, \hat{n}, \duplicatesKey)\})\);
138
                                                                                                11
             var duplicatesList = duplicatesItem. Value;
139
             for (int i = 0: i < duplicatesList.Count: i++)
140
                                                                                                ./Sequences/Frequencies/Cache/LinkFrequenciesCache.cs
141
                                                                                                    using System;
                ulong sequenceIndex = (Integer<TLink>)duplicatesList[i]:
142
                                                                                                    using System.Collections.Generic;
                var formatedSequenceStructure = ulongLinks.FormatStructure(sequenceIndex, x
143
                                                                                                     using System.Runtime.CompilerServices:
                     => Point < ulong > IsPartialPoint(x), (sb, link) => =
                                                                                                    using Platform. Interfaces;
                     UnicodeMap.IsCharLink(link.Index)?
                                                                                                     using Platform. Numbers;
                     sb.Append(UnicodeMap.FromLinkToChar(link.Index)):
                     sb.Append(link.Index));
                                                                                                     namespace Platform. Data. Doublets. Sequences. Frequencies. Cache
                Console.WriteLine(formatedSequenceStructure);
144
                var sequenceString = UnicodeMap.FromSequenceLinkToString(sequenceIndex,
                                                                                                        /// <remarks>
145
                                                                                                        /// Can be used to operate with many CompressingConverters (to keep global frequencies
                     ulongLinks):
                Console. WriteLine(sequenceString);
                                                                                                        → data between them).
146
                                                                                                           TODO: Extract interface to implement frequencies storage inside Links storage
147
                                                                                                11
              Console.WriteLine();
                                                                                                          / < / \text{remarks} >
148
                                                                                                12
                                                                                                        public class LinkFrequenciesCache<TLink>: LinksOperatorBase<TLink>
149
                                                                                                13
150
                                                                                                14
                                                                                                          private static readonly EqualityComparer<TLink> equalityComparer =
151
                                                                                                15
                                                                                                           → EqualityComparer<TLink>.Default;
                                                                                                          private static readonly Comparer<TLink> comparer = Comparer<TLink>.Default;
                                                                                                16
./Sequences/Frequencies/Cache/FrequenciesCacheBasedLinkFrequencyIncrementer.cs
                                                                                                17
                                                                                                          private readonly Dictionary < Doublet < TLink >, LinkFrequency < TLink >>
     using System.Collections.Generic;
                                                                                                18
     using Platform.Interfaces;
                                                                                                                 doubletsCache:
                                                                                                          private readonly ICounter<TLink, TLink> frequencyCounter;
                                                                                                19
     namespace Platform.Data.Doublets.Sequences.Frequencies.Cache
                                                                                                20
```

```
public LinkFrequenciesCache(ILinks<TLink> links, ICounter<TLink, TLink>
                                                                                                                  if (! equalityComparer.Equals(link, default))
21
              frequencyCounter)
             : base(links)
                                                                                                                    data.Frequency = ArithmeticHelpers.Add(data.Frequency,
                                                                                                                         frequencyCounter.Count(link));
23
               doubletsCache = new Dictionary < Doublet < TLink >,
                                                                                                                   doubletsCache.Add(doublet, data);
                 LinkFrequency<TLink>>(4096, DoubletComparer<TLink>,Default):
                                                                                                 86
25
              frequencyCounter = frequencyCounter;
                                                                                                               return data:
                                                                                                 88
27
                                                                                                 89
          [MethodImpl(MethodImplOptions.AggressiveInlining)]
                                                                                                 90
          public LinkFrequency (TLink > GetFrequency (TLink source, TLink target)
                                                                                                            public void ValidateFrequencies()
29
                                                                                                 91
                                                                                                 92
             var doublet = new Doublet < TLink > (source, target);
                                                                                                               foreach (var entry in doubletsCache)
                                                                                                 93
             return GetFrequency(ref doublet):
                                                                                                 94
32
                                                                                                                  var value = entry.Value:
33
                                                                                                 95
                                                                                                                  var linkIndex = value.Link;
                                                                                                 96
34
          [MethodImpl(MethodImplOptions, AggressiveInlining)]
                                                                                                                  if (! equalityComparer.Equals(linkIndex, default))
3.5
                                                                                                 97
          public LinkFrequency (TLink > GetFrequency (ref Doublet < TLink > doublet)
                                                                                                 98
                                                                                                                    var frequency = value. Frequency;
                                                                                                                    var count = frequencyCounter.Count(linkIndex);
               doubletsCache.TryGetValue(doublet, out LinkFrequency<TLink> data);
                                                                                                 100
                                                                                                                     // TODO: Why 'frequency' always greater than 'count' by 1?
39
             return data;
                                                                                                 101
                                                                                                                    if ((( comparer.Compare(frequency, count) > 0) &&
                                                                                                 102
                                                                                                                          comparer.Compare(ArithmeticHelpers.Subtract(frequency, count),
          public void IncrementFrequencies(IList<TLink> sequence)
42
                                                                                                                         Integer < TLink > .One) > 0)
43
                                                                                                                          comparer.Compare(count, frequency) > 0) &&
                                                                                                 103
             for (var i = 1; i < \text{sequence.Count}; i++)
44
                                                                                                                            comparer.Compare(ArithmeticHelpers.Subtract(count, frequency),
                                                                                                                          Integer < TLink > One > 0)))
                IncrementFrequency(sequence[i - 1], sequence[i]);
                                                                                                 104
                                                                                                                       throw new InvalidOperationException("Frequencies validation failed.");
                                                                                                 105
                                                                                                 106
          [MethodImpl(MethodImplOptions.AggressiveInlining)]
                                                                                                 107
50
                                                                                                108
          public LinkFrequency (TLink > IncrementFrequency (TLink source, TLink target)
                                                                                                 109
                                                                                                                       if (value. Frequency > 0)
                                                                                                110
             var doublet = new Doublet < TLink > (source, target);
             return IncrementFrequency(ref doublet):
                                                                                                 111
54
                                                                                                                          var frequency = value. Frequency:
                                                                                                112
                                                                                                                          linkIndex = createLink(entry.Key.Source, entry.Key.Target);
                                                                                                 113
                                                                                                                          var count = countLinkFrequency(linkIndex);
          public void PrintFrequencies(IList<TLink> sequence)
                                                                                                 114
                                                                                                 115
                                                                                                                          if ((frequency > count && frequency - count > 1) || (count > frequency
             for (var i = 1; i < sequence.Count; i++)
                                                                                                                      && count - frequency > 1)
                                                                                                                            throw new Exception("Frequencies validation failed.");
                PrintFrequency(sequence[i - 1], sequence[i]);
                                                                                                117
                                                                                                 119
63
                                                                                                 120
64
          public void PrintFrequency(TLink source, TLink target)
                                                                                                 121
                                                                                                 122
             var number = GetFrequency(source, target). Frequency;
                                                                                                 123
             Console.WriteLine("(\{0\},\{1\}) - \{2\}", source, target, number);
                                                                                                 ./Sequences/Frequencies/Cache/LinkFrequency.cs
          [MethodImpl(MethodImplOptions.AggressiveInlining)]
                                                                                                      using System.Runtime.CompilerServices;
          public LinkFrequency<TLink> IncrementFrequency(ref Doublet<TLink> doublet)
72
                                                                                                     using Platform.Numbers;
               (doubletsCache.TryGetValue(doublet, out LinkFrequency<TLink> data))
74
                                                                                                      namespace Platform.Data.Doublets.Sequences.Frequencies.Cache
                data.IncrementFrequency();
76
                                                                                                         public class LinkFrequency<TLink>
             else
                                                                                                            public TLink Frequency { get; set; }
                                                                                                           public TLink Link { get; set; }
                var link = Links.SearchOrDefault(doublet.Source, doublet.Target);
                data = new LinkFrequency<TLink>(Integer<TLink>.One, link);
                                                                                                            public LinkFrequency (TLink frequency, TLink link)
81
```

3.0

31

37

41

48

49

5.1

5.7

6.1

65

68

71

```
protected TLink total;
                                                                                             16
             Frequency = frequency:
                                                                                             17
                                                                                                       public SequenceSymbolFrequencyOneOffCounter(ILinks<TLink> links, TLink
            Link = link;
                                                                                                           sequenceLink, TLink symbol)
                                                                                             19
          public LinkFrequency() { }
17
                                                                                                           links = links:
                                                                                             20
                                                                                                           sequenceLink = sequenceLink:
                                                                                             21
          [MethodImpl(MethodImplOptions.AggressiveInlining)]
19
                                                                                                           symbol = symbol;
                                                                                             22
          public void IncrementFrequency() => Frequency =
                                                                                                           total = default;
                                                                                             23
          → ArithmeticHelpers<TLink>.Increment(Frequency);
                                                                                             24
21
          [MethodImpl(MethodImplOptions.AggressiveInlining)]
                                                                                                       public virtual TLink Count()
22
                                                                                             26
          public void DecrementFrequency() => Frequency =
23
                                                                                             27
          → ArithmeticHelpers<TLink>.Decrement(Frequency):
                                                                                                          if ( comparer.Compare( total, default) > 0)
                                                                                             28
                                                                                             29
          public override string ToString() => \mathbb{S}^{\parallel}F: \{Frequency\}, L: \{Link\}^{\parallel};
                                                                                                             return total;
25
                                                                                             30
26
                                                                                             31
                                                                                                          StopableSequenceWalker.WalkRight(sequenceLink, links.GetSource,
                                                                                             39
27
                                                                                                               links.GetTarget, IsElement, VisitElement):
                                                                                                          return total;
./Sequences/Frequencies/Counters/MarkedSequenceSymbolFrequencyOneOffCounter.cs 33
    using Platform. Interfaces:
                                                                                             35
                                                                                                       private bool IsElement(TLink x) => equalityComparer.Equals(x, symbol)
                                                                                             36
    namespace Platform.Data.Doublets.Sequences.Frequencies.Counters
                                                                                                             links.IsPartialPoint(x); // TODO: Use SequenceElementCreteriaMatcher
       public class MarkedSequenceSymbolFrequencyOneOffCounter<TLink>:
                                                                                                           instead of IsPartialPoint
                                                                                             37
           SequenceSymbolFrequencyOneOffCounter<TLink>
                                                                                                       private bool VisitElement(TLink element)
                                                                                             38
                                                                                             39
          private readonly ICreteriaMatcher<TLink> markedSequenceMatcher;
                                                                                                          if (equalityComparer.Equals(element, symbol))
                                                                                             40
          public MarkedSequenceSymbolFrequencyOneOffCounter(ILinks<TLink> links,
                                                                                             41
                                                                                                              total = ArithmeticHelpers.Increment(total);
              ICreteriaMatcher<TLink> markedSequenceMatcher, TLink sequenceLink, TLink
                                                                                             42
                                                                                             43
                                                                                                          return true;
                                                                                             44
            : base(links, sequenceLink, symbol)
                                                                                             45
            => markedSequenceMatcher = markedSequenceMatcher;
                                                                                             46
12
                                                                                             47
          public override TLink Count()
13
              (! markedSequenceMatcher.IsMatched( sequenceLink))
                                                                                             ./Sequences/Frequencies/Counters/TotalMarkedSequenceSymbolFrequencyCounter.cs
               return default;
                                                                                                 using Platform.Interfaces;
                                                                                                 namespace Platform.Data.Doublets.Sequences.Frequencies.Counters
             return base.Count();
                                                                                                    public class TotalMarkedSequenceSymbolFrequencyCounter<TLink>: ICounter<TLink,
^{21}
                                                                                                        TLink>
22
                                                                                                       private readonly ILinks<TLink> links:
./Sequences/Frequencies/Counters/SequenceSymbolFrequencyOneOffCounter.cs
                                                                                                       private readonly ICreteriaMatcher < TLink > markedSequenceMatcher;
    using System.Collections.Generic:
    using Platform. Interfaces:
                                                                                                       public TotalMarkedSequenceSymbolFrequencyCounter(ILinks<TLink> links,
                                                                                             10
    using Platform. Numbers:
                                                                                                           ICreteriaMatcher<TLink> markedSequenceMatcher)
    using Platform. Data. Sequences;
                                                                                             1.1
                                                                                                            links = links;
                                                                                             12
    namespace Platform.Data.Doublets.Sequences.Frequencies.Counters
                                                                                                           markedSequenceMatcher = markedSequenceMatcher;
                                                                                             13
       public class SequenceSymbolFrequencyOneOffCounter<TLink>: ICounter<TLink>
                                                                                             14
                                                                                             15
                                                                                                       public TLink Count(TLink argument) => new
                                                                                             16
          private static readonly EqualityComparer<TLink> equalityComparer =
                                                                                                           TotalMarkedSequenceSymbolFrequencyOneOffCounter<TLink>( links,
          → EqualityComparer<TLink>.Default;
                                                                                                            markedSequenceMatcher, argument).Count():
          private static readonly Comparer<TLink> comparer = Comparer<TLink> Default;
                                                                                             17
12
          protected readonly ILinks<TLink> links;
13
                                                                                             18
          protected readonly TLink sequenceLink;
          protected readonly TLink symbol;
```

```
./Sequences/Frequencies/Counters/TotalMarkedSequenceSymbolFrequencyOneOffCounter.cs
                                                                                                          links = links;
                                                                                                          symbol = symbol:
    using Platform.Interfaces:
                                                                                                          visits = new HashSet < TLink > ():
    using Platform. Numbers;
                                                                                            21
                                                                                                          total = default
                                                                                            22
    namespace Platform.Data.Doublets.Sequences.Frequencies.Counters
                                                                                            24
       public class TotalMarkedSequenceSymbolFrequencyOneOffCounter<TLink>:
                                                                                                      public TLink Count()
                                                                                            25
           TotalSequenceSymbolFrequencyOneOffCounter<TLink>
                                                                                            26
                                                                                                         if (comparer.Compare(total, default) > 0 || visits.Count > 0)
                                                                                            27
          private readonly ICreteriaMatcher<TLink> markedSequenceMatcher;
                                                                                            28
                                                                                                            return total;
                                                                                            29
          public TotalMarkedSequenceSymbolFrequencyOneOffCounter(ILinks<TLink> links,
                                                                                            30
             ICreteriaMatcher<TLink> markedSequenceMatcher, TLink symbol): base(links,
                                                                                                         CountCore( symbol);
                                                                                            31
                                                                                                         return total;
                                                                                            32
            => markedSequenceMatcher = markedSequenceMatcher;
                                                                                            33
                                                                                            34
                                                                                                      private void CountCore(TLink link)
          protected override void CountSequenceSymbolFrequency(TLink link)
13
                                                                                            35
                                                                                            36
                                                                                                         var anv = links.Constants.Anv
            var symbolFrequencyCounter = new
                                                                                            37
15
                                                                                                         if (equality Comparer. Equals(links. Count (any, link), default))
                 MarkedSequenceSymbolFrequencyOneOffCounter<TLink>( links,
                                                                                            38
                                                                                            39
                  markedSequenceMatcher, link, symbol):
                                                                                                            CountSequenceSymbolFrequency(link):
                                                                                            40
              total = ArithmeticHelpers.Add( total, symbolFrequencyCounter.Count());
                                                                                            41
17
                                                                                            42
                                                                                            43
19
                                                                                                             links.Each(EachElementHandler, any, link);
                                                                                            44
                                                                                            45
./Sequences/Frequencies/Counters/TotalSequenceSymbolFrequencyCounter.cs
                                                                                            46
    using Platform. Interfaces:
                                                                                            47
                                                                                                      protected virtual void CountSequenceSymbolFrequency(TLink link)
                                                                                            48
    namespace Platform.Data.Doublets.Sequences.Frequencies.Counters
                                                                                            49
                                                                                                         var symbolFrequencyCounter = new
       public class TotalSequenceSymbolFrequencyCounter<TLink>: ICounter<TLink, TLink>
                                                                                                         SequenceSymbolFrequencyOneOffCounter<TLink>( links, link, symbol);
                                                                                                          total = ArithmeticHelpers.Add( total, symbolFrequencyCounter.Count());
                                                                                            51
          private readonly ILinks<TLink> links:
                                                                                            52
          public TotalSequenceSymbolFrequencyCounter(ILinks<TLink> links) => links =
                                                                                            53
             links:
                                                                                                      private TLink EachElementHandler(IList<TLink> doublet)
                                                                                            54
          public TLink Count(TLink symbol) => new
                                                                                            55
              TotalSequenceSymbolFrequencyOneOffCounter<TLink>( links,
                                                                                                         var constants = links.Constants;
                                                                                            56
              symbol).Count():
                                                                                                         var doubletIndex = doublet[constants.IndexPart];
                                                                                            57
                                                                                                         if ( visits.Add(doubletIndex))
                                                                                            58
11
                                                                                            59
                                                                                                            CountCore(doubletIndex);
./Sequences/Frequencies/Counters/TotalSequenceSymbolFrequencyOneOffCounter.cs
    using System.Collections.Generic:
                                                                                                         return constants. Continue;
                                                                                            62
    using Platform. Interfaces:
                                                                                            63
    using Platform. Numbers;
                                                                                            64
                                                                                            65
    namespace Platform.Data.Doublets.Sequences.Frequencies.Counters
       public class TotalSequenceSymbolFrequencyOneOffCounter<TLink>: ICounter<TLink>
                                                                                            ./Sequences/HeightProviders/CachedSequenceHeightProvider.cs
          private static readonly EqualityComparer<TLink> equalityComparer =
                                                                                                using System. Collections. Generic:
                                                                                                using Platform. Interfaces;
          → EqualityComparer<TLink>.Default;
          private static readonly Comparer<TLink> comparer = Comparer<TLink> Default;
                                                                                                namespace Platform.Data.Doublets.Sequences.HeightProviders
1.1
          protected readonly ILinks<TLink> links;
12
          protected readonly TLink symbol:
                                                                                                   public class CachedSequenceHeightProvider<TLink>: LinksOperatorBase<TLink>,
          protected readonly HashSet<TLink> visits;
                                                                                                       ISequenceHeightProvider<TLink>
          protected TLink total;
                                                                                                      private static readonly EqualityComparer<TLink> equalityComparer =
          public TotalSequenceSymbolFrequencyOneOffCounter(ILinks<TLink> links, TLink
                                                                                                      → EqualityComparer<TLink>.Default;
             symbol
                                                                                                      private readonly TLink heightPropertyMarker;
                                                                                            10
```

```
private readonly ISequenceHeightProvider<TLink> baseHeightProvider;
          private readonly IConverter < TLink > addressToUnaryNumberConverter:
                                                                                                                 return height:
12
          private readonly IConverter < TLink > unary Number To Address Converter;
                                                                                                   22
          private readonly IPropertyOperator<TLink, TLink, TLink> propertyOperator;
14
                                                                                                   23
                                                                                                   24
          public CachedSequenceHeightProvider(
             ILinks<TLink> links,
                                                                                                   ./Sequences/HeightProviders/ISequenceHeightProvider.cs
             ISequenceHeightProvider<TLink> baseHeightProvider,
             IConverter < TLink > addressToUnaryNumberConverter.
                                                                                                       using Platform.Interfaces:
19
             IConverter<TLink> unaryNumberToAddressConverter,
             TLink heightPropertyMarker.
                                                                                                       namespace Platform.Data.Doublets.Sequences.HeightProviders
21
             IProperty Operator < TLink, TLink, TLink > property Operator)
                                                                                                           public interface ISequenceHeightProvider<TLink>: IProvider<TLink, TLink>
             : base(links)
23
24
               heightPropertyMarker = heightPropertyMarker;
               baseHeightProvider = baseHeightProvider:
26
               \bar{a} address \bar{T} o \bar{U} nary \bar{N} umber \bar{C} on \bar{V} or \bar{V} and \bar{V} is \bar{U} nary \bar{N} umber \bar{C} on \bar{V} or \bar{V}
27
               \overline{\phantom{a}}unaryNumber\overline{\phantom{a}}oAddress\overline{\phantom{a}}onverter\overline{\phantom{a}}unaryNumber\overline{\phantom{a}}oAddress\overline{\phantom{a}}onverter\overline{\phantom{a}}
              propertyOperator = propertyOperator;
                                                                                                   ./Sequences/Sequences.cs
                                                                                                       using System:
30
                                                                                                       using System.Collections.Generic;
31
           public TLink Get(TLink sequence)
                                                                                                       using System.Ling:
32
                                                                                                       using System.Runtime.CompilerServices;
33
                                                                                                       using Platform.Collections;
             TLink height:
            var height Value = propertyOperator.GetValue(sequence, heightPropertyMarker);
                                                                                                       using Platform.Collections.Lists:
             if (equalityComparer.Equals(heightValue, default))
                                                                                                       using Platform. Threading. Synchronization;
                                                                                                       using Platform. Helpers. Singletons;
37
                                                                                                       using LinkIndex = System.UInt64;
                height = baseHeightProvider.Get(sequence);
                                                                                                       using Platform. Data. Constants:
                heightValue = addressToUnaryNumberConverter.Convert(height);
                                                                                                       using Platform.Data.Sequences;
                 propertyOperator.SetValue(sequence, heightPropertyMarker, heightValue);
                                                                                                   11
                                                                                                       using Platform. Data. Doublets. Sequences. Walkers;
                                                                                                   13
                                                                                                       namespace Platform. Data. Doublets. Sequences
                                                                                                   14
                                                                                                   15
                height = unaryNumberToAddressConverter.Convert(heightValue);
                                                                                                   16
                                                                                                              Представляет коллекцию последовательностей связей.
                                                                                                   17
             return height:
                                                                                                   18
                                                                                                              </summary>
47
                                                                                                              <remarks>
                                                                                                   19
                                                                                                              Обязательно реализовать атомарность каждого публичного метода.
                                                                                                   20
                                                                                                   21
                                                                                                              TODO:
                                                                                                   22
./Sequences/HeightProviders/DefaultSequenceRightHeightProvider.cs
                                                                                                   23
                                                                                                              !!! Повышение вероятности повторного использования групп
    using Platform.Interfaces;
                                                                                                   24
    using Platform. Numbers:
                                                                                                               (подпоследовательностей),
                                                                                                              через естественную группировку по unicode типам, все whitespace вместе, все
                                                                                                   25
    namespace Platform.Data.Doublets.Sequences.HeightProviders
                                                                                                              символы вместе, все числа вместе и т.п.
                                                                                                          /// + использовать ровно сбалансированный вариант, чтобы уменьшать вложенность
                                                                                                   26
       public class DefaultSequenceRightHeightProvider<TLink>:
                                                                                                               (глубину графа)
           LinksOperatorBase<TLink>, ISequenceHeightProvider<TLink>
                                                                                                   27
                                                                                                           ^{\prime\prime}/ х^*у - найти все связи между, в последовательностях любой формы, если не стоит
                                                                                                   28
          private readonly ICreteriaMatcher<TLink> elementMatcher;
                                                                                                              ограничитель на то, что является последовательностью, а что нет,
                                                                                                           /// то находятся любые структуры связей, которые содержат эти элементы именно в
                                                                                                   29
          public DefaultSequenceRightHeightProvider(ILinks<TLink> links,
                                                                                                               таком порядке.
               ICreteriaMatcher<TLink> elementMatcher) : base(links) => elementMatcher
                                                                                                   30
                                                                                                              Рост последовательности слева и справа.
              = elementMatcher:
                                                                                                   31
                                                                                                              Поиск со звёздочкой.
                                                                                                   32
                                                                                                              URL, PURL - реестр используемых во вне ссылок на ресурсы,
          public TLink Get(TLink sequence)
                                                                                                   33
12
                                                                                                              так же проблема может быть решена при реализации дистанционных триггеров.
                                                                                                   34
13
             var height = default(TLink);
                                                                                                              Нужны ли уникальные указатели вообще?
                                                                                                   35
             var pairOrElement = sequence;
                                                                                                              Что если обращение к информации будет происходить через содержимое всегда?
                                                                                                   36
             while (! element Matcher. IsMatched (pair Or Element))
                                                                                                   37
                                                                                                   38
                                                                                                              Писать тесты.
                pairOrElement = Links.GetTarget(pairOrElement);
                                                                                                   39
                height = ArithmeticHelpers.Increment(height):
```

```
Можно убрать зависимость от конкретной реализации Links.
                                                                                                        if (Options.UseSequenceMarker)
                                                                                         100
   на зависимость от абстрактного элемента, который может быть представлен
                                                                                         101
    несколькими способами.
                                                                                         102
                                                                                         103
   Можно ли как-то сделать один общий интерфейс
                                                                                         104
                                                                                         105
                                                                                         106
    Блокчейн и/или гит для распределённой записи транзакций.
                                                                                         107
                                                                                         108
    </remarks>
                                                                                         109
public partial class Sequences: ISequences
    | // IList<string>, IList<ulong</li>

                                                                                         110
                                                                                         111
    (после завершения реализации Sequences)
                                                                                                        return sequence;
                                                                                         112
                                                                                         113
   private static readonly LinksCombinedConstants<br/>
bool, ulong, long> constants =
                                                                                         114
       Default<LinksCombinedConstants<br/>bool, ulong, long>>.Instance;
                                                                                                    #region Count
                                                                                         115
                                                                                         116
       <summary>Возвращает значение ulong, обозначающее любое количество
                                                                                         117
       связей.</summary
                                                                                         118
   public const ulong ZeroOrMany = ulong.MaxValue;
                                                                                         119
                                                                                         120
   public SequencesOptionsulongOptions;
                                                                                         121
   public readonly SynchronizedLinks<ulong> Links:
   public readonly IŠynchronization Sync;
                                                                                         122
   public Sequences(SynchronizedLinks<ulong> links)
                                                                                         123
     : this(links, new SequencesOptions<ulong>())
                                                                                         124
                                                                                         125
                                                                                         126
                                                                                                             return 0:
                                                                                         127
  public Sequences(SynchronizedLinks<ulong> links, SequencesOptions<ulong> options)
                                                                                         128
                                                                                         129
      Links = links:
                                                                                         130
      Svnc = links.SvncRoot:
                                                                                         131
      Options = options;
                                                                                         132
                                                                                         133
     Options. ValidateOptions():
                                                                                         134
     Options.InitOptions(Links);
                                                                                         135
                                                                                         136
   public bool IsSequence (ulong sequence)
                                                                                         137
                                                                                         138
     return Sync. Execute ReadOperation(() =>
                                                                                         139
                                                                                         140
        if (Options. UseSequenceMarker)
                                                                                         141
                                                                                         142
            return Options.MarkedSequenceMatcher.IsMatched(sequence);
                                                                                         143
                                                                                         144
        return !Links.Unsync.IsPartialPoint(sequence);
                                                                                         145
                                                                                                          return 0;
                                                                                         146
                                                                                         147
                                                                                         148
   [MethodImpl(MethodImplOptions.AggressiveInlining)]
                                                                                         149
   private ulong GetSequenceByElements(ulong sequence)
                                                                                         150
                                                                                         151
       (Options. UseSequenceMarker)
                                                                                                             return 0:
                                                                                         152
                                                                                         153
        return Links.SearchOrDefault(Options.SequenceMarkerLink, sequence);
                                                                                         154
                                                                                         155
      return sequence;
                                                                                         156
                                                                                         157
                                                                                         158
   private ulong GetSequenceElements(ulong sequence)
```

44

45

47

51

52

53

54

5.5

56

57

59

61

62

64

67

70

71

73

75

76

77

78

92

94

95

97

```
var linkContents = new UInt64Link(Links.GetLink(sequence)):
     if (linkContents.Source == Options.SequenceMarkerLink)
        return linkContents. Target;
       (linkContents.Target == Options.SequenceMarkerLink)
        return linkContents.Source;
public ulong Count(params ulong | sequence)
  if (sequence.Length == 0)
     return Links.Count( constants.Any, Options.SequenceMarkerLink,
         constants.Any);
     (sequence.Length == 1) // Первая связь это адрес
      if (sequence[0] == constants.Null)
       (\text{sequence}[0] == \text{constants.Anv})
        return Count():
       (Options. UseSequenceMarker)
        return Links.Count(constants.Any, Options.SequenceMarkerLink,
            sequence[0]);
     return Links.Exists(sequence[0]) ? 1UL : 0;
   throw new NotImplementedException();
private ulong CountReferences(params ulong[] restrictions)
  if (restrictions. Length == 0)
    (restrictions.Length == 1) // Первая связь это адрес
      if (restrictions[0] == constants.Null)
        (Options. UseSequenceMarker)
        var elementsLink = GetSequenceElements(restrictions[0]);
        var sequenceLink = GetSequenceByElements(elementsLink);
        if (sequenceLink! = constants.Null)
```

```
return Links.Count(sequenceLink) + Links.Count(elementsLink) - 1;
                                                                                                      var results = new List < ulong > ():
                                                                                       222
                                                                                                      Each(results.AddAndReturnTrue, sequence);
                                                                                       223
        return Links.Count(elementsLink);
                                                                                       224
                                                                                                      return results:
                                                                                       225
     return Links.Count(restrictions[0]);
                                                                                       226
                                                                                                   public bool Each(Func<ulong, bool> handler, IList<ulong> sequence)
                                                                                       227
  throw new NotImplementedException();
                                                                                       228
                                                                                                      return Sync. Execute Read Operation (() = >
                                                                                       229
                                                                                       230
#endregion
                                                                                                         if (sequence.IsNullOrEmpty())
                                                                                       231
                                                                                       232
#region Create
                                                                                       233
                                                                                                            return true;
                                                                                       234
public ulong Create(params ulong | sequence)
                                                                                                         Links.EnsureEachLinkIsAnvOrExists(sequence):
                                                                                       235
                                                                                                         if (sequence.Count == 1)
                                                                                       236
  return Sync.ExecuteWriteOperation(() =>
                                                                                       237
                                                                                                            var link = sequence[0];
                                                                                       238
     if (sequence.IsNullOrEmpty())
                                                                                                            if (link == constants.Any)
                                                                                       239
                                                                                       240
        return constants.Null;
                                                                                                               return Links. Unsync. Each (constants. Any, constants. Any, handler);
                                                                                       241
                                                                                       242
      Links.EnsureEachLinkExists(sequence);
                                                                                                            return handler(link);
                                                                                       243
     return CreateCore(sequence);
                                                                                       244
                                                                                                           (sequence.Count == 2)
                                                                                       245
                                                                                                            return Links. Unsync. Each (sequence [0], sequence [1], handler);
                                                                                       247
private ulong CreateCore(params ulong | sequence)
                                                                                       248
                                                                                                         if (Options.UseIndex &&!Options.Indexer.CheckIndex(sequence))
                                                                                       249
    (Options. UseIndex)
                                                                                       250
                                                                                                            return false:
                                                                                       251
     Options.Indexer.Index(sequence);
                                                                                       252
                                                                                                         return EachCore(handler, sequence);
                                                                                       253
  var sequenceRoot = default(ulong);
                                                                                       254
  if (Options.EnforceSingleSequenceVersionOnWriteBasedOnExisting)
                                                                                       255
                                                                                       256
     var matches = Each(sequence);
                                                                                                   private bool EachCore(Func<ulong, bool> handler, IList<ulong> sequence)
                                                                                       257
     if (matches.Count > 0)
                                                                                       258
                                                                                                      var matcher = new Matcher(this, sequence, new HashSet < LinkIndex > (), handler);
                                                                                       259
        sequenceRoot = matches[0];
                                                                                                      // TODO: Find out why matcher.HandleFullMatched executed twice for the same
                                                                                       260

→ sequence Id.

                                                                                                      FuncFunc<ulorg, bool> innerHandler = Options.UseSequenceMarker? (Func<ulorg,</li>
                                                                                       261
  else if (Options.EnforceSingleSequenceVersionOnWriteBasedOnNew)
                                                                                                         bool>)matcher.HandleFullMatchedSequence: matcher.HandleFullMatched;
                                                                                                      //if (sequence.Length >= 2)
                                                                                       262
     return CompactCore(sequence);
                                                                                                      if (!StepRight(innerHandler, sequence[0], sequence[1]))
                                                                                       263
                                                                                       264
    (sequenceRoot == default)
                                                                                                         return false;
                                                                                       265
                                                                                       266
     sequenceRoot = Options.LinksToSequenceConverter.Convert(sequence);
                                                                                                      var last = sequence.Count - 2;
                                                                                       267
                                                                                                      for (var i = 1; i < last; i++)
                                                                                       268
    (Options. UseSequenceMarker)
                                                                                       269
                                                                                                         if (!PartialStepRight(innerHandler, sequence[i], sequence[i+1]))
                                                                                       270
     Links. Unsync. CreateAndUpdate(Options. SequenceMarkerLink, sequenceRoot);
                                                                                       271
                                                                                                            return false;
                                                                                       272
  return sequenceRoot: // Возвращаем корень последовательности (т.е. сами
                                                                                       273
   → элементы)
                                                                                       274
                                                                                                      if (sequence.Count >= 3)
                                                                                       275
                                                                                       276
#endregion
                                                                                                         if (!StepLeft(innerHandler, sequence[sequence.Count - 2].
                                                                                       277
                                                                                                             sequence[sequence.Count - 1]))
#region Each
                                                                                       ^{278}
                                                                                                            return false;
public List < ulong > Each (params ulong | sequence)
                                                                                       279
```

 $\frac{220}{221}$

```
338
                                                                                                  #region Update
                                                                                       339
  return true;
                                                                                       340
                                                                                                  public ulong Update(ulong[] sequence, ulong[] newSequence)
                                                                                       341
                                                                                       342
private bool PartialStepRight(Func<ulong, bool> handler, ulong left, ulong right)
                                                                                                     if (sequence.IsNullOrEmpty() && newSequence.IsNullOrEmpty())
                                                                                       3/13
                                                                                       344
  return Links. Unsync. Each (constants. Any, left, doublet =>
                                                                                                        return constants.Null;
                                                                                       345
                                                                                       346
      if (!StepRight(handler, doublet, right))
                                                                                                       (sequence.IsNullOrEmpty())
                                                                                       347
                                                                                       ^{348}
         return false;
                                                                                                        return Create(newSequence):
                                                                                       349
                                                                                       350
        (left != doublet)
                                                                                                     if (newSequence.IsNullOrEmpty())
                                                                                       351
                                                                                       352
         return PartialStepRight(handler, doublet, right);
                                                                                                        Delete(sequence);
                                                                                       353
                                                                                                        return constants.Null;
                                                                                       354
     return true;
                                                                                       355
                                                                                                     return Sync.ExecuteWriteOperation(() =>
                                                                                       356
                                                                                       357
                                                                                                        Links. Ensure Each Link Is Any Or Exists (sequence);
                                                                                       358
private bool StepRight(Func<ulong, bool> handler, ulong left, ulong right) =>
                                                                                                        Links. Ensure Each Link Exists (new Sequence):
                                                                                       359
    Links.Unsync.Each(left, constants.Any, rightStep =>
                                                                                                        return UpdateCore(sequence, newSequence);
                                                                                       360
    TryStepRightUp(handler, right, rightStep));
                                                                                       361
                                                                                       362
private bool TryStepRightUp(Func<ulong, bool> handler, ulong right, ulong
                                                                                       363
                                                                                                  private ulong UpdateCore(ulong[] sequence, ulong[] newSequence)
    stepFrom)
                                                                                       364
                                                                                       365
  var upStep = stepFrom;
                                                                                                     ulong best Variant:
                                                                                       366
  var firstSource = Links.Unsync.GetTarget(upStep);
                                                                                                     if (Options.EnforceSingleSequenceVersionOnWriteBasedOnNew &&
                                                                                       367
  while (firstSource != right && firstSource != upStep)
                                                                                                         !sequence.EqualTo(newSequence))
                                                                                       368
      upStep = firstSource:
                                                                                                        bestVariant = CompactCore(newSequence);
                                                                                       369
     firstSource = Links.Unsync.GetSource(upStep);
                                                                                       370
                                                                                                     else
                                                                                       371
    (firstSource == right)
                                                                                       372
                                                                                                        bestVariant = CreateCore(newSequence);
                                                                                       373
     return handler(stepFrom);
                                                                                       374
                                                                                                        TODO: Check all options only ones before loop execution
                                                                                       375
  return true;
                                                                                                        Возможно нужно две версии Each, возвращающий фактические
                                                                                       376
                                                                                                         последовательности и с маркером,
                                                                                                     // или возможно даже возвращать и тот и тот вариант. С другой стороны все
private bool StepLeft(Func<ulong, bool> handler, ulong left, ulong right) =>
                                                                                                         варианты можно получить имея только фактические последовательности.
    Links.Unsync.Each( constants.Any, right, leftStep => TryStepLeftUp(handler,
                                                                                                     foreach (var variant in Each(sequence))
                                                                                      378
    left, leftStep));
                                                                                       379
                                                                                                        if (variant != bestVariant)
                                                                                       380
private bool TryStepLeftUp(Func<ulong, bool> handler, ulong left, ulong stepFrom)
                                                                                       381
                                                                                                           UpdateOneCore(variant, bestVariant)
                                                                                       382
  var upStep = stepFrom;
                                                                                       383
  var first Target = Links. Unsvnc. GetSource(upStep):
                                                                                       384
  while (first Target != left && first Target != upStep)
                                                                                                     return bestVariant;
                                                                                       385
                                                                                       386
      upStep = firstTarget;
                                                                                       387
     first Target = Links. Unsync. Get Target (upStep);
                                                                                                  private void UpdateOneCore(ulong sequence, ulong newSequence)
                                                                                       388
                                                                                       389
     (firstTarget == left)
                                                                                                     if (Options.UseGarbageCollection)
                                                                                       390
                                                                                       391
     return handler(stepFrom);
                                                                                                        var sequenceElements = GetSequenceElements(sequence):
                                                                                       392
                                                                                                        var sequenceElementsContents = new
                                                                                       393
  return true;
                                                                                                        → UInt64Link(Links.GetLink(sequenceElements));
                                                                                                        var sequenceLink = GetSequenceByElements(sequenceElements):
                                                                                       394
                                                                                                        var newSequenceElements = GetSequenceElements(newSequence);
                                                                                       395
#endregion
```

```
var newSequenceLink = GetSequenceByElements(newSequenceElements):
                                                                                                      var sequenceLink = GetSequenceByElements(sequenceElements):
                                                                                     457
     if (Options. UseCascade Update | CountReferences (sequence) == 0)
                                                                                                      if (Options.UseCascadeDelete || CountReferences(link) == 0)
                                                                                      458
                                                                                      459
        if (sequenceLink! = constants.Null)
                                                                                                          if (sequenceLink != constants.Null)
                                                                                      460
                                                                                      461
           Links. Unsvnc. Merge (sequenceLink, newSequenceLink):
                                                                                                            Links.Unsvnc.Delete(sequenceLink):
                                                                                      462
                                                                                      463
        Links.Unsvnc.Merge(sequenceElements, newSequenceElements);
                                                                                                          Links.Unsync.Delete(link);
                                                                                      464
                                                                                      465
     ClearGarbage(sequenceElementsContents.Source):
                                                                                                       ClearGarbage(sequenceElementsContents.Source):
                                                                                      466
     ClearGarbage(sequenceElementsContents.Target);
                                                                                                       ClearGarbage(sequenceElementsContents.Target);
                                                                                      467
                                                                                      468
  else
                                                                                                    else
                                                                                      469
                                                                                      470
       (Options. UseSequenceMarker)
                                                                                                        (Options. UseSequenceMarker)
                                                                                      471
                                                                                     472
        var sequenceElements = GetSequenceElements(sequence);
                                                                                                         var sequenceElements = GetSequenceElements(link);
                                                                                      473
        var sequenceLink = GetSequenceBvElements(sequenceElements);
                                                                                                         var sequenceLink = GetSequenceByElements(sequenceElements);
                                                                                     474
        var newSequenceElements = GetSequenceElements(newSequence);
                                                                                                         if (Options.UseCascadeDelete || CountReferences(link) == 0)
                                                                                      475
        var newSequenceLink = GetSequenceByElements(newSequenceElements);
                                                                                      476
        if (Options.UseCascadeUpdate || CountReferences(sequence) == 0)
                                                                                                            if (sequenceLink! = constants.Null)
                                                                                      477
                                                                                      478
           if (sequenceLink != constants.Null)
                                                                                                               Links. Unsync. Delete (sequence Link):
                                                                                     479
                                                                                      480
              Links.Unsync.Merge(sequenceLink, newSequenceLink);
                                                                                                             Links.Unsync.Delete(link);
                                                                                     481
                                                                                      482
           Links.Unsync.Merge(sequenceElements, newSequenceElements);
                                                                                      483
                                                                                                       else
                                                                                      484
                                                                                      485
                                                                                                          if (Options.UseCascadeDelete || CountReferences(link) == 0)
     else
                                                                                      486
                                                                                      487
        if (Options.UseCascadeUpdate || CountReferences(sequence) == 0)
                                                                                                            Links.Unsync.Delete(link);
                                                                                      488
                                                                                      489
           Links.Unsync.Merge(sequence, newSequence);
                                                                                      490
                                                                                      491
                                                                                      492
                                                                                      493
                                                                                                 #endregion
                                                                                      494
                                                                                      495
                                                                                                 #region Compactification
                                                                                      496
#endregion
                                                                                      497
                                                                                                   / <remarks>
#region Delete
                                                                                      498
                                                                                                    best Variant можно выбирать по максимальному числу использований.
                                                                                      499
public void Delete(params ulong[] sequence)
                                                                                                    но балансированный позволяет гарантировать уникальность (если есть
                                                                                      500
                                                                                                     возможность,
                                                                                                    гарантировать его использование в других местах).
  Sync.ExecuteWriteOperation(() =>
                                                                                     501
                                                                                      502
                                                                                                   / Получается этот метод должен игнорировать
       / TODO: Check all options only ones before loop execution
                                                                                     503
     foreach (var linkToDelete in Each(sequence))
                                                                                                 → Options.EnforceSingleSequenceVersionOnWrite
                                                                                                 /// < / \text{remarks} >
                                                                                     504
        DeleteOneCore(linkToDelete):
                                                                                                 public ulong Compact(params ulong | sequence)
                                                                                     505
                                                                                      506
                                                                                                    return Sync. ExecuteWriteOperation(() =>
                                                                                     507
                                                                                     508
                                                                                                       if (sequence.IsNullOrEmpty())
                                                                                     509
private void DeleteOneCore(ulong link)
                                                                                     510
                                                                                                         return constants.Null;
                                                                                     5.1.1
  if (Options. UseGarbageCollection)
                                                                                                      Links.EnsureEachLinkExists(sequence);
                                                                                     513
     var sequenceElements = GetSequenceElements(link);
                                                                                                      return CompactCore(sequence);
                                                                                     514
     var sequenceElementsContents = new
                                                                                     515
         UInt64Link(Links.GetLink(sequenceElements));
                                                                                     516
```

```
576
                                                                                                       sequences = sequences;
                                                                                                        patternSequence = patternSequence:
[MethodImpl(MethodImplOptions.AggressiveInlining)]
                                                                                     577
                                                                                                        linksInSequence = new HashSet < linkIndex > (patternSequence, Where (x = > x)
private ulong CompactCore(params ulong[] sequence) => UpdateCore(sequence,
                                                                                     578
                                                                                                         != constants.Any && x != ZeroOrMany));

⇒ sequence):

                                                                                                       results = results:
                                                                                     579
                                                                                                       stopableHandler = stopableHandler;
#endregion
                                                                                     580
                                                                                                       readAsElements = readAsElements;
                                                                                     581
#region Garbage Collection
                                                                                     582
                                                                                     583
  / <remarks>
                                                                                                   protected override bool IsElement(IList<ulong> link) => base.IsElement(link) ||
                                                                                     584
   TODO: Добавить дополнительный обработчик / coбытие CanBeDeleted
                                                                                                        ( readAsElements!= null &&
   которое можно определить извне или в унаследованном классе
                                                                                                         readAsElements.Contains(Links.GetIndex(link))) ||
                                                                                                        linksInSequence.Contains(Links.GetIndex(link));
[MethodImpl(MethodImplOptions.AggressiveInlining)]
                                                                                     585
private bool IsGarbage(ulong link) => link! = Options.SequenceMarkerLink &&
                                                                                                   public bool FullMatch(LinkIndex sequenceToMatch)
                                                                                     586
∴ !Links.Unsync.IsPartialPoint(link) && Links.Count(link) == 0;
                                                                                     587
                                                                                                        filterPosition = 0:
                                                                                     588
private void ClearGarbage(ulong link)
                                                                                                      foreach (var part in Walk(sequenceToMatch))
                                                                                     589
                                                                                     500
    (IsGarbage(link))
                                                                                                         if (!FullMatchCore(Links.GetIndex(part)))
                                                                                     591
                                                                                     592
     var contents = new UInt64Link(Links.GetLink(link));
                                                                                                            break:
                                                                                     593
     Links.Unsync.Delete(link);
                                                                                     594
     ClearGarbage(contents.Source);
                                                                                     595
     ClearGarbage(contents.Target);
                                                                                                      return filterPosition == patternSequence.Count;
                                                                                     596
                                                                                     597
                                                                                     598
                                                                                                   private bool FullMatchCore(LinkIndex element)
                                                                                     599
#endregion
                                                                                     600
                                                                                                      if (filterPosition == patternSequence.Count)
                                                                                     601
#region Walkers
                                                                                     602
                                                                                                          filterPosition = -2; // Длиннее чем нужно
public bool EachPart(Func<ulong, bool> handler, ulong sequence)
                                                                                     603
                                                                                                         return false:
                                                                                     604
                                                                                     605
  return Sync.ExecuteReadOperation(() =>
                                                                                                           patternSequence[ filterPosition] != constants.Any
                                                                                                      && element != patternSequence [filterPosition])
                                                                                     607
     var links = Links.Unsvnc;
     var walker = new RightSequenceWalker < ulong > (links);
                                                                                     608
                                                                                                           filterPosition = -1;
                                                                                     609
     foreach (var part in walker. Walk(sequence))
                                                                                                         return false; // Начинается/Продолжается иначе
                                                                                     610
                                                                                     611
        if (!handler(links.GetIndex(part)))
                                                                                                        filterPosition++;
                                                                                     612
                                                                                                      return true;
                                                                                     613
           return false;
                                                                                     614
                                                                                     615
                                                                                                   public void AddFullMatchedToResults(ulong sequenceToMatch)
                                                                                     616
     return true:
                                                                                     617
                                                                                                      if (FullMatch(sequenceToMatch))
                                                                                     618
                                                                                     619
                                                                                                         results.Add(sequenceToMatch);
public class Matcher: RightSequenceWalker<ulong>
                                                                                     620
                                                                                     621
  private readonly Sequences sequences;
                                                                                     622
  private readonly IList<LinkIndex> patternSequence;
                                                                                     623
  private readonly HashSet<LinkIndex> _linksInSequence;
                                                                                                   public bool HandleFullMatched(ulong sequenceToMatch)
                                                                                     624
  private readonly HashSet < LinkIndex > results;
                                                                                     625
                                                                                                      if (FullMatch(sequenceToMatch) && results.Add(sequenceToMatch))
  private readonly Func<ulong, bool> stopableHandler;
                                                                                     626
  private readonly HashSet < ulong > readAsElements;
                                                                                     627
  private int filterPosition;
                                                                                                         return stopableHandler(sequenceToMatch);
                                                                                     628
                                                                                     629
  public Matcher(Sequences sequences, IList<LinkIndex> patternSequence,
                                                                                                      return true:
                                                                                     630
      HashSet < LinkIndex > results, Func < LinkIndex, bool > stopableHandler,
                                                                                     631
      HashSet < LinkIndex > readAsElements = null
                                                                                     632
                                                                                                   public bool HandleFullMatchedSequence(ulong sequenceToMatch)
                                                                                     633
      : base(sequences.Links.Unsync)
```

```
if (PartialMatch(sequenceToMatch))
                                                                                  696
  var sequence = sequences.GetSequenceByElements(sequenceToMatch);
                                                                                  697
  if (sequence! = constants.Null && FullMatch(sequenceToMatch) &&
                                                                                                      return stopableHandler(sequenceToMatch);
                                                                                  698
        results.Add(sequenceToMatch))
                                                                                  699
                                                                                                   return true;
                                                                                  700
                                                                                  701
      return stopableHandler(sequence);
                                                                                  702
                                                                                                public void AddAllPartialMatchedToResults(IEnumerable<ulong>
                                                                                  703
   return true;
                                                                                                    sequencesToMatch)
                                                                                  704
                                                                                                   foreach (var sequenceToMatch in sequencesToMatch)
   <remarks>
                                                                                  705
   TODO: Add support for LinksConstants.Any
                                                                                  706
   </remarks>
                                                                                                      if (PartialMatch(sequenceToMatch))
                                                                                  707
public bool PartialMatch(LinkIndex sequenceToMatch)
                                                                                  708
                                                                                                         results.Add(sequenceToMatch);
                                                                                  709
    filterPosition = -1;
                                                                                  710
   foreach (var part in Walk(sequenceToMatch))
                                                                                  711
                                                                                  712
      if (!PartialMatchCore(Links.GetIndex(part)))
                                                                                  713
                                                                                  714
        break:
                                                                                                     AddAllPartialMatchedToResultsAndReadAsElements(IEnumerable<ulong>
                                                                                                     sequencesToMatch)
                                                                                  715
  return filterPosition == patternSequence.Count - 1;
                                                                                                   foreach (var sequenceToMatch in sequencesToMatch)
                                                                                  716
                                                                                  717
                                                                                                      if (PartialMatch(sequenceToMatch))
                                                                                  718
private bool PartialMatchCore(LinkIndex element)
                                                                                  719
                                                                                                          readAsElements.Add(sequenceToMatch);
                                                                                  720
  if ( filterPosition == ( patternSequence.Count - 1))
                                                                                                          results.Add(sequenceToMatch);
                                                                                  721
                                                                                  722
      return false; // Нашлось
                                                                                  723
                                                                                  724
     ( filterPosition \geq 0)
                                                                                  725
                                                                                  726
      if (element == patternSequence[filterPosition + 1])
                                                                                  727
                                                                                             #endregion
                                                                                  728
         filterPosition++;
                                                                                  729
                                                                                   ./Sequences/Sequences.Experiments.cs
          filterPosition = -1;
                                                                                       using System;
                                                                                       using LinkIndex = System.UInt64;
                                                                                       using System. Collections. Generic:
      filterPosition < 0
                                                                                       using Stack = System.Collections.Generic.Stack<ulong>;
                                                                                       using System.Ling;
      if (element == patternSequence |0|)
                                                                                       using System. Text;
                                                                                       using Platform.Collections:
         filterPosition = 0;
                                                                                       using Platform. Numbers;
                                                                                       using Platform Data Exceptions;
                                                                                       using Platform. Data. Sequences;
  return true; // Ищем дальше
                                                                                       using Platform.Data.Doublets.Sequences.Frequencies.Counters;
                                                                                       using Platform. Data. Doublets. Sequences. Walkers;
                                                                                   12
public void AddPartialMatchedToResults(ulong sequenceToMatch)
                                                                                       namespace Platform.Data.Doublets.Sequences
                                                                                   14
                                                                                   15
  if (PartialMatch(sequenceToMatch))
                                                                                   16
                                                                                          partial class Sequences
                                                                                   17
      results.Add(sequenceToMatch);
                                                                                             #region Create All Variants (Not Practical)
                                                                                   18
                                                                                   19
                                                                                                 <remarks>
                                                                                   20
                                                                                                 Number of links that is needed to generate all variants for
                                                                                   21
public bool HandlePartialMatched(ulong sequenceToMatch)
                                                                                                 sequence of length N corresponds to https://oeis.org/A014143/list sequence.
                                                                                   22
                                                                                                 </remarks>
                                                                                   23
```

655

```
public ulong [ CreateAllVariants2(ulong [ sequence)
                                                                                                                         if (sequence.IsNullOrEmpty())
24
25
                                                                                                        84
              return Sync.ExecuteWriteOperation(() =>
                                                                                                                            return new List < ulong > ():
26
                                                                                                        85
27
                 if (sequence.IsNullOrEmpty())
                                                                                                                         Links.Unsync.EnsureEachLinkExists(sequence);
                                                                                                        87
                                                                                                                         if (sequence.Length == 1)
                                                                                                        88
29
                    return new ulong[0];
                                                                                                        89
                                                                                                                            return new List < ulong > { sequence[0] };
31
                 Links.EnsureEachLinkExists(sequence);
                                                                                                        91
                                                                                                                         var results = new List < ulong > ((int) MathHelpers.Catalan(sequence.Length));
33
                 if (sequence. Length == 1)
                                                                                                        92
                                                                                                                         return CreateAllVariants1Core(sequence, results);
                                                                                                        93
                    return sequence;
35
                                                                                                        94
                                                                                                        95
                 return CreateAllVariants2Core(sequence, 0, sequence.Length - 1);
                                                                                                        96
                                                                                                                   private List<ulong> CreateAllVariants1Core(ulong[] sequence, List<ulong> results)
38
                                                                                                       97
39
                                                                                                        98
                                                                                                                      if (sequence. Length == 2)
                                                                                                       αa
           private ulong CreateAllVariants2Core(ulong sequence, long startAt, long stopAt)
41
                                                                                                       100
                                                                                                                         var link = Links.Unsync.CreateAndUpdate(sequence[0], sequence[1]);
                                                                                                       101
43
                                                                                                                         if (link == constants.Null)
                                                                                                       102
              if ((stopAt - startAt) < 0)
44
                                                                                                       103
45
                                                                                                                            throw new NotImplementedException("Creation cancellation is not
                                                                                                       104
                 throw new ArgumentOutOfRangeException(nameof(startAt), "startAt должен
                                                                                                                            \rightarrow implemented."):
                     быть меньше или равен stopAt");
                                                                                                       105
                                                                                                                         results.Add(link):
                                                                                                       106
    \#endif
48
                                                                                                                         return results:
                                                                                                       107
              if ((stopAt - startAt) == 0)
49
                                                                                                       108
50
                                                                                                                      var innerSequenceLength = sequence.Length - 1:
                                                                                                       109
                 return new[] { sequence[startAt] }:
                                                                                                                      var innerSequence = new ulong[innerSequenceLength];
5.1
                                                                                                       110
                                                                                                                      for (var li = 0; li < innerSequenceLength; <math>li++)
                                                                                                       111
                ((stopAt - startAt) == 1)
53
                                                                                                       112
                                                                                                                         var link = Links.Unsync.CreateAndUpdate(sequence[li], sequence[li + 1]);
54
                                                                                                       113
                 return new [] { Links.Unsync.CreateAndUpdate(sequence[startAt],
                                                                                                       114
                                                                                                                         if (link == constants.Null)
                     sequence[stopAt]) };
                                                                                                       115
                                                                                                                            throw new NotImplementedException ("Creation cancellation is not
                                                                                                       116
              var variants = new ulong[(ulong)MathHelpers.Catalan(stopAt - startAt)];
                                                                                                                            \rightarrow implemented.");
                                                                                                       117
              for (var splitter = startAt; splitter < stopAt; splitter++)
                                                                                                                         for (var isi = 0; isi < li; isi++)
                                                                                                       118
                                                                                                       119
                 var left = CreateAllVariants2Core(sequence, startAt, splitter);
                                                                                                                            innerSequence[isi] = sequence[isi];
                                                                                                       120
                 var right = CreateAllVariants2Core(sequence, splitter + 1, stopAt);
                                                                                                       121
                 for (var i = 0; i < left.Length; i++)
                                                                                                                         innerSequence[li] = link;
                                                                                                       122
                                                                                                                         for (var isi = li + 1; isi < innerSequenceLength; isi++)
                                                                                                       123
                    for (\text{var } \mathbf{j} = 0; \mathbf{j} < \text{right.Length}; \mathbf{j} + +)
                                                                                                       124
                                                                                                                            innerSequence[isi] = sequence[isi + 1];
                                                                                                       125
                       var variant = Links.Unsync.CreateAndUpdate(left[i], right[j]);
                                                                                                       126
                       if (variant == constants.Null)
                                                                                                                         CreateAllVariants1Core(innerSequence, results);
                                                                                                       127
                                                                                                       128
                           throw new NotImplementedException("Creation cancellation is not
                                                                                                                      return results;
                                                                                                       129
                           \rightarrow implemented.");
                                                                                                       130
                                                                                                       131
                                                                                                                  #endregion
                                                                                                       132
                       variants[last++] = variant;
                                                                                                       133
                                                                                                                   public HashSet < ulong > Each1(params ulong | sequence)
                                                                                                       134
                                                                                                       135
75
                                                                                                                      var visitedLinks = new HashSet<ulong>(); // Заменить на bitstring
                                                                                                       136
              return variants:
76
                                                                                                                      Each1(link = >
                                                                                                       137
77
                                                                                                       138
78
                                                                                                                         if (!visitedLinks.Contains(link))
           public List < ulong > Create All Variants 1 (params ulong | sequence)
                                                                                                       139
79
                                                                                                       140
80
              return Sync.ExecuteWriteOperation(() =>
                                                                                                                            visitedLinks.Add(link); // изучить почему случаются повторы
                                                                                                       141
81
82
```

```
var visitedLinks = new HashSet<ulong>(); // Заменить на bitstring
                                                                                                        205
142
                  return true:
143
                                                                                                                        EachPartCore(link =>
                                                                                                        206
                 , sequence);
144
                                                                                                        207
               return visitedLinks:
145
                                                                                                                           if (!visitedLinks.Contains(link))
                                                                                                        208
146
                                                                                                        209
147
                                                                                                                              visitedLinks.Add(link); // изучить почему случаются повторы
                                                                                                        210
            private void Each1(Func<ulong, bool> handler, params ulong[] sequence)
148
                                                                                                                             return handler(link);
                                                                                                        211
149
                                                                                                        212
                 (sequence. Length == 2)
150
                                                                                                                           return true;
                                                                                                        213
151
                                                                                                                        }, sequence);
                                                                                                        214
                  Links.Unsync.Each(sequence[0], sequence[1], handler);
152
                                                                                                        215
153
                                                                                                        216
               else
                                                                                                                    private void EachPartCore(Func<ulong, bool> handler, params ulong[] sequence)
154
                                                                                                        217
155
                                                                                                        218
                  var innerSequenceLength = sequence.Length - 1;
156
                                                                                                                        if (sequence.IsNullOrEmpty())
                                                                                                        219
                  for (var li = 0; li < innerSequenceLength; <math>li++)
157
                                                                                                        220
158
                                                                                                        221
                                                                                                                           return:
                     var left = sequence[li];
159
                                                                                                        222
                                                                                                                        Links.EnsureEachLinkIsAnyOrExists(sequence);
                     var right = sequence[li + 1]:
160
                                                                                                        223
                     if (left == 0 \&\& right == 0)
                                                                                                                        if (sequence.Length == 1)
                                                                                                        224
162
                                                                                                        225
                        continue:
                                                                                                                           var link = sequence[0];
163
                                                                                                        226
                                                                                                                           if (link > 0)
164
                                                                                                        227
                     var linkIndex = li;
165
                                                                                                        228
                     ulong[] innerSequence = null;
                                                                                                                              handler(link);
                                                                                                        229
                     Links.Unsvnc.Each(left, right, doublet =>
167
                                                                                                        230
                                                                                                                           else
                                                                                                        231
                        if (innerSequence == null)
                                                                                                        232
                                                                                                                              Links. Each( constants. Any, constants. Any, handler);
                                                                                                        233
                            innerSequence = new ulong[innerSequenceLength];
171
                                                                                                        234
                            for (var isi = 0; isi < linkIndex; isi++)
172
                                                                                                        235
                                                                                                                        else if (sequence.Length == 2)
                                                                                                        236
                               innerSequence[isi] = sequence[isi];
174
                                                                                                        237
                                                                                                                              links.Each(sequence[0], sequence[1], handler);
                                                                                                        238
                            for (var\ isi = linkIndex + 1;\ isi < innerSequenceLength;\ isi++)
176
                                                                                                        239
                                                                                                        240
                               innerSequence[isi] = sequence[isi + 1];
178
                                                                                                        241
                                                                                                                           Links.Each(sequence|1|, constants.Any, doublet =>
179
                                                                                                        ^{242}
                                                                                                                              var match = Links.SearchOrDefault(sequence[0], doublet);
180
                                                                                                        243
                        innerSequence[linkIndex] = doublet;
181
                                                                                                                              if (match! = constants.Null)
                                                                                                        244
                        Each1(handler, innerSequence);
182
                                                                                                        245
                        return constants.Continue;
                                                                                                                                 handler(match);
                                                                                                        ^{246}
                                                                                                        247
                                                                                                                              return true:
185
                                                                                                        248
186
                                                                                                        ^{249}
187
                                                                                                        250
                                                                                                                                     ... X O
                                                                                                                               Τо
188
                                                                                                        251
            public HashSet < ulong > EachPart (params ulong | sequence)
189
                                                                                                        252
                                                                                                                           Links.Each( constants.Any, sequence[0], doublet =>
190
                                                                                                        253
               var visitedLinks = new HashSet<ulong>(); // Заменить на bitstring
191
                                                                                                                              var match = Links.SearchOrDefault(doublet, sequence[1]);
                                                                                                        254
               EachPartCore(link =>
192
                                                                                                                             if (\text{match } != 0)
                                                                                                        255
193
                                                                                                        256
                  if (!visitedLinks.Contains(link))
194
                                                                                                                                 handler(match);
                                                                                                        257
195
                                                                                                        258
                     visitedLinks.Add(link); // изучить почему случаются повторы
196
                                                                                                                              return true;
                                                                                                        259
197
                                                                                                        260
                  return true;
198
                                                                                                        261
                                                                                                                                      X O
               }, sequence);
199
                                                                                                        262
               return visitedLinks;
200
                                                                                                                           PartialStep\overline{Right}(x => handler(x), sequence[0], sequence[1]);
                                                                                                        263
201
                                                                                                        264
202
                                                                                                                        else
                                                                                                        265
            public void EachPart(Func<ulong, bool> handler, params ulong sequence)
203
                                                                                                        266
204
```

```
// TODO: Implement other variants
                                                                                         330
                                                                                         331
                                                                                                     private void TryStepLeftUp(Action<ulong> handler, ulong left, ulong stepFrom)
                                                                                         332
                                                                                         333
                                                                                                        var upStep = stepFrom:
                                                                                         334
private void PartialStepRight(Action<ulong> handler, ulong left, ulong right)
                                                                                                        var first Target = Links. Unsync.GetSource(upStep);
                                                                                         335
                                                                                                        while (first Target != left && first Target != upStep)
                                                                                         336
  Links. Unsync. Each (constants. Any, left, doublet =>
                                                                                         337
                                                                                                           upStep = firstTarget:
                                                                                         338
     StepRight(handler, doublet, right):
                                                                                                           first Target = Links. Unsync. Get Target (upStep);
                                                                                         339
     if (left != doublet)
                                                                                         340
                                                                                                        if (firstTarget == left)
                                                                                         341
         PartialStepRight(handler, doublet, right);
                                                                                         342
                                                                                                           handler(stepFrom);
                                                                                         343
     return true;
                                                                                         344
                                                                                         345
                                                                                         346
                                                                                                     private bool StartsWith(ulong sequence, ulong link)
                                                                                         347
private void StepRight(Action<ulong> handler, ulong left, ulong right)
                                                                                         348
                                                                                                        var upStep = sequence;
                                                                                         349
  Links.Unsync.Each(left, constants.Any, rightStep =>
                                                                                                        var firstSource = Links.Unsync.GetSource(upStep);
                                                                                         350
                                                                                                        while (firstSource != link && firstSource != upStep)
                                                                                         351
     TryStepRightUp(handler, right, rightStep);
                                                                                         352
     return true:
                                                                                                           upStep = firstSource;
                                                                                         353
                                                                                                           firstSource = Links.Unsync.GetSource(upStep);
                                                                                         354
                                                                                         355
                                                                                                        return firstSource == link;
                                                                                         356
private void TryStepRightUp(Action<ulong> handler, ulong right, ulong stepFrom)
                                                                                         357
                                                                                         358
  var upStep = stepFrom;
                                                                                                     private bool EndsWith(ulong sequence, ulong link)
                                                                                         359
  var firstSource = Links.Unsync.GetTarget(upStep);
                                                                                         360
  while (firstSource != right && firstSource != upStep)
                                                                                                        var upStep = sequence;
                                                                                         361
                                                                                                        var lastTarget = Links.Unsvnc.GetTarget(upStep):
                                                                                         362
     upStep = firstSource:
                                                                                                        while (last Target != link && last Target != upStep)
                                                                                         363
     firstSource = Links.Unsvnc.GetSource(upStep):
                                                                                         364
                                                                                                           upStep = lastTarget;
                                                                                         365
     (firstSource == right)
                                                                                                           lastTarget = Links.Unsync.GetTarget(upStep);
                                                                                         366
                                                                                         367
     handler(stepFrom);
                                                                                                        return lastTarget == link;
                                                                                         368
                                                                                         369
                                                                                         370
                                                                                                     public List<ulong> GetAllMatchingSequences0(params ulong[] sequence)
                                                                                         371
// TODO: Test
                                                                                         372
private void PartialStepLeft(Action < ulong > handler, ulong left, ulong right)
                                                                                                        return Sync. Execute Read Operation (() = >
                                                                                         373
                                                                                         374
  Links.Unsync.Each(right, constants.Any, doublet =>
                                                                                                           var results = new List < ulong > ():
                                                                                         375
                                                                                                           if (sequence. Length > 0)
                                                                                         376
     StepLeft(handler, left, doublet);
                                                                                         377
     if (right != doublet)
                                                                                                              Links.EnsureEachLinkExists(sequence);
                                                                                         378
                                                                                                              var firstElement = sequence[0];
                                                                                         379
        PartialStepLeft(handler, left, doublet);
                                                                                                              if (sequence.Length == 1)
                                                                                         380
                                                                                         381
     return true;
                                                                                                                 results.Add(firstElement);
                                                                                         382
                                                                                                                 return results:
                                                                                         383
                                                                                         384
                                                                                                              if (sequence. Length == 2)
                                                                                         385
private void StepLeft(Action<ulong> handler, ulong left, ulong right)
                                                                                                                 var doublet = Links.SearchOrDefault(firstElement, sequence[1]):
                                                                                         387
  Links.Unsync.Each( constants.Any, right, leftStep =>
                                                                                                                 if (doublet != constants.Null)
                                                                                         388
                                                                                         389
     TryStepLeftUp(handler, left, leftStep);
                                                                                                                    results. Add(doublet);
                                                                                         390
     return true;
                                                                                         391
```

```
return results:
                                                                                             451
                                                                                             452
         var linksInSequence = new HashSet < ulong > (sequence);
                                                                                             453
         void handler(ulong result)
                                                                                             454
                                                                                             455
            var filterPosition = 0:
                                                                                             456
            StopableSequenceWalker.WalkRight(result, Links.Unsync.GetSource,
                                                                                             457
                Links. Unsync. Get Target.
                                                                                             458
               x =  linksInSequence.Contains(x) || Links.Unsync.GetTarget(x) == x,
                                                                                            459
                                                                                             461
                     (filterPosition == sequence.Length)
                                                                                             462
                                                                                             463
                     filterPosition = -2; // Длиннее чем нужно
                                                                                             464
                     return false;
                                                                                             465
                                                                                             466
                     (x \mid = sequence[filterPosition])
                                                                                             467
                                                                                             468
                     filterPosition = -1:
                                                                                             469
                     return false; // Начинается иначе
                                                                                             470
                  filterPosition++;
                                                                                             471
                                                                                             472
                  return true;
                                                                                             473
               (filterPosition == sequence.Length)
                                                                                             474
                                                                                             475
               results.Add(result);
                                                                                             476
                                                                                             478
           (sequence. Length \geq = 2)
                                                                                             479
                                                                                             480
            StepRight(handler, sequence[0], sequence[1]);
                                                                                             481
                                                                                             482
         var last = sequence.Length - 2;
         for (var i = 1; i < last; i++)
                                                                                             483
            PartialStepRight(handler, sequence[i], sequence[i + 1]);
                                                                                             484
           (sequence.Length \geq = 3)
            StepLeft(handler, sequence[sequence.Length - 2], sequence[sequence.Length
             \hookrightarrow -1|);
                                                                                             485
                                                                                             486
     return results:
                                                                                             487
                                                                                             488
public HashSet < ulong > Get AllMatchingSequences1(params ulong[] sequence)
                                                                                             489
                                                                                             490
  return Sync.ExecuteReadOperation(() =>
                                                                                             491
                                                                                             492
      var results = new HashSet < ulong > ();
                                                                                             493
      if (sequence. Length > 0)
         Links.EnsureEachLinkExists(sequence);
                                                                                             495
         var firstElement = sequence |0|;
         if (sequence.Length == 1)
                                                                                             496
                                                                                             497
            results.Add(firstElement);
                                                                                             498
            return results;
                                                                                             499
```

393

394

395

396

398

401

402

405

406

409

410

411

412

413

415

410

420

421

422

423

424

425

426

427

428

429

430

431

432

433

434

435

436

437

438

439

440

441

442

443

444

445

446

448

```
if (sequence. Length == 2)
           var doublet = Links.SearchOrDefault(firstElement, sequence[1]);
           if (doublet != constants.Null)
              results. Add(doublet):
           return results:
        var matcher = new Matcher(this, sequence, results, null);
        if (sequence. Length \geq 2)
           StepRight(matcher.AddFullMatchedToResults, sequence[0], sequence[1]):
        var last = sequence.Length - 2;
        for (var i = 1: i < last: i++)
           PartialStepRight(matcher.AddFullMatchedToResults, sequence[i],
            \rightarrow sequence[i + 1]);
           (sequence. Length \geq = 3)
           StepLeft(matcher.AddFullMatchedToResults, sequence[sequence.Length-
               2], sequence[sequence.Length - 1]);
     return results:
public const int MaxSequenceFormatSize = 200:
public string FormatSequence(LinkIndex sequenceLink, params LinkIndex[]
    knownElements) => FormatSequence(sequenceLink, (sb, x) => sb.Append(x),
    true, knownElements):
public string FormatSequence(LinkIndex sequenceLink, Action < StringBuilder,
    LinkIndex> elementToString, bool insertComma, params LinkIndex[]
    knownElements) => Links.SyncRoot.ExecuteReadOperation(() =>
    FormatSequence(Links.Unsync, sequenceLink, elementToString, insertComma,
    knownElements));
private string FormatSequence(ILinks<LinkIndex> links, LinkIndex sequenceLink,
    Action < StringBuilder, LinkIndex > elementToString, bool insertComma, params
    LinkIndex[] knownElements)
   var linksInSequence = new HashSet < ulong > (knownElements);
   //var entered = new HashSet < ulong > ():
   var sb = new StringBuilder();
   sb.Append('\{'\});
   if (links.Exists(sequenceLink))
     StopableSequenceWalker.WalkRight(sequenceLink, links.GetSource,

→ links.GetTarget,

        x => linksInSequence.Contains(x) || links.IsPartialPoint(x), element => //
             entered.AddAndReturnVoid, x = \{ \}, entered.DoNotContains
              (insertComma \&\& sb.Length > 1)
              sb.Append(',');
```

```
551
             /if (entered.Contains(element))
                                                                                        552
                                                                                         553
                 sb.Append('\{'\});
                                                                                        554
                 elementToString(sb, element);
                                                                                        555
                 sb.Append('}'):
                                                                                        556
                                                                                        557
                                                                                        558
            elementToString(sb, element);
                                                                                        559
                                                                                        560
            if (sb.Length < MaxSequenceFormatSize)
                                                                                        561
                                                                                        562
               return true;
                                                                                        563
            sb.Append(insertComma?", ...": "...");
                                                                                         564
            return false:
                                                                                        565
                                                                                         566
                                                                                        567
  sb.Append(');
                                                                                         568
  return sb.ToString();
                                                                                        569
                                                                                        570
                                                                                        571
public string SafeFormatSequence(LinkIndex sequenceLink, params LinkIndex)
                                                                                        572
    knownElements) => SafeFormatSequence(sequenceLink, (sb, x) =>
                                                                                        573
    sb.Append(x), true, knownElements):
                                                                                        574
                                                                                        575
public string SafeFormatSequence(LinkIndex sequenceLink, Action < StringBuilder,
                                                                                        576
    LinkIndex> elementToString, bool insertComma, params LinkIndex
                                                                                        577
                                                                                        578
    knownElements) => Links.SyncRoot.ExecuteReadOperation(() =>
                                                                                        579
    SafeFormatSequence(Links.Unsync, sequenceLink, elementToString,
    insertComma, knownElements));
                                                                                        580
private string SafeFormatSequence(ILinks<LinkIndex> links, LinkIndex sequenceLink,
    Action < String Builder, Link Index > element To String, bool insert Comma, params
                                                                                        582
    LinkIndex[] knownElements)
                                                                                        583
                                                                                        584
  var linksInSequence = new HashSet < ulong > (knownElements):
                                                                                        585
  var entered = new HashSet < ulong > ():
                                                                                        586
  var sb = new StringBuilder();
                                                                                        587
  sb.Append('\{'\});
                                                                                        588
  if (links.Exists(sequenceLink))
                                                                                        589
                                                                                        590
     StopableSequenceWalker.WalkRight(sequenceLink, links.GetSource,
                                                                                        591
          links.GetTarget.
                                                                                        592
        x =  linksInSequence.Contains(x) || links.IsFullPoint(x),
                                                                                        593
             entered. Add And Return Void, x = \{ \}, entered. Do Not Contains, element
                                                                                        594
                                                                                        595
                                                                                        596
            if (insertComma && sb.Length > 1)
                                                                                        597
                                                                                        598
               sb.Append(',');
                                                                                        599
                                                                                        600
            if (entered.Contains(element))
                                                                                        601
                                                                                         602
              sb.Append('\{'\});
                                                                                        603
              elementToString(sb, element);
                                                                                        604
              sb.Append(');
                                                                                         605
                                                                                         606
                                                                                         607
                                                                                         608
               element ToString(sb, element);
                                                                                        609
                                                                                        610
            if (sb.Length < MaxSequenceFormatSize)
```

502

503

504

505

506

507

509

511

512

513

514

515

516

517

518

519 520

521

522

523

524

525

526

527

528

529

530

531

533

534

535

536

537

538

539

541

542

543

544

545

547

548 549

```
return true:
            sb.Append(insertComma?", ...":"...");
            return false:
   sb. Append('}');
   return sb.ToString();
public List<ulong> GetAllPartiallyMatchingSequences0(params ulong[] sequence)
   return Sync. Execute Read Operation (() = >
      if (sequence.Length > 0)
         Links.EnsureEachLinkExists(sequence);
         var results = new HashSet < vlong > ();
         for (var i = 0; i < \text{sequence.Length}; i++)
            AllUsagesCore(sequence[i], results);
         var filteredResults = new List < ulong > ();
         var linksInSequence = new HashSet < ulong > (sequence);
         foreach (var result in results)
            var filterPosition = -1:
            StopableSequenceWalker.WalkRight(result, Links.Unsync.GetSource,
            → Links.Unsync.GetTarget,
               x =  linksInSequence.Contains(x) || Links.Unsync.GetTarget(x) == x,
                  if (filterPosition == (sequence.Length - 1))
                     return false;
                  if (filterPosition >= 0)
                     if (x == sequence[filterPosition + 1])
                        filterPosition++;
                     else
                        return false;
                  if (filterPosition < 0)
                     if (x == sequence[0])
                        filterPosition = 0;
                  return true:
              (filterPosition == (sequence.Length - 1))
               filteredResults.Add(result);
```

```
return filteredResults:
                                                                                                                  var lastResults = new HashSet < ulong >();
                                                                                           671
                                                                                           672
                                                                                                                  var first = sequence.First(x => x != LinksConstants.Anv):
     return new List < ulong >();
                                                                                           673
                                                                                                                  var last = sequence.Last(x => x != LinksConstants.Anv):
                                                                                           674
                                                                                           675
                                                                                                                  AllUsagesCore(first, firstResults):
                                                                                           676
public HashSet < ulong > Get AllPartiallyMatchingSequences1(params ulong[] sequence)
                                                                                          677
                                                                                                                  AllUsagesCore(last, lastResults);
                                                                                                                  firstResults.IntersectWith(lastResults):
                                                                                           679
  return Sync.ExecuteReadOperation(() =>
                                                                                           680
                                                                                           681
                                                                                                                   //for (var i = 0: i < sequence.Length: i++)
     if (sequence.Length > 0)
                                                                                                                       AllUsagesCore(sequence[i], results);
                                                                                           682
                                                                                           683
         Links.EnsureEachLinkExists(sequence);
                                                                                                                  var filteredResults = new HashSet < ulong > ():
                                                                                           684
         var results = new HashSet < ulong > ():
                                                                                                                  var matcher = new Matcher(this, sequence, filteredResults, null):
                                                                                           685
         for (var i = 0; i < \text{sequence.Length}; i++)
                                                                                                                  matcher.AddAllPartialMatchedToResults(firstResults);
                                                                                           686
                                                                                                                  return filteredResults:
                                                                                           687
            AllUsagesCore(sequence[i], results);
                                                                                           688
                                                                                           689
         var filteredResults = new HashSet < ulong > ();
                                                                                                               return new HashSet < ulong > ();
                                                                                           690
         var matcher = new Matcher(this, sequence, filteredResults, null);
                                                                                           691
         matcher.AddAllPartialMatchedToResults(results);
                                                                                           692
         return filteredResults:
                                                                                           693
                                                                                                       public HashSet < ulong > GetAllPartiallyMatchingSequences3(params ulong | sequence)
                                                                                           694
     return new HashSet < ulong > ();
                                                                                           695
                                                                                                          return Sync.ExecuteReadOperation(() =>
                                                                                           696
                                                                                           697
                                                                                                             if (sequence.Length > 0)
                                                                                           698
public bool Get All Partially Matching Sequences 2 (Func < ulong, bool > handler, params
                                                                                           699
    ulong | sequence)
                                                                                                                Links. Ensure Each Link Is Any Or Exists (sequence):
                                                                                           700
                                                                                                                var firstResults = new HashSet < ulong > ():
                                                                                           701
  return Sync. Execute ReadOperation(() =>
                                                                                                                var lastResults = new HashSet < ulong > ():
                                                                                           702
                                                                                                                var first = sequence.First(x => x != constants.Any);
                                                                                           703
     if (sequence.Length > 0)
                                                                                                                var last = sequence.Last(x => x != constants.Any);
                                                                                           704
                                                                                                                AllUsagesCore(first, firstResults):
                                                                                           705
         Links.EnsureEachLinkExists(sequence);
                                                                                                                AllUsagesCore(last, lastResults);
                                                                                           706
                                                                                                                first Results. Intersect With (last Results):
                                                                                           707
         var results = new HashSet < ulong > ():
                                                                                                                //for (var i = 0; i < sequence. Length; i++)
                                                                                           708
         var filteredResults = new HashSet < ulong > ();
                                                                                                                     AllUsagesCore(sequence[i], results);
                                                                                           709
         var matcher = new Matcher(this, sequence, filteredResults, handler);
                                                                                                                var filteredResults = new HashSet < ulong > ();
                                                                                           710
         for (var i = 0; i < \text{sequence.Length}; i++)
                                                                                                                var matcher = new Matcher(this, sequence, filteredResults, null);
                                                                                           711
                                                                                                                matcher.AddAllPartialMatchedToResults(firstResults);
                                                                                           712
            if (!AllUsagesCore1(sequence[i], results, matcher.HandlePartialMatched))
                                                                                                                return filteredResults;
                                                                                           713
                                                                                           714
               return false:
                                                                                                             return new HashSet < ulong > ():
                                                                                           715
                                                                                           716
                                                                                           717
         return true;
                                                                                           718
                                                                                                      public HashSet < ulong > GetAllPartiallyMatchingSequences4(HashSet < ulong >
                                                                                           719
     return true;
                                                                                                           readAsElements, IList<ulong> sequence)
                                                                                           720
                                                                                                          return Sync. Execute Read Operation (() = >
                                                                                           721
//public HashSet<ulong> GetAllPartiallyMatchingSequences3(params ulong[]
                                                                                           722
                                                                                                             if (sequence.Count > 0)
                                                                                           723
    sequence)
                                                                                           724
                                                                                           725
                                                                                                                Links.EnsureEachLinkExists(sequence);
     return Sync.ExecuteReadOperation(() =>
                                                                                                                var results = new HashSet < LinkIndex > ();
                                                                                           726
                                                                                                                //var nextResults = new HashSet < ulong > ():
                                                                                           727
        if (sequence. Length > 0)
                                                                                                                  for (var i = 0; i < \text{sequence.Length}; i++)
                                                                                           728
                                                                                           729
            links.EnsureEachLinkIsAnyOrExists(sequence);
                                                                                                                     AllUsagesCore(sequence[i], nextResults);
                                                                                           730
                                                                                                                     if (results.IsNullOrEmpty())
                                                                                           731
           var firstResults = new HashSet < ulong > ();
                                                                                           732
```

```
results = nextResults;
                                                                                            791
                  nextResults = new HashSet < ulong >():
                                                                                             792
                                                                                             793
              else
                                                                                             794
                                                                                             795
                  results.IntersectWith(nextResults):
                                                                                            796
                  nextResults.Clear():
                                                                                            797
                                                                                            798
                                                                                             799
         var collector1 = new AllUsagesCollector1(Links.Unsvnc. results):
         collector1.Collect(Links.Unsync.GetLink(sequence[0]));
                                                                                            800
         var next = new HashSet < ulong > ():
                                                                                             801
         for (var i = 1: i < sequence.Count: i++)
                                                                                            802
                                                                                             803
            var collector = new AllUsagesCollector1(Links.Unsvnc. next):
                                                                                             804
            collector.Collect(Links.Unsvnc.GetLink(sequence[i])):
                                                                                             805
                                                                                             806
            results.IntersectWith(next);
                                                                                             807
            next.Clear();
                                                                                             808
         var filteredResults = new HashSet < ulong >();
                                                                                            809
         var matcher = new Matcher(this, sequence, filteredResults, null,
                                                                                            810
         \rightarrow readAsElements):
                                                                                             811
     matcher.AddAllPartialMatchedToResultsAndReadAsElements(results.OrderBy(x
         \Rightarrow => x)); // OrderBy is a Hack
                                                                                            812
         return filteredResults:
                                                                                            813
                                                                                             814
     return new HashSet < ulong > ();
                                                                                            815
                                                                                             816
                                                                                            817
                                                                                            818
  Does not work
                                                                                            819
public HashSet < ulong > Get AllPartiallyMatchingSequences5(HashSet < ulong >
                                                                                             820
    readAsElements, params ulong sequence
                                                                                            821
                                                                                            822
  var\ visited = new\ HashSet < ulong > ():
                                                                                            823
  var results = new HashSet < ulong > ();
                                                                                             824
  var matcher = new Matcher(this, sequence, visited, x =  { results.Add(x); return
   \rightarrow true: \}. readAsElements\):
                                                                                             826
  var last = sequence.Length - 1;
                                                                                            827
  for (var i = 0; i < last; i++)
                                                                                             828
                                                                                             829
      PartialStepRight(matcher.PartialMatch, sequence[i], sequence[i+1]);
                                                                                             830
                                                                                            831
  return results;
                                                                                             832
                                                                                            833
                                                                                             834
public List < ulong > GetAllPartiallyMatchingSequences(params ulong | sequence)
                                                                                            835
  return Sync.ExecuteReadOperation(() =>
                                                                                             836
                                                                                             837
       (\text{sequence.Length} > 0)
                                                                                             838
                                                                                             839
         Links.EnsureEachLinkExists(sequence);
                                                                                            840
          /var firstElement = sequence |0|
                                                                                            841
           / \text{if (sequence.Length} == 1)
                                                                                            842
                                                                                             843
               //results.Add(firstElement);
                                                                                             844
              return results;
                                                                                             845
                                                                                             846
             (sequence. Length == 2)
```

735

736

737

738

739

740

742

743

744

745

746

7/10

750

751

752

754

756

757

758

759

760

761

762

763

764

765

767

769

770

771

772

773

774

775

776

777

778

779

780

781

782

783

784

785

787

```
//var doublet = links.SearchCore(firstElement, sequence[1])
      '/if (doublet != \overline{D}oublets.Links.Null)
          results. Add(doublet):
     return results:
  var lastElement = sequence | sequence | Length - 1 |
 Func<ulong, bool> handler = x =>
     if (StartsWith(x, firstElement) && EndsWith(x, lastElement))
    results. Add(x):
     return true:
 if (sequence.Length \geq = 2)
     StepRight(handler, sequence[0], sequence[1]);
  var last = sequence. Length - 2;
  for (var i = 1; i < last; i++)
     PartialStepRight(handler, sequence[i], sequence[i + 1]):
 /if (sequence Length \geq = 3)
     StepLeft(handler, sequence[sequence.Length - 2],
    sequence[sequence.Length - 1]);
    //if (sequence.Length == 1)
         throw new NotImplementedException(); // all sequences, containing
\rightarrow this element?
      'if (sequence.Length == 2)
          var results = new List < ulong > ():
          PartialStepRight(results.Add, sequence[0], sequence[1]);
          return results:
      var matches = new List<List<ulong>>();
      var last = sequence.Length - 1;
      for (\text{var i} = 0; i < \text{last}; i++)
          var results = new List < ulong > ():
           /\mathrm{StepRight}(\mathrm{results.Add}, \mathrm{sequence[i]}, \mathrm{sequence[i+1]});
          PartialStepRight(results.Add, sequence[i], sequence[i + 1]);
          if (results.Count > 0)
             matches. Add(results);
          else
             return results:
          if (matches.Count == 2)
             var merged = new List < ulong >():
             for (\text{var } j = 0; j < \text{matches}[0].\text{Count}; j++)
                for (var k = 0; k < matches[1]. Count; k++)
                    CloseInnerConnections(merged.Add, matches | 0 | | j | .
    matches[1][k]);
               (\text{merged.Count} > 0)
                matches = new List<List<ulong>> { merged }:
                return new List < ulong >();
      'if (matches.Count > 0)
          var usages = new HashSet < ulong > ();
          for (int i = 0; i < \text{sequence.Length}; i++)
```

```
AllUsagesCore(sequence[i], usages);
                                                                                                         AllBottomUsagesCore(link, visits, usages);
                                                                                        ans
                                                                                                         return usages:
                                                                                        906
                    for (int i = 0: i < matches[0]. Count: i++)
                                                                                        907
                       AllUsagesCore(matches[0][i], usages);
                                                                                        908
                                                                                       909
                    /usages.UnionWith(matches[0]);
                                                                                                   private void AllBottomUsagesCore(ulong link, HashSet<ulong> visits,
                  return usages. ToList():
                                                                                       910

→ HashSet<ulong> usages)

         var firstLinkUsages = new HashSet < ulong > ():
                                                                                       911
                                                                                                      bool handler(ulong doublet)
         AllUsagesCore(sequence[0], firstLinkUsages);
                                                                                       912
         firstLinkUsages.Add(sequence[0]);
                                                                                       913
                                                                                                         if (visits.Add(doublet))
         //var previousMatchings = firstLinkUsages.ToList(); //new List<ulong>()
                                                                                       914
         ⇒ sequence[0] }; // or all sequences, containing this element?
                                                                                       915
                                                                                                            AllBottomUsagesCore(doublet, visits, usages);
                                                                                       916
         //return Get All Partially Matching Sequences Core (sequence, first Link Usages,
                                                                                       917
                                                                                                         return true:
                                                                                       918
         var results = new HashSet < ulong > ();
                                                                                       919
         foreach (var match in GetAllPartiallyMatchingSequencesCore(sequence,
                                                                                                        (Links.Unsync.Count(constants.Any, link) == 0)
                                                                                       920
             firstLinkUsages, 1))
                                                                                       921
                                                                                                         usages. Add(link);
                                                                                       922
           AllUsagesCore(match, results):
                                                                                       923
                                                                                                      else
                                                                                       924
         return results. ToList();
                                                                                       925
                                                                                                         Links. Unsync. Each(link, constants. Any, handler);
                                                                                        926
     return new List < ulong >();
                                                                                                         Links.Unsync.Each( constants.Any, link, handler)
                                                                                       927
                                                                                        928
                                                                                       929
                                                                                       930
   <remarks>
                                                                                                   public ulong CalculateTotalSymbolFrequencyCore(ulong symbol)
                                                                                       931
   TODO: Может потробоваться ограничение на уровень глубины рекурсии
                                                                                       932
                                                                                                      if (Options.UseSequenceMarker)
                                                                                       933
public HashSet < ulong > AllUsages (ulong link)
                                                                                       934
                                                                                                         var counter = new
                                                                                       935
  return Sync.ExecuteReadOperation(() =>
                                                                                                              TotalMarkedSequenceSymbolFrequencyOneOffCounter<ulong>(Links,
                                                                                                             Options.MarkedSequenceMatcher, symbol);
     var usages = new HashSet < ulong > ();
                                                                                                         return counter.Count();
                                                                                        936
     AllUsagesCore(link, usages);
                                                                                       937
     return usages;
                                                                                                      else
                                                                                        938
                                                                                       939
                                                                                                         var counter = new
                                                                                       940
                                                                                                         → TotalSequenceSymbolFrequencyOneOffCounter<ulong>(Links, symbol);
// При сборе всех использований (последовательностей) можно сохранять
                                                                                                         return counter.Count();
                                                                                       941
→ обратный путь к той связи с которой начинался поиск (STTTSSSTT).
                                                                                       942
// причём достаточно одного бита для хранения перехода влево или вправо
                                                                                       943
private void AllUsagesCore(ulong link, HashSet<ulong> usages)
                                                                                       944
                                                                                                   private bool AllUsagesCore1(ulong link, HashSet < ulong > usages, Func < ulong, bool >
                                                                                       945
  bool handler(ulong doublet)
                                                                                                       outerHandler)
                                                                                       946
       (usages.Add(doublet))
                                                                                                      bool handler(ulong doublet)
                                                                                       947
                                                                                       948
         AllUsagesCore(doublet, usages);
                                                                                                         if (usages.Add(doublet))
                                                                                        949
                                                                                       950
     return true:
                                                                                                            if (!outerHandler(doublet))
                                                                                        951
                                                                                       952
  Links, Unsvnc, Each (link, constants, Anv. handler):
                                                                                                               return false:
                                                                                        953
  Links. Unsync. Each (constants. Any, link, handler);
                                                                                       954
                                                                                                              (!AllUsagesCore1(doublet, usages, outerHandler))
                                                                                        955
                                                                                       956
public HashSet < ulong > AllBottomUsages(ulong link)
                                                                                                               return false;
                                                                                       957
                                                                                       958
  return Sync.ExecuteReadOperation(() =>
                                                                                        959
                                                                                                         return true:
                                                                                       960
     var visits = new HashSet < ulong > ();
                                                                                       961
     var usages = new HashSet < ulong > ();
```

849

850

852

853

856

860

865

867

869

870

871

872

873

874

875

876

877

878

879

880

882

883

884

885

889

890

892

893

894

895

897

898

899

900

901

902

903

```
return Links. Unsync. Each(link, constants. Any, handler)
                                                                                                          public void Calculate() => links.Each( constants.Any, constants.Any,
                                                                                          1025
      && Links. Unsync. Each (constants. Any, link, handler);
                                                                                                           → CalculateCore):
                                                                                          1026
                                                                                                          private bool IsElement(ulong link)
                                                                                          1027
public void CalculateAllUsages(ulong totals)
                                                                                          1028
                                                                                                              // linksInSequence.Contains(link)
                                                                                          1029
  var calculator = new AllUsagesCalculator(Links, totals);
                                                                                                             return links.Unsync.GetTarget(link) == link || links.Unsync.GetSource(link)
                                                                                          1030
  calculator.Calculate();
                                                                                                              \Rightarrow == link:
                                                                                          1031
                                                                                          1032
public void CalculateAllUsages2(ulong[] totals)
                                                                                          1033
                                                                                                          private bool CalculateCore(ulong link)
                                                                                          1034
  var calculator = new AllUsagesCalculator2(Links, totals);
                                                                                                                TODO: Проработать защиту от зацикливания
                                                                                          1035
  calculator.Calculate():
                                                                                                                Основано на SequenceWalker.WalkLeft
                                                                                          1036
                                                                                                             Func<ulong, ulong> getSource = _links.Unsync.GetSource; Func<ulong, ulong> getTarget = _links.Unsync.GetTarget;
                                                                                          1037
                                                                                          1038
private class AllUsagesCalculator
                                                                                                             Func<ulong, bool> isElement = IsElement;
                                                                                          1039
                                                                                                             void visitLeaf(ulong parent)
                                                                                          1040
  private readonly SynchronizedLinks<ulong> links;
                                                                                          1041
  private readonly ulong totals;
                                                                                                                if (link!= parent)
                                                                                          1042
                                                                                          1043
  public AllUsagesCalculator(SynchronizedLinks<ulong> links, ulong[] totals)
                                                                                                                    totals[parent]++;
                                                                                          1044
                                                                                          1045
       links = links;
                                                                                          1046
      -totals = totals;
                                                                                                             void visitNode(ulong parent)
                                                                                          1047
                                                                                          1048
                                                                                                                if (link!= parent)
                                                                                          1049
  public void Calculate() => links.Each( constants.Any, constants.Any,
                                                                                          1050

→ CalculateCore):

                                                                                          1051
                                                                                                                    totals[parent]++;
                                                                                          1052
   private bool CalculateCore(ulong link)
                                                                                          1053
                                                                                                             var stack = new Stack()
                                                                                          1054
     if (\text{totals}|\text{link}| == 0)
                                                                                                             var element = link;
                                                                                          1055
                                                                                                             if (isElement(element))
                                                                                          1056
         var total = 1UL;
                                                                                          1057
          totals[link] = total:
                                                                                                                visitLeaf(element);
                                                                                          1058
         var\ visitedChildren = new\ HashSet < ulong > ():
                                                                                          1059
         bool linkCalculator(ulong child)
                                                                                                             else
                                                                                          1060
                                                                                          1061
            if (link! = child && visitedChildren.Add(child))
                                                                                                                while (true)
                                                                                          1062
                                                                                          1063
               total += totals[child] == 0 ? 1 : totals[child];
                                                                                                                    if (isElement(element))
                                                                                          1064
                                                                                          1065
            return true;
                                                                                          1066
                                                                                                                       if (stack.Count == 0)
           links.Unsync.Each(link, constants.Any, linkCalculator);
                                                                                          1067
                                                                                                                          break:
          links.Unsync.Each(constants.Any, link, linkCalculator);
                                                                                          1068
                                                                                          1069
          totals[link] = total;
                                                                                                                      element = stack.Pop():
                                                                                          1070
                                                                                                                      var source = getSource(element):
     return true;
                                                                                          1071
                                                                                                                      var target = getTarget(element);
                                                                                          1072
                                                                                                                         / Обработка элемента
                                                                                          1073
                                                                                                                      if (isElement(target))
                                                                                          1074
private class AllUsagesCalculator2
                                                                                          1075
                                                                                                                          visitLeaf(target);
                                                                                          1076
  private readonly SynchronizedLinks<ulong> links;
                                                                                          1077
  private readonly ulong totals;
                                                                                                                         (isElement(source))
                                                                                          1078
                                                                                          1079
  public AllUsagesCalculator2(SynchronizedLinks<ulong> links, ulong[] totals)
                                                                                                                          visitLeaf(source);
                                                                                          1080
                                                                                          1081
       links = links:
                                                                                                                       element = source;
                                                                                          1082
       totals = totals;
                                                                                          1083
```

```
else
                                                                                                        public AllUsagesCollector2(ILinks<ulong> links, BitString usages)
                                                                                        1148
                                                                                        1149
              stack.Push(element):
                                                                                        1150
                                                                                                            links = links:
                                                                                                            usages = usages;
                                                                                        1151
              visitNode(element);
              element = getTarget(element);
                                                                                        1152
                                                                                        1153
                                                                                                        public bool Collect(ulong link)
                                                                                        1154
                                                                                        1155
                                                                                                           if ( usages.Add((long)link))
                                                                                        1156
       totals[link]++;
                                                                                        1157
     return true:
                                                                                                               links. Each(link, constants. Any, Collect);
                                                                                        1158
                                                                                        1159
                                                                                                               links.Each( constants.Any, link, Collect);
                                                                                        1160
private class AllUsagesCollector
                                                                                                          return true:
                                                                                        1161
                                                                                        1162
  private readonly ILinks<ulong> links;
                                                                                        1163
  private readonly HashSet<ulong> usages;
                                                                                        1164
                                                                                                     private class AllUsagesIntersectingCollector
                                                                                        1165
  public AllUsagesCollector(ILinks<ulong> links, HashSet<ulong> usages)
                                                                                        1166
                                                                                                        private readonly SynchronizedLinks<ulong> links;
                                                                                        1167
                                                                                                        private readonly HashSet < ulong > intersect With;
       links = links:
                                                                                        1168
                                                                                                        private readonly HashSet < ulong > usages;
       usages = usages;
                                                                                        1169
                                                                                        1170
                                                                                                        private readonly HashSet < ulong > enter;
                                                                                        1171
                                                                                                        public AllUsagesIntersectingCollector(SynchronizedLinks<ulong> links,
  public bool Collect(ulong link)
                                                                                        1172
                                                                                                            HashSet < ulong > intersectWith, HashSet < ulong > usages)
     if ( usages.Add(link))
                                                                                        1173
                                                                                                            links = links;
                                                                                        1174
                                                                                                            intersectWith = intersectWith:
          links. Each(link, constants. Any, Collect);
                                                                                        1175
                                                                                                            usages = usages;
         links.Each( constants.Any, link, Collect);
                                                                                        1176
                                                                                                            enter = new HashSet <ulong>(); // защита от зацикливания
                                                                                        1177
     return true;
                                                                                        1178
                                                                                        1179
                                                                                                        public bool Collect(ulong link)
                                                                                        1180
                                                                                        1181
private class AllUsagesCollector1
                                                                                                           if ( enter.Add(link))
                                                                                        1182
                                                                                        1183
  private readonly ILinks<ulong> links;
                                                                                                              if ( intersectWith.Contains(link))
                                                                                        1184
  private readonly HashSet<ulong> usages;
                                                                                        1185
  private readonly ulong continue;
                                                                                                                  usages.Add(link);
                                                                                        1186
                                                                                        1187
  public AllUsagesCollector1(ILinks<ulong> links, HashSet<ulong> usages)
                                                                                                               links.Unsync.Each(link, constants.Any, Collect)
                                                                                        1188
                                                                                                               links.Unsync.Each( constants.Any, link, Collect)
                                                                                        1189
       links = links:
                                                                                        1190
       usages = usages
                                                                                                           return true;
                                                                                        1191
       continue = links.Constants.Continue;
                                                                                        1192
                                                                                        1193
                                                                                        1194
  public ulong Collect(IList<ulong> link)
                                                                                                     private void CloseInnerConnections(Action<ulong> handler, ulong left, ulong right)
                                                                                        1195
                                                                                        1196
     var linkIndex = links.GetIndex(link);
                                                                                                        TryStepLeftUp(handler, left, right);
                                                                                        1197
     if ( usages.Add(linkIndex))
                                                                                                        TryStepRightUp(handler, right, left);
                                                                                        1198
                                                                                        1199
          links.Each(Collect, constants.Any, linkIndex);
                                                                                        1200
                                                                                                    private void AllCloseConnections(Action < ulong > handler, ulong left, ulong right)
                                                                                        1201
     return continue;
                                                                                        1202
                                                                                                         Direct
                                                                                        1203
                                                                                                       if (left == right)
                                                                                        1204
                                                                                        1205
private class AllUsagesCollector2
                                                                                                           handler(left):
                                                                                        1206
                                                                                        1207
  private readonly ILinks<ulong> links;
                                                                                                       var doublet = Links.Unsync.SearchOrDefault(left, right);
                                                                                        1208
  private readonly BitString usages;
                                                                                                       if (doublet != constants.Null)
                                                                                        1209
```

1120

```
1210
                                                                                                    1260
                  handler(doublet):
                                                                                                                          throw new ArgumentLinkDoesNotExistsException<ulong>(sequence[i].
1211
                                                                                                    1261
                                                                                                                              $\patternSequence[\{i\}]");
1212
                 Inner
1213
                                                                                                    1262
               CloseInnerConnections(handler, left, right);
1214
                                                                                                    1263
                 Outer
1215
                                                                                                    1264
               StepLeft(handler, left, right);
1216
                                                                                                    1265
               StepRight(handler, left, right):
                                                                                                                  / Pattern Matching -> Key To Triggers
1217
                                                                                                    1266
               PartialStepRight(handler, left, right);
                                                                                                                 public HashSet < ulong > MatchPattern(params ulong[] patternSequence)
1218
                                                                                                    1267
1219
               PartialStepLeft(handler, left, right);
                                                                                                    1268
                                                                                                                    return Sync.ExecuteReadOperation(() =>
1220
                                                                                                    1269
1221
                                                                                                    1270
            private HashSet < ulong > GetAllPartiallyMatchingSequencesCore(ulong[] sequence,
1222
                                                                                                                       patternSequence = Simplify(patternSequence);
                                                                                                    1271
                HashSet < ulong > previousMatchings, long startAt)
                                                                                                                       if (patternSequence.Length > 0)
                                                                                                    1272
1223
                                                                                                    1273
                 (\text{startAt} > = \text{sequence.Length}) / ?
                                                                                                                          EnsureEachLinkIsAnyOrZeroOrManyOrExists(Links, patternSequence);
1224
                                                                                                    1274
1225
                                                                                                                          var uniqueSequenceElements = new HashSet < ulong > ():
                                                                                                    1275
                  return previousMatchings:
1226
                                                                                                                          for (var i = 0; i < patternSequence.Length; <math>i++)
                                                                                                    1276
1227
                                                                                                    1277
               var secondLinkUsages = new HashSet < ulong > ():
1228
                                                                                                                             if (patternSequence[i]! constants.Any && patternSequence[i]!=
                                                                                                    1278
               AllUsagesCore(sequence[startAt], secondLinkUsages);
1229
                                                                                                                                 ZeroOrMany)
               secondLinkUsages.Add(sequence[startAt]);
1230
                                                                                                    1279
               var matchings = new HashSet < ulong > ():
1231
                                                                                                                                uniqueSequenceElements.Add(patternSequence[i]);
                                                                                                    1280
                f/for (var i = 0; i < previousMatchings.Count; i++)
1232
                                                                                                    1281
               foreach (var secondLinkUsage in secondLinkUsages)
1233
                                                                                                    1282
1234
                                                                                                                          var results = new HashSet < ulong > ():
                                                                                                    1283
                  foreach (var previousMatching in previousMatchings)
1235
                                                                                                                          foreach (var uniqueSequenceElement in uniqueSequenceElements)
                                                                                                    1284
                                                                                                    1285
                     //AllCloseConnections(matchings.AddAndReturnVoid, previousMatching,
1237
                                                                                                                             AllUsagesCore(uniqueSequenceElement, results);
                                                                                                    1286

→ secondLinkUsage)

                                                                                                    1287
                     StepRight(matchings.AddAndReturnVoid, previousMatching,
                                                                                                                          var filteredResults = new HashSet < ulong > ();
1238
                                                                                                    1288

→ secondLinkUsage);

                                                                                                                          var matcher = new PatternMatcher(this, patternSequence, filteredResults);
                                                                                                    1289
                     TryStepRightUp(matchings.AddAndReturnVoid, secondLinkUsage,
                                                                                                                          matcher.AddAllPatternMatchedToResults(results);
1239
                                                                                                    1290
                                                                                                                          return filteredResults;
                         previousMatching);
                                                                                                    1291
                     //PartialStepRight(matchings.AddAndReturnVoid, secondLinkUsage,
                                                                                                    1292
1240
                                                                                                                       return new HashSet < ulong > ();
                         sequence[startAt]); // почему-то эта ошибочная запись приводит к
                                                                                                    1293
                         желаемым результам.
                                                                                                    1294
                     PartialStepRight(matchings.AddAndReturnVoid, previousMatching,
                                                                                                    1295
1241
                                                                                                    1296
                        secondLinkUsage);
                                                                                                                   Найти все возможные связи между указанным списком связей.
                                                                                                    1297
1242
                                                                                                                    Находит связи между всеми указанными связями в любом порядке.
                                                                                                    1298
1243
                                                                                                                   / TODO: решить что делать с повторами (когда одни и те же элементы
                                                                                                    1299
                 (\text{matchings.Count} == 0)
1244
                                                                                                                 → встречаются несколько раз в последовательности)
                                                                                                                 public HashSet < ulong > GetAllConnections(params ulong | linksToConnect)
                  return matchings;
                                                                                                    1300
1246
                                                                                                    1301
1247
                                                                                                                    return Sync. Execute ReadOperation(() =>
               return GetAllPartiallyMatchingSequencesCore(sequence, matchings, startAt + 1)
                                                                                                    1302
1248
                                                                                                    1303
                                                                                                                       var results = new HashSet < ulong > ();
                                                                                                    1304
1249
                                                                                                                       if (linksToConnect.Length > 0)
                                                                                                    1305
1250
            private static void
                                                                                                    1306
1251
                                                                                                                          Links.EnsureEachLinkExists(linksToConnect);
                 EnsureEachLinkIsAnyOrZeroOrManyOrExists(SynchronizedLinks<ulong> links,
                                                                                                    1307
                                                                                                                          AllUsagesCore(linksToConnect[0], results);
                                                                                                    1308
                 params ulong | sequence)
                                                                                                                          for (var i = 1; i < linksToConnect.Length; <math>i++)
                                                                                                    1309
1252
                                                                                                    1310
                 (\text{sequence} == \text{null})
1253
                                                                                                                             var next = new HashSet < ulong >();
                                                                                                    1311
1254
                                                                                                                             AllUsagesCore(linksToConnect[i], next);
                  return;
                                                                                                    1312
1255
                                                                                                                             results.IntersectWith(next);
                                                                                                    1313
1256
               for (var i = 0; i < sequence.Length; i++)
                                                                                                    1314
1257
                                                                                                    1315
1258
                                                                                                                       return results;
                  if (sequence[i] != constants.Any && sequence[i] != ZeroOrMany &&
                                                                                                    1316
1259
                                                                                                    1317
                      !links.Exists(sequence[i]))
```

```
1379
                                                                                                                  var next = new BitString((long)Links.Unsync.Count() + 1); //new
                                                                                        1380
public HashSet < ulong > Get AllConnections1 (params ulong | linksToConnect)
                                                                                                                  \rightarrow BitArray((int) links.Total + 1):
                                                                                                                  var collector = new AllUsagesCollector2(Links.Unsvnc. next);
                                                                                        1381
  return Sync.ExecuteReadOperation(() =>
                                                                                                                  collector.Collect(linksToConnect[i]);
                                                                                        1382
                                                                                                                 results = results.And(next);
                                                                                        1383
     var results = new HashSet < ulong > ();
                                                                                        1384
     if (linksToConnect.Length > 0)
                                                                                        1385
                                                                                                           return results.GetSetUInt64Indices();
                                                                                        1386
         Links.EnsureEachLinkExists(linksToConnect):
                                                                                        1387
         var collector1 = new AllUsagesCollector(Links.Unsync, results);
                                                                                        1388
         collector1.Collect(linksToConnect[0]);
                                                                                        1389
         var next = new HashSet < ulong > ():
                                                                                                     private static ulong [Simplify(ulong sequence)
                                                                                        1390
         for (var i = 1; i < linksToConnect.Length; <math>i++)
                                                                                        1391
                                                                                                          / Считаем новый размер последовательности
                                                                                        1392
            var collector = new AllUsagesCollector(Links.Unsync, next);
                                                                                                        long newLength = 0;
                                                                                        1393
            collector.Collect(linksToConnect[i]);
                                                                                                        var zeroOrManyStepped = false;
                                                                                        1394
            results.IntersectWith(next);
                                                                                                        for (var i = 0; i < \text{sequence.Length}; i++)
                                                                                         1395
            next.Clear();
                                                                                        1396
                                                                                                           if (\text{sequence}|i| == \text{ZeroOrMany})
                                                                                        1397
                                                                                        1398
     return results:
                                                                                                               if (zeroOrManyStepped)
                                                                                        1399
                                                                                        1400
                                                                                                                  continue;
                                                                                        1401
                                                                                        1402
public HashSet < ulong > GetAllConnections2(params ulong | linksToConnect)
                                                                                                              zeroOrManyStepped = true;
                                                                                        1403
                                                                                        1404
  return Sync.ExecuteReadOperation(() =>
                                                                                                           else
                                                                                        1405
                                                                                        1406
     var results = new HashSet < ulong >();
                                                                                                               //if (zeroOrManyStepped) Is it efficient?
                                                                                        1407
     if (linksToConnect.Length > 0)
                                                                                                              zeroOrManyStepped = false;
                                                                                        1408
                                                                                        1409
                                                                                                           newLength++;
         Links.EnsureEachLinkExists(linksToConnect);
                                                                                        1410
         var collector1 = new AllUsagesCollector(Links, results);
                                                                                        1411
         collector1.Collect(linksToConnect[0]);
                                                                                                           Строим новую последовательность
                                                                                        1412
                                                                                                        zeroOrManvStepped = false:
          //AllUsagesCore(linksToConnect[0], results);
                                                                                        1413
                                                                                                        var newSequence = new ulong[newLength];
         for (var i = 1; i < linksToConnect.Length; <math>i++)
                                                                                        1414
                                                                                                        long \mathbf{j} = 0;
                                                                                        1415
                                                                                                        for (var i = 0; i < \text{sequence.Length}; i++)
            var next = new HashSet < ulong > ():
                                                                                        1416
            var collector = new AllUsagesIntersectingCollector(Links, results, next);
                                                                                        1417
                                                                                                             /var current = zeroOrManyStepped;
                                                                                        1418
            collector.Collect(linksToConnect[i]);
                                                                                                             zeroOrManyStepped = patternSequence[i] == zeroOrMany;
            //AllUsagesCore(linksToConnect[i], next);
                                                                                        1419
                                                                                                             /if (current && zeroOrManyStepped)
            //results.IntersectWith(next);
                                                                                        1420
                                                                                                                continue:
                                                                                        1421
            results = next;
                                                                                                             /var newZeroOrManyStepped = patternSequence[i] == zeroOrMany;
                                                                                        1422
                                                                                                             /if (zeroOrManyStepped && newZeroOrManyStepped)
                                                                                        1423
     return results:
                                                                                        1424
                                                                                                             /zeroOrManyStepped = newZeroOrManyStepped;
                                                                                        1425
                                                                                                            if (sequence|i| == ZeroOrMany)
                                                                                        1426
                                                                                        1427
public List < ulong > GetAllConnections3(params ulong | linksToConnect)
                                                                                                               if (zeroOrManyStepped)
                                                                                        1428
                                                                                        1429
  return Sync. Execute ReadOperation(() =>
                                                                                                                  continue;
                                                                                        1430
                                                                                        1431
     var results = new BitString((long)Links.Unsync.Count() + 1); // new
                                                                                                              zeroOrManyStepped = true;
                                                                                        1432
         BitArray((int) links.Total + 1);
                                                                                        1433
     if (linksToConnect.\overline{L}ength > 0)
                                                                                        1434
                                                                                        1435
                                                                                                                /if (zeroOrManyStepped) Is it efficient?
                                                                                        1436
         Links.EnsureEachLinkExists(linksToConnect);
                                                                                                              zeroOrManyStepped = false;
                                                                                        1437
         var collector1 = new AllUsagesCollector2(Links.Unsync, results);
                                                                                        1438
         collector1.Collect(linksToConnect[0]);
                                                                                                           newSequence[i++] = sequence[i];
                                                                                        1439
         for (var i = 1; i < linksToConnect.Length; <math>i++)
```

```
1440
                                                                                                         1499
                return newSequence:
                                                                                                                               for (var i = elements.Length - 1; i >= 0; i--)
1441
                                                                                                         1500
1442
                                                                                                         1501
1443
                                                                                                                                  var element = elements[i];
                                                                                                         1502
             public static void TestSimplify()
1444
                                                                                                                                  if (element != 0)
                                                                                                         1503
1445
                                                                                                         1504
                var sequence = new ulong[] { ZeroOrMany, ZeroOrMany, 2, 3, 4, ZeroOrMany,
1446
                                                                                                                                     results. Add(element);
                                                                                                         1505

→ ZeroOrMany, ZeroOrMany, 4, ZeroOrMany, ZeroOrMany, ZeroOrMany };

                                                                                                         1506
                var simplifiedSequence = Simplify(sequence):
1447
                                                                                                         1507
1448
                                                                                                         1508
1449
                                                                                                         1509
             public List < ulong > GetSimilarSequences() => new List < ulong >();
1450
                                                                                                                         else
                                                                                                         1510
1451
                                                                                                         1511
             public void Prediction()
1452
                                                                                                                            var nextRightLink = middleLinks[rightBound];
                                                                                                         1512
1453
                                                                                                                            var elements = GetLeftElements(rightLink, nextRightLink);
                                                                                                         1513
                 // links
1454
                                                                                                         1514
                                                                                                                            if (leftBound <= rightBound)
1455
                 /sequences
                                                                                                         1515
1456
                                                                                                                               for (var i = elements.Length - 1; i >= 0; i--)
                                                                                                         1516
1457
                                                                                                         1517
             #region From Triplets
1458
                                                                                                                                  var element = elements[i];
                                                                                                         1518
1459
                                                                                                                                  if (element != 0)
                                                                                                         1519
             //public static void DeleteSequence(Link sequence)
1460
                                                                                                         1520
1461
                                                                                                                                     CollectMatchingSequences(leftLink, leftBound, middleLinks, elements[i]
                                                                                                         1521
1462
                                                                                                                                         rightBound - 1, ref results):
1463
             public List < ulong > Collect Matching Sequences (ulong | links)
1464
                                                                                                         1522
                                                                                                         1523
1465
                if (links.Length == 1)
1466
                                                                                                         1524
                                                                                                         1525
1467
                   throw new Exception ("Подпоследовательности с одним элементом не
                                                                                                         1526
1468
                                                                                                                               for (var i = elements.Length - 1; i >= 0; i--)
                                                                                                         1527
                       поддерживаются.");
                                                                                                         1528
1469
                var leftBound = 0;
                                                                                                                                  var element = elements[i]:
                                                                                                        1529
1470
                var rightBound = links.Length - 1;
                                                                                                                                  if (element != 0)
                                                                                                         1530
1471
                var left = links[leftBound++];
                                                                                                         1531
1472
                                                                                                                                     results.Add(element);
                var right = links[rightBound--]:
                                                                                                         1532
1473
                var results = new List < ulong > ();
1474
                                                                                                         1533
                Collect Matching Sequences (left, left Bound, links, right, right Bound, ref results);
                                                                                                         1534
1475
                return results:
                                                                                                         1535
                                                                                                         1536
1477
1478
                                                                                                         1537
             private void CollectMatchingSequences(ulong leftLink, int leftBound, ulong[]
1479
                                                                                                         1538
                                                                                                                      public ulong[] GetRightElements(ulong startLink, ulong rightLink)
                 middleLinks, ulong rightLink, int rightBound, ref List<ulong> results)
                                                                                                         1539
                                                                                                         1540
1480
                var leftLinkTotalReferers = Links.Unsync.Count(leftLink);
                                                                                                                         var result = new ulong[5];
                                                                                                         1541
1481
                                                                                                                         TryStepRight(startLink, rightLink, result, 0):
                var rightLinkTotalReferers = Links.Unsync.Count(rightLink);
                                                                                                         1542
1482
                if (leftLinkTotalReferers <= rightLinkTotalReferers)
                                                                                                                         Links.Each( constants.Any, startLink, couple =>
                                                                                                         1543
1483
                                                                                                         1544
1484
                                                                                                                            if (couple != startLink)
                   var nextLeftLink = middleLinks[leftBound];
                                                                                                         1545
1485
                   var elements = GetRightElements(leftLink, nextLeftLink);
                                                                                                         1546
                                                                                                                               if (TryStepRight(couple, rightLink, result, 2))
                   if (leftBound <= rightBound)
                                                                                                         1547
1487
                                                                                                         1548
                                                                                                                                  return false;
                      for (var i = elements.Length - 1; i >= 0; i--)
                                                                                                         1549
1489
                                                                                                         1550
                         var element = elements[i];
                                                                                                         1551
1491
                                                                                                                            return true;
                         if (element != 0)
                                                                                                         1552
1492
                                                                                                         1553
1493
                                                                                                                           (Links.GetTarget(Links.GetTarget(startLink)) == rightLink)
                                                                                                         1554
                            Collect Matching Sequences (element, left Bound + 1, middle Links)
1494
                                                                                                         1555
                                 rightLink, rightBound, ref results);
                                                                                                                            result |4| = start Link;
                                                                                                         1556
1495
                                                                                                         1557
1496
                                                                                                                         return result:
                                                                                                         1558
1497
                   else
1498
```

```
if (coupleSource == leftLink)
1559
                                                                                                       1621
1560
                                                                                                       1622
            public bool TryStepRight(ulong startLink, ulong rightLink, ulong[] result, int offset)
                                                                                                                                result[offset] = couple;
                                                                                                       1623
1562
                                                                                                                                if (++added == 2)
                                                                                                       1624
               var added = 0:
1563
                                                                                                       1625
               Links.Each(startLink, constants.Any, couple =>
1564
                                                                                                                                   return false;
                                                                                                       1626
1565
                                                                                                       1627
                  if (couple != startLink)
1566
                                                                                                       1628
1567
                                                                                                                             else if (Links.GetTarget(coupleSource) == leftLink) // coupleSource.Linker
                                                                                                       1629
                      var coupleTarget = Links.GetTarget(couple):
1568
                                                                                                                                  == Net.And &&
                      if (coupleTarget == rightLink)
                                                                                                       1630
1570
                                                                                                                                result[offset + 1] = couple;
                                                                                                       1631
                         result[offset] = couple;
                                                                                                                                if (++added == 2)
                                                                                                       1632
                         if (++added == 2)
1572
                                                                                                       1633
                                                                                                                                   return false;
                                                                                                       1634
                           return false;
1574
                                                                                                       1635
                                                                                                       1636
                                                                                                       1637
                      else if (Links.GetSource(coupleTarget) == rightLink) // coupleTarget.Linker
                                                                                                                          return true;
1577
                                                                                                       1638
                          == Net.And &&
                                                                                                       1639
                                                                                                       1640
                                                                                                                       return added > 0;
1578
                         result [offset +1] = couple;
                                                                                                       1641
1579
                         if (++added == 2)
                                                                                                       1642
1580
                                                                                                                    #endregion
                                                                                                       1643
1581
                                                                                                       1644
                           return false;
1582
                                                                                                                    #region Walkers
                                                                                                       1645
                                                                                                       1646
1584
                                                                                                                    public class PatternMatcher: RightSequenceWalker<ulong>
                                                                                                       1647
1585
                                                                                                       1648
                  return true;
1586
                                                                                                       1649
                                                                                                                       private readonly Sequences sequences;
1587
                                                                                                                       private readonly ulong patternSequence;
                                                                                                       1650
               return added > 0:
                                                                                                                       private readonly HashSet < LinkIndex > linksInSequence;
                                                                                                       1651
1589
                                                                                                       1652
                                                                                                                       private readonly HashSet < LinkIndex > results;
1590
                                                                                                       1653
            public ulong GetLeftElements (ulong startLink, ulong leftLink)
1591
                                                                                                                       #region Pattern Match
                                                                                                       1654
1592
                                                                                                       1655
               var result = new ulong[5]
1593
                                                                                                                       enum PatternBlockType
                                                                                                       1656
               TryStepLeft(startLink, leftLink, result, 0);
1594
                                                                                                       1657
               Links.Each(startLink, constants.Any, couple =>
                                                                                                                          Undefined,
1595
                                                                                                       1658
                                                                                                                          Gap,
1596
                                                                                                       1659
                                                                                                                          Elements
                  if (couple != startLink)
1597
                                                                                                       1660
                                                                                                       1661
1598
                      if (TryStepLeft(couple, leftLink, result, 2))
                                                                                                       1662
1599
                                                                                                                       struct PatternBlock
                                                                                                       1663
                                                                                                       1664
                         return false;
1601
                                                                                                                          public PatternBlockType Type;
                                                                                                       1665
1602
                                                                                                                          public long Start;
                                                                                                       1666
1603
                                                                                                                          public long Stop;
                                                                                                       1667
                  return true;
1604
                                                                                                       1668
1605
                                                                                                       1669
                  (Links.GetSource(Links.GetSource(leftLink)) == startLink)
1606
                                                                                                                       private readonly List<PatternBlock> pattern;
                                                                                                       1670
1607
                                                                                                                       private int patternPosition;
                                                                                                       1671
                  result[4] = leftLink;
1608
                                                                                                       1672
                                                                                                                       private long sequencePosition;
                                                                                                       1673
               return result:
1610
                                                                                                                       #endregion
                                                                                                       1674
1611
                                                                                                       1675
1612
                                                                                                                       public PatternMatcher(Sequences sequences, LinkIndex[] patternSequence,
                                                                                                       1676
            public bool TryStepLeft(ulong startLink, ulong leftLink, ulong | result, int offset)
1613
                                                                                                                          HashSet < LinkIndex > results)
1614
                                                                                                                          : base(sequences.Links.Unsync)
                                                                                                       1677
               var added = 0;
                                                                                                       1678
               Links.Each( constants.Any, startLink, couple =>
1616
                                                                                                                            sequences = sequences;
                                                                                                       1679
1617
                                                                                                                            patternSequence = patternSequence;
                                                                                                       1680
                  if (couple != startLink)
1618
                                                                                                                            linksInSequence = new HashSet < linkIndex > (patternSequence.Where(x = > x))
                                                                                                       1681
1619
                                                                                                                              != constants.Any && x != ZeroOrMany));
                      var coupleSource = Links.GetSource(couple);
1620
```

```
pattern.Add(patternBlock);
   results = results;
                                                                                     1743
   _pattern = CreateDetailedPattern();
                                                                                                                 patternBlock = new PatternBlock
                                                                                     1744
                                                                                     1745
                                                                                                                     Type = PatternBlockType.Gap,
                                                                                     1746
protected override bool IsElement(IList<ulong> link) =>
                                                                                                                    Start = 0.
                                                                                     1747
                                                                                                                    Stop = long.MaxValue
→ linksInSequence.Contains(Links.GetIndex(link)) || base.IsElement(link);
                                                                                     1748
                                                                                     1749
public bool PatternMatch(LinkIndex sequenceToMatch)
                                                                                     1750
                                                                                     1751
                                                                                                              else
   patternPosition = 0:
                                                                                     1752
    \overline{\text{sequencePosition}} = 0:
                                                                                                                 patternBlock.Stop = i;
                                                                                     1753
   foreach (var part in Walk(sequenceToMatch))
                                                                                     1754
                                                                                     1755
      if (!PatternMatchCore(Links.GetIndex(part)))
                                                                                                           else // patternBlock.Type == PatternBlockType.Gap
                                                                                     1757
                                                                                                                ( patternSequence[i] == constants.Any)
         break:
                                                                                     1758
                                                                                     1759
                                                                                                                 patternBlock.Start++;
                                                                                     1760
  return patternPosition == pattern.Count || ( patternPosition ==
                                                                                                                 if (patternBlock.Stop < patternBlock.Start)
                                                                                     1761
        pattern.Count - 1 & \& pattern[patternPosition].Start == 0);
                                                                                     1762
                                                                                                                    patternBlock.Stop = patternBlock.Start;
                                                                                     1763
                                                                                     1764
private List<PatternBlock> CreateDetailedPattern()
                                                                                     1765
                                                                                                              else if ( patternSequence[i] == ZeroOrMany)
                                                                                     1766
  var pattern = new List<PatternBlock>();
                                                                                     1767
                                                                                                                 patternBlock.Stop = long.MaxValue;
  var patternBlock = new PatternBlock();
                                                                                     1768
  for (\text{var } i = 0; i < \text{patternSequence.Length}; i++)
                                                                                     1769
                                                                                                              else
                                                                                     1770
      if (patternBlock.Type == PatternBlockType.Undefined)
                                                                                     1771
                                                                                                                 pattern.Add(patternBlock);
                                                                                     1772
                                                                                                                 patternBlock = new PatternBlock
                                                                                     1773
         if ( patternSequence |i| = constants.Any)
                                                                                     1774
                                                                                                                     Type = PatternBlockType.Elements,
            patternBlock.Type = PatternBlockType.Gap;
                                                                                     1776
                                                                                                                    Start = i.
            patternBlock.Start = 1;
                                                                                                                    Stop = i
                                                                                     1777
            patternBlock.Stop = 1;
                                                                                     1778
         else if ( patternSequence[i] == ZeroOrMany)
                                                                                     1779
                                                                                     1780
            patternBlock.Type = PatternBlockType.Gap;
                                                                                     1781
                                                                                                         if (patternBlock.Type != PatternBlockType.Undefined)
            patternBlock.Start = 0;
                                                                                     1782
            patternBlock.Stop = long.MaxValue;
                                                                                     1783
                                                                                                           pattern.Add(patternBlock);
                                                                                     1784
                                                                                     1785
                                                                                                        return pattern;
                                                                                     1786
            patternBlock.Type = PatternBlockType.Elements;
                                                                                     1787
                                                                                     1788
            patternBlock.Start = i;
                                                                                                       //* match: search for regexp anywhere in text */
            patternBlock.Stop = i;
                                                                                     1789
                                                                                                       /int match(char* regexp, char* text)
                                                                                     1790
                                                                                     1791
                                                                                                          do
      else if (patternBlock.Type == PatternBlockType.Elements)
                                                                                     1792
                                                                                     1793
                                                                                                            while (*text++ != ' \setminus 0');
         if (patternSequence[i] == constants.Any)
                                                                                     1794
                                                                                     1795
                                                                                                          return 0:
            pattern.Add(patternBlock);
                                                                                     1796
            patternBlock = new PatternBlock
                                                                                     1797
                                                                                                      //* matchhere: search for regexp at beginning of text */
                                                                                     1798
                                                                                                       /int matchhere(char* regexp, char* text)
               Type = PatternBlockType.Gap,
                                                                                     1799
               Start = 1.
                                                                                     1800
                                                                                                          if (\operatorname{regexp}[0] == ' \setminus 0')
               Stop = 1
                                                                                     1801
                                                                                     1802
                                                                                                             return 1
                                                                                                          if (\operatorname{regexp}[1] == "*")
                                                                                     1803
                                                                                                             return matchstar(regexp[0], regexp + 2, text);
         else if ( patternSequence|i| == ZeroOrMany)
                                                                                     1804
```

1728

 $1741 \\ 1742$

```
if (regexp[0] == '\$' \&\& regexp[1] == '\setminus 0')
                                                                                                                           else
                                                                                                     1865
1805
                       return *text == 1 \cdot 0:
1806
                                                                                                     1866
                    if (*text != ' \ 0' \&\& (regexp[0] == '.' || regexp[0] == *text))
                                                                                                                                ( sequencePosition > currentPatternBlock.Stop)
                                                                                                     1867
1807
                       return matchhere (regexp + 1, text + 1);
1808
                                                                                                     1868
                    return 0;
                                                                                                                                return false; // Соответствие невозможно
1809
                                                                                                     1869
                                                                                                     1870
1810
1811
                                                                                                                              var nextPatternBlock = pattern[patternPosition + 1];
                                                                                                     1871
                ///* matchstar: search for c*regexp at beginning of text */
1812
                                                                                                                              if ( patternSequence[nextPatternBlock.Start] == element)
                                                                                                     1872
                //int matchstar(int c, char* regexp, char* text)
1813
                                                                                                     1873
                                                                                                                                 if (nextPatternBlock.Start < nextPatternBlock.Stop)
1814
                                                                                                     1874
1815
                                                                                                     1875
                         /* a * matches zero or more instances */
                                                                                                                                    patternPosition++;
1816
                                                                                                     1876
1817
                       if (matchhere(regexp, text))
                                                                                                     1877
                                                                                                                                    \overline{\phantom{a}} sequence Position = 1;
                          return 1;
                                                                                                     1878
                     else
1819
                                                                                                     1879
                    return 0:
1820
                                                                                                     1880
                                                                                                                                     patternPosition += 2:
                                                                                                     1881
1891
                                                                                                                                    sequencePosition = 0:
1822
                                                                                                     1882
               //private void GetNextPatternElement(out LinkIndex element, out long
1823
                                                                                                     1883
                   mininumGap, out long maximumGap)
                                                                                                     1884
                                                                                                     1885
1824
                    mininumGap = 0:
                                                                                                     1886
1825
                                                                                                                             // currentPatternBlock.Type == PatternBlockType.Elements
                    maximumGap = 0:
                                                                                                     1887
1826
                    element = 0:
                                                                                                     1888
1827
                                                                                                                         var patternElementPosition = currentPatternBlock.Start + sequencePosition;
                    for (; patternPosition < patternSequence.Length; patternPosition++)
                                                                                                     1889
                                                                                                                          if ( patternSequence[patternElementPosition] != element)
                                                                                                     1890
1829
                                                                                                     1891
                       if (patternSequence| patternPosition| == Doublets.Links.Null)
1830
                                                                                                                              return false: // Соответствие невозможно
                           mininumGap++;
                                                                                                     1892
1831
                       else if ( patternSequence[ patternPosition] == ZeroOrMany)
                                                                                                     1893
1832
                                                                                                                             (patternElementPosition == currentPatternBlock.Stop)
                          maximumGap = long.MaxValue;
                                                                                                     1894
1833
                       else
                                                                                                     1895
                                                                                                                              patternPosition++;
                                                                                                     1896
                          break;
1835
                                                                                                                               sequencePosition = 0;
                                                                                                     1897
1836
                                                                                                     1898
1837
                                                                                                                          else
                                                                                                     1899
                    if (maximumGap < mininumGap)
1838
                                                                                                     1900
                       maximumGap = mininumGap;
1839
                                                                                                                               sequencePosition++;
                                                                                                     1901
1840
                                                                                                     1902
1841
               private bool PatternMatchCore(LinkIndex element)
                                                                                                     1903
1842
                                                                                                                       return true:
                                                                                                     1904
1843
                                                                                                                        //if ( patternSequence[ patternPosition] != element)
                                                                                                     1905
                  if ( patternPosition >= pattern.Count)
1844
                                                                                                                            return false;
                                                                                                     1906
1845
                                                                                                                         /else
                       patternPosition = -2;
                                                                                                     1907
1846
                     return false:
                                                                                                     1908
1847
                                                                                                                              sequencePosition++;
                                                                                                     1909
1848
                                                                                                                              patternPosition++;
                  var currentPatternBlock = pattern[ patternPosition];
                                                                                                     1910
1849
                                                                                                                             return true:
                  if (currentPatternBlock.Type == PatternBlockType.Gap)
                                                                                                     1911
                                                                                                     1912
1851
                                                                                                     1913
                      //var currentMatchingBlockLength = ( sequencePosition -
                                                                                                                         / 	ext{if ( filterPosition ==  patternSequence.Length)}
                                                                                                     1914
                            lastMatchedBlockPosition)
                                                                                                     1915
                     if ( sequencePosition < currentPatternBlock.Start)
1853
                                                                                                                              filterPosition = -2; // Длиннее чем нужно
                                                                                                     1916
1854
                                                                                                                             return false;
                          sequencePosition++;
                                                                                                     1917
1855
                        return true; // Двигаемся дальше
                                                                                                     1918
                                                                                                                         / if (element != patternSequence | filterPosition |)
                                                                                                     1919
1857
                                                                                                     1920
                        Это последний блок
                                                                                                                              filterPosition = -1;
                         pattern.Count == patternPosition + 1)
                                                                                                     1921
                                                                                                                             return false; // Начинается иначе
                                                                                                     1922
1860
                         patternPosition++;
                                                                                                     1923
1861
                                                                                                                           filterPosition++;
                          sequencePosition = 0;
                                                                                                     1924
1862
                                                                                                                         \overline{\text{if}} ( filterPosition == ( patternSequence.Length - 1))
                        return false; // Полное соответствие
1863
                                                                                                                            return false;
                                                                                                     1926
1864
```

```
'/if (filterPosition >= 0)
                                                                                                           #endif
1927
                                                                                                       32
                                                                                                                       hasElements = false:
                                                                                                       33
1928
                                                                                                                        for (var i = 0: i < array.Length: i++)
                       if (element == patternSequence filterPosition + 1)
                                                                                                       34
1929
                            filterPosition++;
1930
                                                                                                       35
                                                                                                                          var candidate = arrav[i]:
                                                                                                       36
1931
                                                                                                                          if (candidate == 0)
                          return false:
                                                                                                       37
1932
1933
                                                                                                       38
                                                                                                                              continue:
                      ( filterPosition < 0)
                                                                                                       39
1934
1935
                                                                                                                          var doubletOffset = i * 2;
                       if (element == patternSequence[0])
1936
                                                                                                       41
                            filterPosition = 0;
                                                                                                                          if (isElement(candidate))
                                                                                                       42
1937
                                                                                                       43
1938
                                                                                                                              nextArray[doubletOffset] = candidate;
                                                                                                       44
1939
                                                                                                       45
               public void AddAllPatternMatchedToResults(IEnumerable<ulong>
1941
                                                                                                       46
                    sequencesToMatch)
                                                                                                       47
                                                                                                                              var link = links.GetLink(candidate):
                                                                                                       48
1942
                                                                                                                              var linkSource = links.GetSource(link);
                  foreach (var sequenceToMatch in sequencesToMatch)
                                                                                                       49
1943
                                                                                                                              var linkTarget = links.GetTarget(link);
                                                                                                       50
1944
                                                                                                                              nextArray[doubletOffset] = linkSource;
                     if (PatternMatch(sequenceToMatch))
1945
                                                                                                                              nextArray[doubletOffset + 1] = linkTarget;
                                                                                                       52
1946
                                                                                                                              if (!hasElements)
                         _results.Add(sequenceToMatch);
                                                                                                       53
1947
                                                                                                       54
1948
                                                                                                                                hasElements = !(isElement(linkSource) && isElement(linkTarget));
1949
1950
                                                                                                       56
                                                                                                       57
1951
1952
                                                                                                           #if USEARRAYPOOL
            #endregion
1953
                                                                                                                        if (array.Length > 1)
1954
                                                                                                       60
1955
                                                                                                       61
                                                                                                                          ArrayPool.Free(array);
                                                                                                       62
                                                                                                       63
 ./Sequences/Sequences.Experiments.ReadSequence.cs
                                                                                                           #endif
                                                                                                      64
        #define USEARRAYPOOL
                                                                                                                       array = nextArray;
                                                                                                       65
      using System;
                                                                                                       66
      using System Runtime Compiler Services;
                                                                                                                     while (hasElements):
      #if USEARRAYPOOL
                                                                                                                     var filledElementsCount = CountFilledElements(array);
                                                                                                       68
      using Platform.Collections;
                                                                                                                     if (filledElementsCount == array.Length)
                                                                                                       70
                                                                                                                       return array;
                                                                                                       71
      namespace Platform.Data.Doublets.Sequences
                                                                                                                     else
                                                                                                       73
         partial class Sequences
                                                                                                       74
                                                                                                                       return CopyFilledElements(array, filledElementsCount);
                                                                                                       75
            public ulong ReadSequenceCore(ulong sequence, Func<ulong, bool> isElement)
                                                                                                       77
               var links = Links.Unsync;
                                                                                                       78
               var length = 1;
                                                                                                                  [MethodImpl(MethodImplOptions.AggressiveInlining)]
                                                                                                       79
               var array = new ulong[length];
                                                                                                                 private static ulong[] CopyFilledElements(ulong[] array, int filledElementsCount)
                                                                                                       80
               \operatorname{array}|0| = \operatorname{sequence};
                                                                                                       81
                                                                                                                    var finalArray = new ulong[filledElementsCount];
                                                                                                       82
                 (isElement(sequence))
                                                                                                                     for (int i = 0, j = 0; i < array.Length; i++)
                                                                                                       83
                  return array;
                                                                                                       84
 2.1
                                                                                                                        if (array[i] > 0)
                                                                                                       85
 22
 23
               bool hasElements;
 24
                                                                                                                          finalArray[j] = array[i];
                                                                                                       87
               do
 25
                  length *= 2:
      #if USEARRAYPOOL
                                                                                                           #if USEARRAYPOOL
 28
                                                                                                       91
                  var nextArray = ArrayPool.Allocate<ulong>(length);
                                                                                                                       ArrayPool.Free(array);
 29
                                                                                                      92
      #else
                                                                                                           #endif
 30
                                                                                                       93
                  var nextArray = new ulong[length];
 31
```

```
return finalArray;
                                                                                                     21
                                                                                                                   </summary>
                                                                                                                    <param name="sequence">Последовательность для индексации.
                                                                                                     22
95
96
                                                                                                     23
           [MethodImpl(MethodImplOptions, AggressiveInlining)]
97
                                                                                                                    True если последовательность уже была проиндексирована ранее и
                                                                                                     24
           private static int CountFilledElements(ulong array)
98
                                                                                                                   False если последовательность была проиндексирована только что.
                                                                                                     25
99
                                                                                                     26
              var count = 0:
100
                                                                                                                public bool Index(TLink[] sequence)
                                                                                                     27
              for (var i = 0; i < array.Length; i++)
101
                                                                                                     28
102
                                                                                                                   var indexed = true;
                                                                                                     20
                 if (array[i] > 0)
103
                                                                                                                   var i = sequence. Length:
                                                                                                     30
                                                                                                     31
                                                                                                                   while (--i >= 1 \&\& (indexed =
105
                    count++;
                                                                                                                      ! equalityComparer.Equals( links.SearchOrDefault(sequence[i - 1],
106
                                                                                                                   \rightarrow sequence[i]), null))) { }
107
                                                                                                                   for (; i \ge 1; i-)
              return count:
108
                                                                                                     33
109
                                                                                                                      links.GetOrCreate(sequence[i - 1], sequence[i]);
                                                                                                     34
110
                                                                                                     35
111
                                                                                                                   return indexed:
                                                                                                     37
./Sequences/SequencesExtensions.cs
                                                                                                     38
     using Platform. Data. Sequences;
                                                                                                                public bool BulkIndex(TLink[] sequence)
                                                                                                     39
     using System.Collections.Generic;
                                                                                                     40
                                                                                                                   var indexed = true;
                                                                                                     41
     namespace Platform.Data.Doublets.Sequences
                                                                                                                   var i = sequence.Length;
                                                                                                     42
                                                                                                                   var links = links.Unsvnc:
                                                                                                     43
        public static class SequencesExtensions
                                                                                                                    links.Sync\overline{R}oot.ExecuteReadOperation(() =>
                                                                                                     44
                                                                                                     45
           public static TLink Create<TLink>(this ISequences<TLink> sequences,
                                                                                                                      while (--i >= 1 \&\& (indexed =
               IList < TLink || > grouped Sequence)
                                                                                                                         ! equalityComparer.Equals(links.SearchOrDefault(sequence[i - 1],
                                                                                                                      \rightarrow sequence[i]), null))) { }
              var finalSequence = new TLink[groupedSequence.Count];
                                                                                                     47
              for (var i = 0; i < \text{finalSequence.Length}; i++)
1.1
                                                                                                                   if (indexed == false)
                                                                                                     48
                                                                                                     49
                 var part = groupedSequence[i];
                                                                                                                       links.SyncRoot.ExecuteWriteOperation(() =>
                                                                                                     50
                 finalSequence[i] = part.Length == 1 ? part[0] : sequences.Create(part);
                                                                                                     51
                                                                                                                         for (; i >= 1; i--)
                                                                                                     52
              return sequences. Create (final Sequence);
                                                                                                                            links.GetOrCreate(sequence[i - 1], sequence[i]);
                                                                                                     54
18
                                                                                                     55
                                                                                                                      });
                                                                                                     56
                                                                                                     57
./Sequences/SequencesIndexer.cs
                                                                                                                   return indexed;
                                                                                                     58
     using System Collections Generic;
                                                                                                     59
                                                                                                     60
     namespace Platform.Data.Doublets.Sequences
                                                                                                                public bool BulkIndexUnsync(TLink[] sequence)
                                                                                                     61
                                                                                                     62
         public class SequencesIndexer<TLink>
                                                                                                                   var indexed = true;
                                                                                                     63
                                                                                                                   var i = sequence.Length:
                                                                                                     64
           private static readonly EqualityComparer<TLink> equalityComparer =
                                                                                                                   var links = links.Unsvnc;
                                                                                                     65
            → EqualityComparer<TLink>.Default;
                                                                                                                   while (--i >= 1 \&\& (indexed =
                                                                                                     66
                                                                                                                      ! equalityComparer.Equals(links.SearchOrDefault(sequence[i - 1],
           private readonly ISynchronizedLinks<TLink> links;
                                                                                                                   \rightarrow sequence[i]), null))) { }
           private readonly TLink null;
                                                                                                                   for (; i >= 1; i-)
           public SequencesIndexer(ISynchronizedLinks<TLink> links)
                                                                                                     68
                                                                                                                      links.GetOrCreate(sequence[i - 1], sequence[i]);
                links = links;
                                                                                                     70
               \overline{\phantom{a}}null = links.Constants.Null;
                                                                                                                   return indexed;
                                                                                                     71
                                                                                                     72
17
                                                                                                     73
                                                                                                                public bool CheckIndex(IList<TLink> sequence)
18
                                                                                                     74
               Индексирует последовательность глобально, и возвращает значение,
19
                                                                                                     75
               определяющие была ли запрошенная последовательность проиндексирована
```

```
var indexed = true;
             var i = sequence.Count:
                                                                                                 5.1
             while (--i \ge 1 \&\& (indexed =
                                                                                                 52
                ! equalityComparer.Equals( links.SearchOrDefault(sequence[i - 1],
                                                                                                                   (MarkedSequenceMatcher == null)
                                                                                                 5.3
                \overline{\text{sequence}[i]}, \text{null})) { }
                                                                                                 54
                                                                                                                    MarkedSequenceMatcher = new
             return indexed:
                                                                                                 55
                                                                                                                    → MarkedSequenceCreteriaMatcher<TLink>(links, SequenceMarkerLink);
81
                                                                                                 56
82
                                                                                                 57
                                                                                                              var balancedVariantConverter = new BalancedVariantConverter<TLink>(links);
                                                                                                 5.8
                                                                                                              if (UseCompression)
./Sequences/SequencesOptions.cs
                                                                                                 60
    using System:
                                                                                                                 if (LinksToSequenceConverter == null)
                                                                                                 61
    using System.Collections.Generic:
    using Platform.Interfaces:
                                                                                                                    ICounter<TLink, TLink> totalSequenceSymbolFrequencyCounter;
    using Platform.Data.Doublets.Sequences.Frequencies.Cache;
                                                                                                                    if (UseSequenceMarker)
    using Platform. Data. Doublets. Sequences. Frequencies. Counters:
                                                                                                 64
    using Platform. Data. Doublets. Sequences. Converters;
                                                                                                 65
                                                                                                                       total Sequence Symbol Frequency Counter = new
    using Platform. Data. Doublets. Sequences. Creteria Matchers:
                                                                                                 66
                                                                                                                           TotalMarkedSequenceSymbolFrequencyCounter<TLink>(links.
    namespace Platform.Data.Doublets.Sequences
                                                                                                                           MarkedSequenceMatcher);
       public class SequencesOptions<TLink> // TODO: To use type parameter <TLink> the
1.1
                                                                                                                    else
           ILinks<TLink> must contain GetConstants function.
                                                                                                 69
                                                                                                                       total Sequence Symbol Frequency Counter = new
                                                                                                 70
          private static readonly EqualityComparer < TLink > equalityComparer =
13
                                                                                                                       → TotalSequenceSymbolFrequencyCounter<TLink>(links);
          → EqualityComparer<TLink>.Default:
14
                                                                                                                    var doubletFrequenciesCache = new LinkFrequenciesCache < TLink > (links,
                                                                                                 72
          public TLink SequenceMarkerLink { get; set; }
15
                                                                                                                        totalSequenceSymbolFrequencyCounter):
          public bool UseCascadeUpdate { get; set; }
                                                                                                                    var compressingConverter = new CompressingConverter < TLink > (links,
          public bool UseCascadeDelete { get; set; }
17
                                                                                                                       balancedVariantConverter, doubletFrequenciesCache);
          public bool UseIndex { get; set; } // TODO: Update Index on sequence update/delete.
                                                                                                                    LinksToSequenceConverter = compressingConverter;
          public bool UseSequenceMarker { get; set; }
19
                                                                                                 75
          public bool UseCompression { get; set; }
                                                                                                 76
          public bool UseGarbageCollection { get; set; }
21
                                                                                                              else
                                                                                                 77
          public bool EnforceSingleSequenceVersionOnWriteBasedOnExisting { get; set; }
                                                                                                 78
          public bool EnforceSingleSequenceVersionOnWriteBasedOnNew { get; set; }
23
                                                                                                                 if (LinksToSequenceConverter == null)
                                                                                                 79
24
          public MarkedSequenceCreteriaMatcher<TLink> MarkedSequenceMatcher { get; set; }
                                                                                                 80
25
                                                                                                                    LinksToSequenceConverter = balancedVariantConverter;
                                                                                                 81
          public IConverter < IList < TLink >, TLink > LinksToSequenceConverter { get; set; }
26
                                                                                                 82
          public SequencesIndexer<TLink> Indexer { get; set; }
27
                                                                                                 83
                                                                                                              if (UseIndex && Indexer == null)
                                                                                                 84
           // TODO: Реализовать компактификацию при чтении
29
            public bool EnforceSingleSequenceVersionOnRead { get; set; }
                                                                                                                 Indexer = new SequencesIndexer<TLink>(links);
                                                                                                 86
            /public bool UseRequestMarker { get; set; }
31
           /public bool StoreRequestResults { get; set; }
                                                                                                 87
32
                                                                                                 88
                                                                                                 89
          public void InitOptions(ISynchronizedLinks<TLink> links)
34
                                                                                                           public void ValidateOptions()
                                                                                                 90
35
                                                                                                 91
               (UseSequenceMarker)
                                                                                                              if (UseGarbageCollection && !UseSequenceMarker)
                                                                                                 92
37
                                                                                                 93
                    equalityComparer.Equals(SequenceMarkerLink, links.Constants.Null))
                                                                                                                 throw new NotSupportedException("To use garbage collection
39
                                                                                                                    UseSequenceMarker option must be on."):
                   SequenceMarkerLink = links.CreatePoint();
                                                                                                 95
                                                                                                 96
                                                                                                 97
                   if (!links.Exists(SequenceMarkerLink))
                                                                                                 98
                      var link = links.CreatePoint();
                      if (! equalityComparer.Equals(link, SequenceMarkerLink))
                                                                                                 ./Sequences/UnicodeMap.cs
                         throw new InvalidOperationException("Cannot recreate sequence
                                                                                                     using System:
                                                                                                     using System.Collections.Generic:
                         → marker link.");
```

```
using System.Globalization;
                                                                                                  66
using System.Runtime.CompilerServices;
                                                                                                  67
using System. Text:
                                                                                                  68
using Platform. Data. Sequences:
                                                                                                  69
                                                                                                  70
namespace Platform.Data.Doublets.Sequences
                                                                                                  71
                                                                                                  72
   public class UnicodeMap
                                                                                                  73
                                                                                                  74
      public static readonly ulong FirstCharLink = 1;
                                                                                                  75
      public static readonly ulong LastCharLink = FirstCharLink + char.MaxValue:
                                                                                                  76
      public static readonly ulong MapSize = 1 + \text{char.MaxValue};
                                                                                                  77
                                                                                                  78
      private readonly ILinks<ulong> links:
                                                                                                  79
      private bool initialized;
                                                                                                  80
      public UnicodeMap(ILinks<ulong> links) => links = links;
                                                                                                  81
                                                                                                  82
      public static UnicodeMap InitNew(ILinks<ulong> links)
                                                                                                  83
                                                                                                  84
         var map = new UnicodeMap(links);
                                                                                                  85
         map.Init():
         return map;
                                                                                                  86
                                                                                                  87
                                                                                                  88
      public void Init()
                                                                                                  90
         if (initialized)
                                                                                                  91
                                                                                                  92
            return:
                                                                                                  93
                                                                                                  94
          initialized = true;
         \overline{\text{var}} first Link = links. CreatePoint():
                                                                                                  95
         if (firstLink!= FirstCharLink)
                                                                                                  96
                                                                                                  97
             links.Delete(firstLink);
                                                                                                  98
                                                                                                  99
         else
                                                                                                 100
                                                                                                 101
            for (var i = FirstCharLink + 1; i \le LastCharLink; i++)
                                                                                                 102
                                                                                                 103
               // From NIL to It (NIL -> Character) transformation meaning, (or infinite
                                                                                                 104
               → amount of NIL characters before actual Character)
                                                                                                 105
               var createdLink = links.CreatePoint():
                                                                                                 106
                                                                                                 107
                 links.Update(createdLink, firstLink, createdLink);
               if (createdLink!= i)
                                                                                                 108
                                                                                                 109
                 throw new InvalidOperationException("Unable to initialize UTF 16 table.");
                                                                                                 110
                                                                                                 111
                                                                                                 112
                                                                                                 113
                                                                                                 114
                                                                                                 115
      // 0 - null link
                                                                                                 116
        1 - nil character (0 character)
                                                                                                 117
                                                                                                 118
                                                                                                 119
      //65536 (0(1) + 65535 = 65536 \text{ possible values})
                                                                                                 120
      [MethodImpl(MethodImplOptions.AggressiveInlining)]
                                                                                                 121
      public static ulong FromCharToLink(char character) => (ulong)character + 1;
                                                                                                 122
                                                                                                 123
      [MethodImpl(MethodImplOptions.AggressiveInlining)]
                                                                                                 124
      public static char FromLinkToChar(ulong link) => (char)(link - 1);
                                                                                                 125
```

11

12

13

14

15

17

19

20

21

22

23

24

25

26

27

28

3.0

31

32

33

34

3.5

36

37

41

51

52

53

54

55

56

57

58

60

61 62

63

```
[MethodImpl(MethodImplOptions.AggressiveInlining)]
public static bool IsCharLink(ulong link) => link <= MapSize;
public static string FromLinksToString(IList<ulong> linksList)
   var sb = new StringBuilder():
   for (int i = 0; i < linksList.Count; i++)
      sb.Append(FromLinkToChar(linksList[i]));
   return sb.ToString();
public static string FromSequenceLinkToString(ulong link, ILinks<ulong> links)
   var sb = new StringBuilder():
   if (links.Exists(link))
      StopableSequenceWalker.WalkRight(link, links.GetSource, links.GetTarget,
         \dot{x} => x \le \text{MapSize } || \text{links.GetSource}(x) == x || \text{links.GetTarget}(x) == x,
             element =>
            sb.Append(FromLinkToChar(element));
            return true:
   return sb.ToString();
public static ulong[] FromCharsToLinkArray(char[] chars) =>
→ FromCharsToLinkArray(chars, chars.Length);
public static ulong [] From Chars To Link Array (char [] chars, int count)
   // char array to ulong array
   var linksSequence = new ulong[count]:
   for (var i = 0: i < count: i++)
      linksSequence[i] = FromCharToLink(chars[i]);
   return linksSequence;
public static ulong[] FromStringToLinkArray(string sequence)
   // char array to ulong array
   var linksSequence = new ulong[sequence.Length];
   for (var i = 0; i < \text{sequence.Length}; i++)
      linksSequence[i] = FromCharToLink(sequence[i]);
   return linksSequence:
public static List<ulong[]> FromStringToLinkArrayGroups(string sequence)
   var result = new List < ulong[] > ();
   var offset = 0;
   while (offset < sequence. Length)
      var current Category = CharUnicodeInfo.GetUnicodeCategory(sequence[offset])
      var relativeLength = 1;
```

```
var absoluteLength = offset + relativeLength;
                                                                                                               return result:
                while (absoluteLength < sequence.Length &&
                                                                                                 185
128
                      currentCategory ==
                                                                                                 186
                      → CharUnicodeInfo.GetUnicodeCategory(sequence[absoluteLength]))
                                                                                                 187
129
                    relativeLength++;
                                                                                                  ./Sequences/Walkers/LeftSequenceWalker.cs
                    absoluteLength++;
131
                                                                                                      using System.Collections.Generic:
                                                                                                      using System.Runtime.CompilerServices:
                   char array to ulong array
133
                var innerSequence = new ulong[relativeLength]:
                                                                                                      namespace Platform. Data. Doublets. Sequences. Walkers
                 var maxLength = offset + relativeLength:
135
                 for (var i = offset; i < maxLength; i++)
136
                                                                                                         public class LeftSequenceWalker<TLink>: SequenceWalkerBase<TLink>
137
                    innerSequence[i - offset] = FromCharToLink(sequence[i]);
138
                                                                                                            public LeftSequenceWalker(ILinks<TLink> links) : base(links) { }
139
                result.Add(innerSequence);
140
                                                                                                            [MethodImpl(MethodImplOptions.AggressiveInlining)]
                 offset += relativeLength;
141
                                                                                                            protected override IList<TLink> GetNextElementAfterPop(IList<TLink> element)
                                                                                                  1.1
142
                                                                                                             ⇒ => Links.GetLink(Links.GetSource(element));
              return result:
143
                                                                                                  12
144
                                                                                                            [MethodImpl(MethodImplOptions.AggressiveInlining)]
                                                                                                  13
145
                                                                                                            protected override IList < TLink > GetNextElement AfterPush (IList < TLink > element)
                                                                                                  14
           public static List<ulong[]> FromLinkArrayToLinkArrayGroups(ulong[] array)
146
                                                                                                             ⇒ => Links.GetLink(Links.GetTarget(element)):
147
                                                                                                  15
              var result = new List < ulong[] > ();
148
                                                                                                            [MethodImpl(MethodImplOptions.AggressiveInlining)]
                                                                                                  16
              var offset = 0:
149
                                                                                                            protected override IEnumerable < IList < TLink >> Walk Contents (IList < TLink >>
                                                                                                  17
              while (offset < array.Length)
150
                                                                                                  18
                 var relativeLength = 1;
152
                                                                                                               var start = Links.Constants.IndexPart + 1:
                                                                                                  19
                if (array[offset] <= LastCharLink)
153
                                                                                                               for (var i = element.Count - 1; i >= start; i--)
                                                                                                  20
154
                                                                                                  21
                    var currentCategory =
155
                                                                                                                  var partLink = Links.GetLink(element[i]);
                                                                                                  22
                    → CharUnicodeInfo.GetUnicodeCategory(FromLinkToChar(array[offset]));
                                                                                                                  if (IsElement(partLink))
                                                                                                  23
                    var absoluteLength = offset + relativeLength;
156
                                                                                                  24
                    while (absoluteLength < array.Length & &
157
                                                                                                                     yield return partLink;
                                                                                                  25
                         array[absoluteLength] <= LastCharLink &&
                        current Category = CharUnicodeInfo.GetUnicodeCategory(FromLinkT_1
159
                                                                                                  27
                         → oChar(array[absoluteLength])))
                                                                                                  28
160
                                                                                                  29
                       relativeLength++:
161
                                                                                                  30
                       absoluteLength++;
                                                                                                  ./Sequences/Walkers/RightSequenceWalker.cs
                                                                                                      using System. Collections. Generic:
                 else
165
                                                                                                      using System.Runtime.CompilerServices;
                    var absoluteLength = offset + relativeLength;
167
                                                                                                      namespace Platform.Data.Doublets.Sequences.Walkers
                    while (absoluteLength < array.Length && array[absoluteLength] >
                        Last CharLink)
                                                                                                         public class RightSequenceWalker<TLink>: SequenceWalkerBase<TLink>
                       relativeLength++;
                                                                                                            public RightSequenceWalker(ILinks<TLink> links) : base(links) { }
                       absoluteLength++:
171
172
                                                                                                            [MethodImpl(MethodImplOptions.AggressiveInlining)]
173
                                                                                                            protected override IList<TLink> GetNextElementAfterPop(IList<TLink> element)
                                                                                                  11
                 // copy array
174
                                                                                                             → => Links.GetLink(Links.GetTarget(element));
                 var innerSequence = new ulong[relativeLength];
175
                                                                                                  12
                 var maxLength = offset + relativeLength:
176
                                                                                                            [MethodImpl(MethodImplOptions.AggressiveInlining)]
                                                                                                  13
                 for (var i = offset; i < maxLength; i++)
                                                                                                            protected override IList < TLink > GetNextElementAfterPush(IList < TLink > element)
                                                                                                  14
                                                                                                             ⇒ => Links.GetLink(Links.GetSource(element));
                    innerSequence[i - offset] = array[i];
179
                                                                                                  15
180
                                                                                                            [MethodImpl(MethodImplOptions.AggressiveInlining)]
                                                                                                  16
                result.Add(innerSequence);
181
                                                                                                            protected override IEnumerable<IList<TLink>> WalkContents(IList<TLink>
                                                                                                  17
                 offset += relativeLength;
182
                                                                                                                element)
183
                                                                                                  18
```

```
for (var i = Links.Constants.IndexPart + 1; i < element.Count; i++)
                var partLink = Links.GetLink(element[i]);
21
                if (IsElement(partLink))
                                                                                                           [MethodImpl(MethodImplOptions.AggressiveInlining)]
                   yield return partLink;
                                                                                                5.1
                                                                                                          protected virtual bool IsElement(IList<TLink> elementLink) =>
25
                                                                                                52
                                                                                                           → Point < TLink > .IsPartialPointUnchecked(elementLink);
                                                                                                53
                                                                                                           [MethodImpl(MethodImplOptions, AggressiveInlining)]
                                                                                                54
                                                                                                          protected abstract IList<TLink> GetNextElementAfterPop(IList<TLink> element);
                                                                                                5.5
                                                                                                56
                                                                                                           [MethodImpl(MethodImplOptions.AggressiveInlining)]
                                                                                                5.7
./Sequences/Walkers/SequenceWalkerBase.cs
                                                                                                          protected abstract IList<TLink> GetNextElementAfterPush(IList<TLink> element);
    using System.Collections.Generic:
                                                                                                59
                                                                                                           [MethodImpl(MethodImplOptions.AggressiveInlining)]
    using System.Runtime.CompilerServices;
                                                                                                60
                                                                                                          protected abstract IEnumerable<IList<TLink>> WalkContents(IList<TLink>
    using Platform. Data. Sequences;
                                                                                                           \rightarrow element):
    namespace Platform.Data.Doublets.Sequences.Walkers
                                                                                                62
                                                                                                63
       public abstract class SequenceWalkerBase<TLink>: LinksOperatorBase<TLink>,
           ISequenceWalker<TLink>
                                                                                                ./Stacks/Stack.cs
                                                                                                    using System Collections Generic;
          // TODO: Use IStack indead of System.Collections.Generic.Stack, but IStack should
                                                                                                    using Platform.Collections.Stacks;
          namespace Platform.Data.Doublets.Stacks
          private readonly Stack<IList<TLink>> stack;
1.1
          protected SequenceWalkerBase(ILinks<TLink> links) : base(links) => stack = new
                                                                                                        public class Stack<TLink>: IStack<TLink>
          \rightarrow Stack<IList<TLink>>():
                                                                                                          private static readonly EqualityComparer<TLink> equalityComparer =
          public IEnumerable<IList<TLink>> Walk(TLink sequence)
                                                                                                           → EqualityComparer<TLink>.Default;
14
                                                                                                          private readonly ILinks<TLink> links;
             if ( stack.Count > 0)
                                                                                                          private readonly TLink stack;
                                                                                                11
                stack.Clear(); // This can be replaced with while(! stack.IsEmpty)
                                                                                                           public Stack(ILinks<TLink> links, TLink stack)
                                                                                                13
                    stack.Pop()
                                                                                                14
                                                                                                               links = links:
                                                                                                1.5
             var element = Links.GetLink(sequence);
                                                                                                               \operatorname{stack} = \operatorname{stack};
                                                                                                16
             if (IsElement(element))
21
                                                                                                17
                                                                                                18
                yield return element:
23
                                                                                                          private TLink GetStackMarker() => links.GetSource( stack);
                                                                                                19
                                                                                                20
             else
                                                                                                          private TLink GetTop() => links.GetTarget( stack);
                                                                                                21
                                                                                                22
                while (true)
                                                                                                          public TLink Peek() => links.GetTarget(GetTop());
                                                                                                23
27
                                                                                                24
                   if (IsElement(element))
                                                                                                           public TLink Pop()
                                                                                                26
                                                                                                              var element = Peek();
                      if ( stack.Count == 0)
                                                                                                              if (! equalityComparer.Equals(element, stack))
                         break:
                                                                                                29
                                                                                                                var top = GetTop():
                                                                                                30
                                                                                                                var previousTop = links.GetSource(top);
                      element = stack.Pop();
                                                                                                31
                      foreach (var output in WalkContents(element))
                                                                                                                  links.Update( stack, GetStackMarker(), previousTop);
                                                                                                32
                                                                                                                 links.Delete(top);
                                                                                                33
                        yield return output;
                                                                                                34
                                                                                                              return element;
                                                                                                35
                      element = GetNextElementAfterPop(element);
                                                                                                36
                                                                                                37
                                                                                                          public void Push(TLink element) => links.Update( stack, GetStackMarker(),
                                                                                                38
                                                                                                               links.GetOrCreate(GetTop(), element));
                       stack.Push(element):
                                                                                                39
                      \overline{element} = \overline{GetNextElementAfterPush(element)}:
                                                                                                40
```

```
./Stacks/StackExtensions.cs
                                                                                                          //public T Trigger(IList<T> restriction, Func<IList<T>, IList<T>, T>
                                                                                                              matchedHandler, IList<T> substitution, Func<IList<T>, IList<T>, T>
    namespace Platform.Data.Doublets.Stacks
                                                                                                              substitutedHandler)
       public static class StackExtensions
                                                                                                               if (restriction!= null && substitution!= null &&
          public static TLink CreateStack<TLink>(this ILinks<TLink> links, TLink
                                                                                                               !substitution.EqualTo(restriction))
                                                                                                                  return SyncRoot. Execute Write Operation (restriction, matched Handler,
              stackMarker)
                                                                                                              substitution, substitutedHandler, Unsync, Trigger):
             var stackPoint = links.CreatePoint():
                                                                                                41
                                                                                                               return SyncRoot. ExecuteReadOperation(restriction, matchedHandler,
             var stack = links.Update(stackPoint, stackMarker, stackPoint);
                                                                                                               substitution, substitutedHandler, Unsync.Trigger);
             return stack:
                                                                                                43
11
                                                                                                44
          public static void DeleteStack<TLink>(this ILinks<TLink> links, TLink stack) =>
                                                                                                45
              links.Delete(stack);
                                                                                                ./UInt64Link.cs
14
                                                                                                    using System:
                                                                                                    using System.Collections:
                                                                                                    using System.Collections.Generic:
./SynchronizedLinks.cs
                                                                                                    using Platform. Exceptions;
                                                                                                    using Platform.Ranges;
    using System;
                                                                                                    using Platform. Helpers. Singletons:
    using System.Collections.Generic;
                                                                                                    using Platform.Data.Constants:
    using Platform. Data. Constants:
    using Platform.Data.Doublets:
                                                                                                    namespace Platform.Data.Doublets
    using Platform. Threading. Synchronization;
                                                                                                10
                                                                                                        /// <summary>
                                                                                                11
    namespace Platform.Data.Doublets
                                                                                                          Структура описывающая уникальную связь.
                                                                                                12
           <remarks>
                                                                                                       public struct UInt64Link: IEquatable < UInt64Link >, IReadOnlyList < ulong >,
                                                                                                14
           TODO: Autogeneration of synchronized wrapper (decorator).
                                                                                                       → IList<ulong>
       /// TODO: Try to unfold code of each method using IL generation for performance
                                                                                                15
       → improvements.
                                                                                                          private static readonly LinksCombinedConstants<br/>
bool, ulong, int> constants =
                                                                                                16
       /// TODO: Or even to unfold multiple layers of implementations.
                                                                                                           → Default<LinksCombinedConstants<br/>
bool, ulong, int>>.Instance;
13
                                                                                                17
       public class SynchronizedLinks<T>: ISynchronizedLinks<T>
                                                                                                          private const int Length = 3;
                                                                                                18
                                                                                                19
          public LinksCombinedConstants<T, T, int> Constants { get; }
                                                                                                          public readonly ulong Index:
                                                                                                20
          public ISynchronization SyncRoot { get; }
                                                                                                          public readonly ulong Source;
17
                                                                                                21
          public ILinks<T> Sync { get; }
                                                                                                          public readonly ulong Target;
                                                                                                22
          public ILinks<T> Unsync { get; }
                                                                                                23
                                                                                                          public static readonly UInt64Link Null = new UInt64Link();
20
                                                                                                24
          public SynchronizedLinks(ILinks<T> links): this(new
                                                                                                25
21
                                                                                                          public UInt64Link(params ulong | values)
                                                                                                26
          → ReaderWriterLockSynchronization(), links) { }
                                                                                                27
                                                                                                             Index = values.Length > constants.IndexPart ? values[ constants.IndexPart] :
          public SynchronizedLinks(ISynchronization synchronization, ILinks<T> links)
                                                                                                28
23
                                                                                                              SvncRoot = svnchronization;
                                                                                                             Source = values.Length > constants.SourcePart? values[ constants.SourcePart]:
25
                                                                                                29
             Svnc = this:
26
                                                                                                                   constants.Null;
             Unsync = links:
27
                                                                                                             Target = values.Length > constants.TargetPart? values[ constants.TargetPart]:
                                                                                                30
             Constants = links.Constants;
28
                                                                                                                  constants.Null;
29
                                                                                                31
30
                                                                                                32
          public T Count(IList<T> restriction) =>
31
                                                                                                          public UInt64Link(IList<ulong> values)
                                                                                                33
              SyncRoot.ExecuteReadOperation(restriction, Unsync.Count);
                                                                                                34
          public T Each(Func<IList<T>, T> handler, IList<T> restrictions) =>
                                                                                                             Index = values.Count > constants.IndexPart ? values[ constants.IndexPart] :
32
                                                                                                35
              SyncRoot.ExecuteReadOperation(handler, restrictions, (handler1, restrictions1)
                                                                                                             \hookrightarrow constants. Null:
              => Unsvnc.Each(handler1, restrictions1));
                                                                                                             Source = values.Count > constants.SourcePart? values[ constants.SourcePart]:
                                                                                                36
          public T Create() => SyncRoot.ExecuteWriteOperation(Unsync.Create);
          public T Update(IList<T> restrictions) =>
                                                                                                             Target = values.Count > constants.TargetPart? values[ constants.TargetPart]:

→ SyncRoot.ExecuteWriteOperation(restrictions, Unsync.Update);

                                                                                                                  constants.Null;
          public void Delete(T link) => SyncRoot.ExecuteWriteOperation(link, Unsync.Delete):
                                                                                                38
35
36
```

```
public UInt64Link(ulong index, ulong source, ulong target)
                                                                                                               return Source:
                                                                                           99
   Index = index:
                                                                                                             if (index == constants.TargetPart)
                                                                                          100
   Source = source:
                                                                                          101
   Target = target:
                                                                                                               return Target:
                                                                                          102
                                                                                          103
                                                                                                            throw new NotSupportedException(); // Impossible path due to
                                                                                          104
public UInt64Link(ulong source, ulong target)
                                                                                                             → Ensure.ArgumentInRange
   : this( constants.Null, source, target)
                                                                                          105
                                                                                                         set => throw new NotSupportedException():
                                                                                          106
   Source = source:
                                                                                          107
   Target = target;
                                                                                          108
                                                                                                      public int Count => Length:
                                                                                          109
                                                                                          110
public static UInt64Link Create(ulong source, ulong target) => new
                                                                                                      public bool IsReadOnly => true;
                                                                                          111
→ UInt64Link(source, target):
                                                                                          112
                                                                                                      IEnumerator IEnumerable.GetEnumerator() => GetEnumerator();
                                                                                          113
public override int GetHashCode() => (Index, Source, Target).GetHashCode();
                                                                                          114
                                                                                                      public IEnumerator < ulong > GetEnumerator()
                                                                                          115
public bool IsNull() => Index == constants.Null
                                                                                          116
                && Source == constants.Null
                                                                                                         vield return Index:
                                                                                          117
                && Target == constants. Null;
                                                                                                         vield return Source;
                                                                                          118
                                                                                          119
                                                                                                         vield return Target:
public override bool Equals(object other) => other is UInt64Link &&
                                                                                          120
→ Equals((UInt64Link)other):
                                                                                          121
                                                                                                      public void Add(ulong item) => throw new NotSupportedException();
                                                                                          122
public bool Equals(UInt64Link other) => Index == other.Index
                                                                                          123
                                                                                                      public void Clear() => throw new NotSupportedException();
                            && Source == other Source
                                                                                          124
                            && Target == other. Target:
                                                                                          125
                                                                                                      public bool Contains(ulong item) => IndexOf(item) >= 0;
                                                                                          126
public static string ToString(ulong index, ulong source, ulong target) => \mathbb{S}"({index}:
                                                                                          127
                                                                                                      public void CopyTo(ulong[] array, int arrayIndex)
                                                                                          128
\rightarrow {source}->{target})";
                                                                                          129
                                                                                                         Ensure. Always. ArgumentNotNull(array, nameof(array));
public static string ToString(ulong source, ulong target) => \( \bar{\string} \) (\{\string \text{varget}}\) ";
                                                                                          130
                                                                                                         Ensure. Always. Argument In Range (array Index, new Range < int > (0, array. Length -
                                                                                          131
public static implicit operator ulong[](UInt64Link link) => link.ToArray();
                                                                                                          \rightarrow 1), nameof(arrayIndex));
                                                                                                         if (arrayIndex + Length > array.Length)
                                                                                          132
public static implicit operator UInt64Link(ulong | linkArray) => new
                                                                                          133

    UInt64Link(linkArray);

                                                                                                            throw new ArgumentException();
                                                                                          134
                                                                                          135
public ulong ToArray()
                                                                                                         array[arrayIndex++] = Index:
                                                                                          136
                                                                                                         \operatorname{array}[\operatorname{array} \operatorname{Index} + +] = \operatorname{Source};
                                                                                          137
   var array = new ulong[Length];
                                                                                                         array[arrayIndex] = Target;
                                                                                          138
   CopvTo(array, 0):
                                                                                          139
   return array;
                                                                                          140
                                                                                                      public bool Remove(ulong item) =>
                                                                                          141
                                                                                                       → Throw.A.NotSupportedExceptionAndReturn<br/>
<bool>();
public override string ToString() => Index == constants.Null? ToString(Source,
                                                                                          142
→ Target) : ToString(Index, Source, Target):
                                                                                                      public int IndexOf(ulong item)
                                                                                          143
                                                                                          144
#region IList
                                                                                                         if (Index == item)
                                                                                          145
                                                                                          146
public ulong this[int index]
                                                                                                            return constants.IndexPart;
                                                                                          147
                                                                                          148
                                                                                                           (Source == item)
                                                                                          149
                                                                                          150
      Ensure. Always. Argument InRange (index, new Range < int > (0, Length - 1),
                                                                                          151
                                                                                                            return constants.SourcePart;
      \rightarrow nameof(index));
                                                                                          152
     if (index == constants.IndexPart)
                                                                                                         if (Target == item)
                                                                                          153
                                                                                          154
         return Index:
                                                                                                            return constants. TargetPart;
                                                                                          155
                                                                                          156
      if (index == constants.SourcePart)
                                                                                          157
```

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

5.8

59

60

61

64

65

71

72

73

74

76

77

81

82

83

85

86

87

90

91

92

```
return -1;
                                                                                                                    if (sequence == null)
                                                                                                      34
159
                                                                                                      35
                                                                                                                       return:
160
                                                                                                      36
            public void Insert(int index, ulong item) => throw new NotSupportedException();
161
                                                                                                      37
162
                                                                                                                     for (var i = 0; i < sequence.Count; i++)
                                                                                                      38
            public void RemoveAt(int index) => throw new NotSupportedException();
163
                                                                                                      39
164
                                                                                                                        if (sequence[i] != Constants.Any && !links.Exists(sequence[i]))
                                                                                                      40
            #endregion
165
                                                                                                      41
166
                                                                                                                          throw new ArgumentLinkDoesNotExistsException<ulong>(sequence[i],
                                                                                                      42
167
                                                                                                                           \rightarrow $\square\[square\[square\]\]');
                                                                                                      43
./UInt64LinkExtensions.cs
                                                                                                      44
     namespace Platform.Data.Doublets
                                                                                                      45
                                                                                                      46
                                                                                                                  public static bool AnyLinkIsAny(this ILinks<ulong> links, params ulong[] sequence)
        public static class UInt64LinkExtensions
                                                                                                      47
                                                                                                      48
                                                                                                                    if (sequence == null)
            public static bool IsFullPoint(this UInt64Link link) =>
                                                                                                      49
            → Point < ulong >. IsFullPoint(link):
                                                                                                      50
                                                                                                                        return false:
           public static bool IsPartialPoint(this UInt64Link link) =>
                                                                                                      5.1
                                                                                                      52
            → Point < ulong > .IsPartialPoint(link);
                                                                                                                     var constants = links.Constants:
                                                                                                      53
                                                                                                                     for (var i = 0; i < \text{sequence.Length}; i++)
                                                                                                      54
                                                                                                      5.5
                                                                                                                        if (sequence[i] == constants.Any)
./UInt64LinksExtensions.cs
                                                                                                      5.7
     using System:
                                                                                                                          return true;
                                                                                                      58
     using System. Text:
     using System.Collections.Generic:
                                                                                                      60
     using Platform. Helpers. Singletons;
                                                                                                                     return false:
                                                                                                      61
     using Platform.Data.Constants;
                                                                                                      62
     using Platform Data Exceptions:
                                                                                                      63
     using Platform. Data. Doublets. Sequences;
                                                                                                                 public static string FormatStructure(this ILinks<ulong> links, ulong linkIndex,
                                                                                                                      Func<UInt64Link, bool> isElement, bool renderIndex = false, bool renderDebug
     namespace Platform.Data.Doublets
                                                                                                                      = false)
        public static class UInt64LinksExtensions
1.1
                                                                                                                    var sb = new StringBuilder():
                                                                                                      66
12
                                                                                                                     var\ visited = new\ HashSet < ulong > ():
                                                                                                      67
           public static readonly LinksCombinedConstants<br/>
bool, ulong, int > Constants =
13
                                                                                                                     links.AppendStructure(sb, visited, linkIndex, isElement, (innerSb, link) =>
                                                                                                      68
               Default<LinksCombinedConstants<br/>bool, ulong, int>>.Instance;
                                                                                                                     → innerSb.Append(link.Index), renderIndex, renderDebug);
14
                                                                                                                     return sb.ToString():
                                                                                                      69
            public static void UseUnicode(this ILinks<ulong> links) =>
15
                                                                                                      70
               UnicodeMap.InitNew(links);
                                                                                                      71
                                                                                                                 public static string FormatStructure(this ILinks<ulong> links, ulong linkIndex,
                                                                                                      72
            public static void EnsureEachLinkExists(this ILinks<ulong> links, IList<ulong>
17
                                                                                                                      Func<UInt64Link, bool> isElement, Action<StringBuilder, UInt64Link>
                sequence)
                                                                                                                      appendElement, bool renderIndex = false, bool renderDebug = false)
                                                                                                      73
              if (sequence == null)
1.9
                                                                                                                     var sb = new StringBuilder():
                                                                                                      74
                                                                                                                     var\ visited = new\ HashSet < ulong > ():
                                                                                                      75
                 return;
21
                                                                                                                    links. AppendStructure(sb, visited, linkIndex, isElement, appendElement,
                                                                                                      76
22
                                                                                                                     → renderIndex, renderDebug);
               for (var i = 0; i < sequence.Count; i++)
23
                                                                                                                     return sb.ToString();
                                                                                                      77
24
                  if (!links.Exists(sequence[i]))
                                                                                                      78
25
                                                                                                      79
                                                                                                                 public static void AppendStructure(this ILinks<ulong> links, StringBuilder sb,
                                                                                                      80
                     throw new ArgumentLinkDoesNotExistsException<ulong>(sequence[i],
27
                                                                                                                      HashSet < ulong > visited, ulong linkIndex, Func < UInt 64Link, bool > is Element,
                         \mathbb{S}" sequence[{i}]");
                                                                                                                      Action < StringBuilder, UInt64Link > appendElement, bool renderIndex = false,
                                                                                                                      bool renderDebug = false)
29
                                                                                                      81
30
                                                                                                                    if (sb == null)
                                                                                                      82
31
                                                                                                      83
            public static void EnsureEachLinkIsAnyOrExists(this ILinks<ulong> links,
32
                                                                                                                        throw new ArgumentNullException(nameof(sb));
                                                                                                      84
                IList < ulong > sequence)
                                                                                                      85
33
```

```
if (linkIndex == Constants.Null || linkIndex == Constants.Any || linkIndex ==
                                                                                                   else
                                                                                     145
    Constants.Itself)
                                                                                     146
                                                                                                        (renderDebug)
                                                                                     147
  return;
                                                                                     148
                                                                                                         sb.Append(1^{-1});
                                                                                     149
  (links.Exists(linkIndex))
                                                                                     150
                                                                                                      sb.Append(linkIndex);
                                                                                     151
   if (visited.Add(linkIndex))
                                                                                     152
                                                                                     153
      sb.Append('(')):
                                                                                     154
      var link = new UInt64Link(links.GetLink(linkIndex));
                                                                                     155
      if (renderIndex)
                                                                                      ./UInt64LinksTransactionsLayer.cs
         sb.Append(link.Index);
                                                                                          using System;
         sb.Append(':');
                                                                                          using System Ling;
                                                                                          using System.Collections.Generic;
        (link.Source == link.Index)
                                                                                          using System.IO:
                                                                                          using System.Runtime.CompilerServices;
         sb.Append(link.Index);
                                                                                          using System. Threading;
                                                                                          using System. Threading. Tasks:
                                                                                          using Platform. Disposables:
                                                                                          using Platform. Timestamps;
         var source = new UInt64Link(links.GetLink(link.Source));
                                                                                          using Platform.Unsafe;
         if (isElement(source))
                                                                                          using Platform.IO:
                                                                                      11
                                                                                          using Platform. Data. Doublets. Decorators;
           appendElement(sb, source);
                                                                                      13
                                                                                          namespace Platform.Data.Doublets
                                                                                      14
                                                                                      15
                                                                                             public class UInt64LinksTransactionsLayer: LinksDisposableDecoratorBase<ulong>
                                                                                      16
           links. AppendStructure(sb, visited, source. Index, isElement,
                                                                                                  //-V3073
                appendElement, renderIndex);
                                                                                      17
                                                                                                    <remarks>
                                                                                      18
                                                                                                    Альтернативные варианты хранения трансформации (элемента транзакции):
                                                                                      19
      sb.Append('');
                                                                                      20
      if (link.Target == link.Index)
                                                                                                    private enum TransitionType
                                                                                     21
                                                                                      22
         sb.Append(link.Index);
                                                                                                       Creation.
                                                                                     23
                                                                                                       UpdateOf,
                                                                                      ^{24}
      else
                                                                                                       UpdateTo,
                                                                                      25
                                                                                                       Deletion
                                                                                      26
         var target = new UInt64Link(links.GetLink(link.Target));
                                                                                      27
         if (isElement(target))
                                                                                      28
                                                                                                    private struct Transition
                                                                                      29
           appendElement(sb, target);
                                                                                      30
                                                                                                       public ulong TransactionId;
                                                                                     31
                                                                                                       public UniqueTimestamp Timestamp;
                                                                                      32
                                                                                                       public TransactionItemType Type:
                                                                                      33
           links.AppendStructure(sb, visited, target.Index, isElement,
                                                                                                       public Link Source:
                                                                                      34
               appendElement, renderIndex);
                                                                                                       public Link Linker;
                                                                                      35
                                                                                                       public Link Target;
                                                                                      36
                                                                                      37
      sb.Append(')');
                                                                                      38
                                                                                                    Или
                                                                                      39
                                                                                      40
                                                                                                    public struct TransitionHeader
                                                                                      41
      if (renderDebug)
                                                                                      42
                                                                                                       public ulong TransactionIdCombined;
                                                                                      43
         sb.Append(***);
                                                                                                       public ulong TimestampCombined;
                                                                                      44
                                                                                      45
      sb.Append(linkIndex);
                                                                                                       public ulong TransactionId
                                                                                      46
                                                                                      47
                                                                                                          get
                                                                                      48
```

0.1

100

101

102

103

104

106

109

110

111

112

113

114

115

116

117

119

122

123

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

142

143

```
108
            return (ulong) mask & TransactionIdCombined:
                                                                                                public override string ToString() => S'' \{Timestamp\} \{TransactionId\}: \{Before\}
                                                                                  100
                                                                                                \Rightarrow => {After}";
                                                                                  110
                                                                                  111
      public UniqueTimestamp Timestamp
                                                                                                <remarks>
                                                                                  112
                                                                                                Другие варианты реализации транзакций (атомарности):
                                                                                  113
                                                                                                   1. Разделение хранения значения связи ((Source Target) или (Source Linker
         get
                                                                                  114
                                                                                                 Target)) и индексов.
            return (UniqueTimestamp) mask & TransactionIdCombined:
                                                                                                   2. Хранение трансформаций/операций в отдельном хранилище Links, но
                                                                                                 дополнительно потребуется решить вопрос
                                                                                                     со ссылками на внешние идентификаторы, или как-то иначе решить
                                                                                  116
                                                                                                 вопрос с пересечениями идентификаторов.
      public TransactionItemType Type
                                                                                  117
                                                                                                Где хранить промежуточный список транзакций?
                                                                                  118
                                                                                  119
                                                                                                В оперативной памяти:
                                                                                  120
              Использовать по одному биту из TransactionId и Timestamp.
                                                                                                 Минусы:
                                                                                  121
              для значения в 2 бита, которое представляет тип операции
                                                                                                   1. Может усложнить систему, если она будет функционировать
                                                                                  122
           throw new NotImplementedException();
                                                                                                 самостоятельно.
                                                                                                   так как нужно отдельно выделять память под список трансформаций.
                                                                                  123
                                                                                                   2. Выделенной оперативной памяти может не хватить, в том случае,
                                                                                  124
                                                                                                   если транзакция использует слишком много трансформаций.
                                                                                  125

    Можно использовать жёсткий диск для слишком длинных транзакций.

                                                                                  126
   private struct Transition
                                                                                                      -> Максимальный размер списка трансформаций можно ограничить
                                                                                  127
                                                                                                 задать константой.
      public TransitionHeader Header;
                                                                                                   3. При подтверждении транзакции (Commit) все трансформации
                                                                                  128
      public Link Source:
                                                                                                 записываются разом создавая задержку.
      public Link Linker;
                                                                                  129
      public Link Target:
                                                                                                На жёстком диске:
                                                                                  130
                                                                                                 Минусы:
                                                                                  131
                                                                                                   1. Длительный отклик, на запись каждой трансформации.
                                                                                  132
                                                                                                   2. Лог транзакций дополнительно наполняется отменёнными транзакциями.
   </remarks>
                                                                                  133
public struct Transition
                                                                                                     -> Это может решаться упаковкой/исключением дублирующих операций.
                                                                                  134
                                                                                                      -> Также это может решаться тем, что короткие транзакции вообще
                                                                                  135
  public static readonly long Size = StructureHelpers.SizeOf<Transition>():
                                                                                                        не будут записываться в случае отката.
                                                                                  136
                                                                                                   3. Перед тем как выполнять отмену операций транзакции нужно дождаться
                                                                                  137
  public readonly ulong TransactionId:
                                                                                                 пока все операции (трансформации)
  public readonly UInt64Link Before:
                                                                                                     будут записаны в лог.
                                                                                  138
  public readonly UInt64Link After:
                                                                                  139
  public readonly Timestamp;
                                                                                                </remarks>
                                                                                  140
                                                                                             public class Transaction: DisposableBase
                                                                                  141
  public Transition(UniqueTimestampFactory uniqueTimestampFactory, ulong
                                                                                  142
      transactionId, UInt64Link before, UInt64Link after)
                                                                                                private readonly Queue < Transition > transitions;
                                                                                  143
                                                                                                private readonly UInt64LinksTransactionsLayer layer;
                                                                                  144
     TransactionId = transactionId;
                                                                                                public bool IsCommitted { get; private set; }
                                                                                  145
     Before = before;
                                                                                                public bool IsReverted { get; private set; }
                                                                                  146
     After = after:
                                                                                  147
     Timestamp = uniqueTimestampFactory.Create();
                                                                                                public Transaction(UInt64LinksTransactionsLayer layer)
                                                                                  148
                                                                                  149
                                                                                                    laver = laver:
                                                                                  150
  public Transition(UniqueTimestampFactory uniqueTimestampFactory, ulong
                                                                                                  \overline{if} ( layer. currentTransactionId!= 0)
                                                                                  151
   152
     : this(uniqueTimestampFactory, transactionId, before, default)
                                                                                                     throw new NotSupportedException("Nested transactions not supported.");
                                                                                  153
                                                                                  154
                                                                                                   \hat{I}sCommitted = false;
                                                                                  155
                                                                                                  IsReverted = false:
                                                                                  156
  public Transition(UniqueTimestampFactory uniqueTimestampFactory, ulong
                                                                                                   transitions = new Queue<Transition>();
                                                                                  157
      transactionId)
                                                                                                  SetCurrentTransaction(layer, this);
                                                                                  158
     : this(uniqueTimestampFactory, transactionId, default, default)
                                                                                  159
                                                                                  160
                                                                                                public void Commit()
                                                                                  161
```

54

55

62

73

74

82

93

97

100

101

102

103

105

106

```
private readonly string logAddress;
162
                                                                                                    222
                 EnsureTransactionAllowsWriteOperations(this):
                                                                                                                private readonly FileStream log:
                                                                                                    223
163
                                                                                                                private readonly Queue < Transition > transitions;
                 while ( transitions. Count > 0)
                                                                                                    224
164
                                                                                                                private readonly UniqueTimestampFactory uniqueTimestampFactory;
                                                                                                    225
165
                                                                                                                private Task transitionsPusher:
                     var transition = transitions.Dequeue();
                                                                                                    226
166
                                                                                                                private Transition lastCommittedTransition;
                     layer. transitions. Enqueue(transition):
                                                                                                    227
167
                                                                                                                private ulong currentTransactionId:
                                                                                                    228
168
                                                                                                                private Queue Transition current Transaction Transitions:
                                                                                                    229
                   layer. lastCommitedTransactionId = layer. currentTransactionId;
                                                                                                                private Transaction current Transaction;
                                                                                                    230
                 \overline{\text{IsCommitted}} = \text{true};
170
                                                                                                                private ulong lastCommittedTransactionId;
                                                                                                    231
                                                                                                    232
172
                                                                                                                public UInt64LinksTransactionsLayer(ILinks<ulong> links, string logAddress)
                                                                                                    233
              private void Revert()
173
                                                                                                                   : base(links)
                                                                                                    234
174
                                                                                                    235
                 EnsureTransactionAllowsWriteOperations(this):
175
                                                                                                                   if (string.IsNullOrWhiteSpace(logAddress))
                                                                                                     236
                 var transitionsToRevert = new Transition transitions.Count
176
                                                                                                    237
                  transitions. Copy To(transitions ToRevert, \overline{0}):
177
                                                                                                                      throw new ArgumentNullException(nameof(logAddress));
                                                                                                    238
                 for (var i = transitionsToRevert.Length - 1: i \geq 0: i--)
178
                                                                                                    239
179
                                                                                                    240
                                                                                                                      В первой строке файла хранится последняя закоммиченную транзакцию.
                     layer.RevertTransition(transitionsToRevert[i]);
                                                                                                                      При запуске это используется для проверки удачного закрытия файла лога.
                                                                                                    241
181
                                                                                                                      In the first line of the file the last committed transaction is stored.
                                                                                                    242
                  \mathbf{IsReverted} = \mathbf{true}
182
                                                                                                                      On startup, this is used to check that the log file is successfully closed.
                                                                                                    243
183
                                                                                                                   var lastCommitedTransition =
                                                                                                    244
184
                                                                                                                      FileHelpers.ReadFirstOrDefault<Transition>(logAddress);
              public static void SetCurrentTransaction(UInt64LinksTransactionsLayer layer,
185
                                                                                                                   var lastWrittenTransition =
                                                                                                    245
                  Transaction transaction)
                                                                                                                        FileHelpers.ReadLastOrDefault<Transition>(logAddress);
                                                                                                                   if (!lastCommittedTransition.Equals(lastWrittenTransition))
                 layer. current Transaction Id = layer. last Committed Transaction Id + 1;
                                                                                                    246
187
                 layer. current Transaction Transitions = transaction. transitions;
                                                                                                    247
                 layer. current Transaction = transaction:
                                                                                                    248
189
                                                                                                                      throw new NotSupportedException("Database is damaged, autorecovery is not
190
                                                                                                    249
191

→ supported yet.");

              public static void EnsureTransactionAllowsWriteOperations(Transaction
192
                                                                                                    250
                  transaction)
                                                                                                                     (lastCommittedTransition.Equals(default(Transition)))
                                                                                                    251
193
                                                                                                    252
                 if (transaction.IsReverted)
194
                                                                                                                      FileHelpers.WriteFirst(logAddress, lastCommittedTransition);
                                                                                                    253
                                                                                                    254
                     throw new InvalidOperationException("Transation is reverted.");
                                                                                                                     lastCommitedTransition = lastCommitedTransition;
196
                                                                                                    255
                                                                                                                    // TODO: Think about a better way to calculate or store this value
197
                                                                                                    256
                    (transaction.IsCommitted)
                                                                                                                   var allTransitions = FileHelpers.ReadAll<Transition>(logAddress);
198
                                                                                                    257
                                                                                                                    lastCommittedTransactionId = allTransitions.Max(x = > x.TransactionId):
199
                                                                                                    258
                     throw new InvalidOperationException("Transation is committed.");
                                                                                                                     uniqueTimestampFactory = new UniqueTimestampFactory():
                                                                                                    259
200
                                                                                                                    \log Address = \log Address;
201
                                                                                                    260
                                                                                                                     \log = \text{FileHelpers.Append}(\log \text{Address});
                                                                                                    261
202
203
                                                                                                                     transitions = new Queue < Transition > ():
                                                                                                    262
              protected override void DisposeCore(bool manual, bool wasDisposed)
204
                                                                                                                     transitionsPusher = new Task(TransitionsPusher)
                                                                                                     263
205
                                                                                                                    transitionsPusher.Start();
                                                                                                    264
                 if (!wasDisposed && layer != null &&! layer.IsDisposed)
206
                                                                                                    265
207
                                                                                                     266
                    if (!IsCommitted && !IsReverted)
                                                                                                                public IList<ulong> GetLinkValue(ulong link) => Links.GetLink(link);
                                                                                                    267
208
                                                                                                    268
                                                                                                                public override ulong Create()
                       Revert():
                                                                                                    269
210
                                                                                                    270
211
                                                                                                                   var createdLinkIndex = Links.Create();
                      layer.ResetCurrentTransation();
                                                                                                    271
212
                                                                                                                   var createdLink = new UInt64Link(Links.GetLink(createdLinkIndex));
                                                                                                    272
213
                                                                                                                   CommitTransition(new Transition(uniqueTimestampFactory,
                                                                                                    273
214
                                                                                                                         current TransactionId, default, createdLink));
215
              // TODO: THIS IS EXCEPTION WORKAROUND, REMOVE IT THEN
                                                                                                                   return createdLinkIndex;
                                                                                                    274
216
                  https://github.com/linksplatform/Disposables/issues/13 FIXED
                                                                                                    275
              protected override bool AllowMultipleDisposeCalls => true;
                                                                                                    276
217
                                                                                                                public override ulong Update(IList<ulong> parts)
                                                                                                    277
218
^{219}
                                                                                                    278
           public static readonly TimeSpan DefaultPushDelay = TimeSpan.FromSeconds(0.1);
                                                                                                                   var beforeLink = new UInt64Link(Links.GetLink(parts[Constants.IndexPart]));
                                                                                                    279
220
221
```

```
parts[Constants.IndexPart] = Links.Update(parts);
                                                                                                                                                                                    333
                                                                                                                                                                                                                        return:
     var afterLink = new UInt64Link(Links.GetLink(parts|Constants.IndexPart|)):
                                                                                                                                                                                    334
     CommitTransition(new Transition(uniqueTimestampFactory,
                                                                                                                                                                                                                  for (var i = 0; i < transitions.Count; <math>i++)
                                                                                                                                                                                    335
               currentTransactionId, beforeLink, afterLink)):
                                                                                                                                                                                    336
                                                                                                                                                                                                                        var transition = transitions. Dequeue();
     return parts[Constants.IndexPart];
                                                                                                                                                                                    337
                                                                                                                                                                                    338
                                                                                                                                                                                                                           log.Write(transition);
                                                                                                                                                                                    339
                                                                                                                                                                                                                           -lastCommittedTransition = transition;
public override void Delete(ulong link)
                                                                                                                                                                                    340
                                                                                                                                                                                    341
      var deletedLink = new UInt64Link(Links.GetLink(link));
                                                                                                                                                                                    342
                                                                                                                                                                                    343
     Links. Delete(link):
                                                                                                                                                                                                            private void TransitionsPusher()
                                                                                                                                                                                    344
     CommitTransition(new Transition(uniqueTimestampFactory,
                                                                                                                                                                                    345
      while (!IsDisposed && transitionsPusher!= null)
                                                                                                                                                                                    347
                                                                                                                                                                                                                        Thread.Sleep(DefaultPushDelay);
[MethodImpl(MethodImplOptions.AggressiveInlining)]
                                                                                                                                                                                    348
                                                                                                                                                                                                                        PushTransitions();
                                                                                                                                                                                    349
private Queue < Transition > GetCurrentTransitions() = >
                                                                                                                                                                                    350
351
private void CommitTransition(Transition transition)
                                                                                                                                                                                    352
                                                                                                                                                                                                           public Transaction BeginTransaction() => new Transaction(this);
                                                                                                                                                                                    353
                                                                                                                                                                                    354
     if ( current Transaction != null)
                                                                                                                                                                                                            private void DisposeTransitions()
                                                                                                                                                                                    355
            Transaction. Ensure Transaction Allows Write Operations (current Transaction):
                                                                                                                                                                                    357
                                                                                                                                                                                                                 try
                                                                                                                                                                                    358
      var transitions = GetCurrentTransitions();
                                                                                                                                                                                                                        var pusher = transitionsPusher;
                                                                                                                                                                                    359
     transitions. Enqueue(transition);
                                                                                                                                                                                                                        if (pusher != null)
                                                                                                                                                                                    360
                                                                                                                                                                                    361
                                                                                                                                                                                                                                 transitionsPusher = null:
                                                                                                                                                                                    362
private void RevertTransition(Transition transition)
                                                                                                                                                                                                                              pusher.Wait():
                                                                                                                                                                                    363
                                                                                                                                                                                    364
      if (transition.After.IsNull()) // Revert Deletion with Creation
                                                                                                                                                                                                                            ( transitions != null)
                                                                                                                                                                                    365
                                                                                                                                                                                    366
           Links.Create();
                                                                                                                                                                                                                              PushTransitions();
                                                                                                                                                                                    367
                                                                                                                                                                                    368
      else if (transition.Before.IsNull()) // Revert Creation with Deletion
                                                                                                                                                                                                                        Disposable. Try Dispose (log):
                                                                                                                                                                                                                        FileHelpers.WriteFirst( logAddress, lastCommitedTransition);
                                                                                                                                                                                    370
           Links.Delete(transition.After.Index);
                                                                                                                                                                                    371
                                                                                                                                                                                    372
                                                                                                                                                                                                                  catch
     else // Revert Update
                                                                                                                                                                                    374
           Links. Update(new[] { transition. After. Index, transition. Before. Source,
                                                                                                                                                                                    375
                   transition.Before.Target \});
                                                                                                                                                                                                           #region DisposalBase
                                                                                                                                                                                    377
                                                                                                                                                                                    378
                                                                                                                                                                                                            protected override void DisposeCore(bool manual, bool wasDisposed)
                                                                                                                                                                                    379
private void ResetCurrentTransation()
                                                                                                                                                                                    380
                                                                                                                                                                                                                  if (!wasDisposed)
                                                                                                                                                                                    381
         current Transaction Id = 0;
                                                                                                                                                                                    382
        Current Transaction Transitions = null;
                                                                                                                                                                                                                        DisposeTransitions():
                                                                                                                                                                                    383
        \overline{\text{currentTransaction}} = \text{null};
                                                                                                                                                                                    384
                                                                                                                                                                                                                   base.DisposeCore(manual, wasDisposed);
                                                                                                                                                                                    386
private void PushTransitions()
                                                                                                                                                                                    387
                                                                                                                                                                                                            #endregion
    \inf_{\{ x \in \mathcal{A} \mid x = 0 \text{ if } (x) = 0 \text{ or } x = 0 \text{ o
                                                                                                                                                                                    389
                                                                                                                                                                                    390
```

	Converters/Address (OutaryNumberConverter.cs, 1
	/Converters/LinkToltsFrequencyNumberConveter.cs, 1
	/Converters/PowerOf2ToUnaryNumberConverter.cs, 1
	/Converters/UnaryNumberToAddressAddOperationConverter.cs, 2
	/Converters/UnaryNumberToAddressOrOperationConverter.cs, 2
	/Decorators/LinksCascadeDependenciesResolver.cs, 3
	/Decorators/LinksCascadeUniquenessAndDependenciesResolver.cs, 3
	/Decorators/LinksDecoratorBase.cs, 3
	/Decorators/LinksDependenciesValidator.cs, 4
	/Decorators/LinksDisposableDecoratorBase.cs, 4
	/Decorators/LinksInnerReferenceValidator.cs, 4
	/Decorators/LinksNonExistentReferencesCreator.cs, 4
	/Decorators/LinksNullToSelfReferenceResolver.cs, 5
	/Decorators/LinksSelfReferenceResolver.cs, 5
	/Decorators/LinksUniquenessResolver.cs, 5
	/Decorators/LinksUniquenessValidator.cs, 6
	/Decorators/NonNullContentsLinkDeletionResolver.cs, 6
	/Decorators/UInt64Links.cs, 6
	/Decorators/UniLinks.cs, 7
	/Doublet.cs, 10
	/DoubletComparer.cs, 10
	/Hybrid.cs, 10
	/ILinks.cs, 11
	/ILinksExtensions.cs, 11
	/ISynchronizedLinks.cs, 18
	/Incrementers/FrequencyIncrementer.cs, 17
	/Incrementers/LinkFrequencyIncrementer.cs, 17
	/Incrementers/UnaryNumberIncrementer.cs, 18
	Link.cs, 18
	/LinkExtensions.cs, 20
	/LinksOperatorBase.cs, 20
٠	/PropertyOperators/DefaultLinkPropertyOperator.cs, 20
	/PropertyOperators/FrequencyPropertyOperator.cs, 20
	/ResizableDirectMemory/ResizableDirectMemoryLinks.ListMethods.cs, 27
	/ResizableDirectMemory/ResizableDirectMemoryLinks.TreeMethods.cs, 27
	/ResizableDirectMemory/ResizableDirectMemoryLinks.cs, 21
	/ResizableDirectMemory/UInt64ResizableDirectMemoryLinks.ListMethods.cs, 35
	$/ Resizable Direct Memory/UInt 64 Resizable Direct Memory Links. Tree Methods. cs,\ 36 Resizable Direct Memory Annual Memory M$
	/ResizableDirectMemory/UInt64ResizableDirectMemoryLinks.cs, 31
	/Sequences/Converters/BalancedVariantConverter.cs, 40
	/Sequences/Converters/CompressingConverter.cs, 40

Index

```
./Sequences/Converters/LinksListToSequenceConverterBase.cs, 42
/Sequences/Converters/OptimalVariantConverter.cs, 42
//Sequences/Converters/SequenceToltsLocalElementLevelsConverter.cs, 43
/Sequences/Creteria Matchers/DefaultSequenceElementCreteria Matcher.cs, 44
./Sequences/Creteria Matchers/Marked Sequence Creteria Matcher.cs. 44
/Sequences/DefaultSequenceAppender.cs, 44
/Sequences/DuplicateSegmentsCounter.cs, 44
./Sequences/DuplicateSegmentsProvider.cs, 45
./Sequences/Frequencies/Cache/FrequenciesCacheBasedLinkFrequencyIncrementer.cs, 46
./Sequences/Frequencies/Cache/FrequenciesCacheBasedLinkToltsFrequencyNumberConverter.cs.
./Sequences/Frequencies/Cache/LinkFrequenciesCache.cs, 46
/Sequences/Frequencies/Cache/LinkFrequency.cs, 47
/Sequences/Frequencies/Counters/MarkedSequenceSymbolFrequencyOneOffCounter.cs, 48
/Sequences/Frequencies/Counters/SequenceSymbolFrequencyOneOffCounter.cs, 48
/Sequences/Frequencies/Counters/TotalMarkedSequenceSymbolFrequencyCounter.cs, 48
/Sequences/Frequencies/Counters/TotalMarkedSequenceSymbolFrequencyOneOffCounter.cs,
./Sequences/Frequencies/Counters/TotalSequenceSymbolFrequencyCounter.cs, 49
/Sequences/Frequencies/Counters/TotalSequenceSymbolFrequencyOneOffCounter.cs, 49
./Sequences/HeightProviders/CachedSequenceHeightProvider.cs, 49
./Sequences/HeightProviders/DefaultSequenceRightHeightProvider.cs, 50
/Sequences/HeightProviders/ISequenceHeightProvider.cs, 50
./Sequences/Sequences.Experiments.ReadSequence.cs, 73
/Sequences/Sequences.Experiments.cs. 56
/Sequences/Sequences.cs, 50
./Sequences/SequencesExtensions.cs, 74
./Sequences/SequencesIndexer.cs, 74
/Sequences/SequencesOptions.cs, 75
./Sequences/UnicodeMap.cs, 75
/Sequences/Walkers/LeftSequenceWalker.cs, 77
/Sequences/Walkers/RightSequenceWalker.cs, 77
./Sequences/Walkers/SequenceWalkerBase.cs, 78
/Stacks/Stack.cs, 78
./Stacks/StackExtensions.cs, 79
./SynchronizedLinks.cs, 79
./UInt64Link.cs, 79
./UInt64LinkExtensions.cs, 81
./UInt64LinksExtensions.cs, 81
./UInt64LinksTransactionsLayer.cs, 82
```

./obj/Debug/netstandard2.0/Platform.Data.Doublets.AssemblyInfo.cs, 20