

# Namespace BelNytheraSeiche.WaveletMatrix

## Classes

### [AggregateSparseTable<T>](#)

Provides a general-purpose data structure for answering range aggregation queries on a static array. Preprocessing is  $O(N \log N)$ , and queries are  $O(\log N)$ .

### [BitSet](#)

Provides a mutable bit set for efficiently building large bit vectors.

### [BurrowsWheelerTransform](#)

Provides a static method for performing the Burrows-Wheeler Transform (BWT).

### [BurrowsWheelerTransform.BwtResult](#)

Encapsulates the result of a Burrows-Wheeler Transform operation.

### [FMIIndex](#)

Provides a high-performance full-text search index based on the FM-Index algorithm. This implementation supports both a full Suffix Array mode for fast locating and a memory-efficient sampling mode.

### [FMIIndex.Snippet](#)

Represents a snippet of text, including its content and position.

### [FischerHeunSparseTable<T>](#)

Provides a data structure for answering range queries on a static array in  $O(1)$  time after an  $O(N)$  preprocessing step, based on the Fischer-Heun structure. This implementation is the state-of-the-art solution for idempotent operations like finding the minimum or maximum.

### [ImmutableBitSet](#)

Represents a read-only, immutable bit set.

### [LcpIndex](#)

Provides an index structure over a Suffix Array's LCP (Longest Common Prefix) array to answer advanced stringology queries efficiently.

### [LcpIndex.Repeat](#)

Represents a repeated substring and all its occurrences.

### [LcpIndex.SimilarityMatcher](#)

Provides functionality to find all common substrings between two texts using a combined [LcpIndex](#). An instance of this class is created via the [CreateSimilarityMatcher\(string, string\)](#) factory method.

## [LcpIndex.SimilarityMatcher.Match](#)

Represents a common substring found between two texts.

## [LcpIndex.SimilarityMatcher.Palindrome](#)

Represents a palindromic substring found in the text.

## [LcpIndex.TandemRepeat](#)

Represents a tandem repeat found in the text.

## [LcpIndex.TextWithPosition](#)

Represents a substring with its starting position in the original text.

## [RankSelectBitSet](#)

An immutable bit set optimized for high-performance Rank and Select operations.

## [SerializationOptions](#)

Represents options for serialization.

## [SparseTable<T>](#)

Provides a data structure for answering range queries on a static array in O(1) time after an O(N log N) preprocessing step. This version is optimized for **idempotent** operations like finding the minimum or maximum, and it also tracks the index of the result.

## [SuffixArray](#)

Provides a powerful string searching data structure using a Suffix Array and LCP Array.

## [SuffixArray.WildcardOptions](#)

Provides options to customize the behavior of a wildcard search.

## [ValueBuffer<T>](#)

Represents a buffer for value types, implemented using a list of array chunks.

## [WaveletMatrixCore](#)

Provides the core, high-performance implementation of a Wavelet Matrix for integer sequences. This class is not generic and operates directly on coordinate-compressed integer arrays.

## [WaveletMatrixCore.ValueWithFrequency](#)

Represents a value and its frequency of occurrence.

## [WaveletMatrixGeneric<T>](#)

A generic, compressed data structure for sequences of any comparable type. It provides fast Rank, Select, Quantile, and other advanced queries. This class uses coordinate compression internally, making it highly memory-efficient for data with a small alphabet size.

## [WaveletMatrixGeneric<T>.ByteSerializer](#)

Provides a default serializer for the [byte](#) type.

### [WaveletMatrixGeneric<T>.CharSerializer](#)

Provides a default serializer for the [char](#) type.

### [WaveletMatrixGeneric<T>.Int32Serializer](#)

Provides a default serializer for the [int](#) type.

### [WaveletMatrixGeneric<T>.ValueWithFrequency<Tx>](#)

Represents a value and its frequency of occurrence.

## Structs

### [SparseTable<T>.ValueWithIndex](#)

Represents a value and its original index in the input array.

### [SuffixArray.Match](#)

Provides a single match found in a text.

### [SuffixArray.MatchGroup](#)

Provides a common substring found between two texts.

### [SuffixArray.MatchRepeated](#)

Provides a repeated substring and all its occurrences.

## Interfaces

### [WaveletMatrixGeneric<T>.IGenericSerializer<Tx>](#)

Defines the contract for serializing and deserializing elements of type [Tx](#).

## Enums

### [SortOrder](#)

Specifies the sort order for the results of a Locate operation.

# Class AggregateSparseTable<T>

Namespace: [BelNytheraSeiche.WaveletMatrix](#)

Assembly: BelNytheraSeiche.WaveletMatrix.dll

Provides a general-purpose data structure for answering range aggregation queries on a static array. Preprocessing is  $O(N \log N)$ , and queries are  $O(\log N)$ .

```
public sealed class AggregateSparseTable<T> where T : IComparable<T>
```

## Type Parameters

T

The type of elements in the array. Must be comparable.

## Inheritance

[object](#) ← AggregateSparseTable<T>

## Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

## Remarks

This class is suitable for **non-idempotent** (and associative) operations like Range Sum Query. Unlike [SparseTable<T>](#), it correctly handles operations where overlapping subproblems would lead to incorrect results. For faster prefix-sum-based queries, consider using a Fenwick Tree.

## Constructors

### AggregateSparseTable(ReadOnlySpan<T>, Func<T, T, T>)

Initializes a new instance of the [AggregateSparseTable<T>](#) class. The constructor performs preprocessing which takes  $O(N \log N)$  time and  $O(N \log N)$  space.

```
public AggregateSparseTable(ReadOnlySpan<T> array, Func<T, T, T> aggregator)
```

## Parameters

**array** [ReadOnlySpan](#)<T>

The static array of data to be queried.

**aggregator** [Func](#)<T, T>

A function that aggregates two elements (e.g., (a, b) => a + b for a sum).

## Exceptions

[ArgumentNullException](#)

Thrown if **aggregator** is null.

# Properties

## Size

Gets the total number of elements in the sequence.

```
public int Size { get; }
```

## Property Value

[int](#)

# Methods

## Query()

Queries the range [start, end) to get an aggregated value. This operation takes **O(log N)** time.

```
public T Query()
```

## Returns

T

The aggregated value.

## Remarks

This method is a shorthand for calling `Query(0, Size)`.

## `Query(int, int)`

Queries the range [start, end) to get an aggregated value. This operation takes **O(log N)** time.

```
public T Query(int start, int end)
```

## Parameters

`start` [int](#)

The inclusive, zero-based start of the range.

`end` [int](#)

The exclusive, zero-based end of the range.

## Returns

T

The aggregated value for the specified range.

## Exceptions

[ArgumentOutOfRangeException](#)

Thrown if `start` or `end` are outside the valid bounds.

[ArgumentException](#)

Thrown if `start` is greater than or equal `end`.

# Class BitSet

Namespace: [BelNytheraSeiche.WaveletMatrix](#)

Assembly: BelNytheraSeiche.WaveletMatrix.dll

Provides a mutable bit set for efficiently building large bit vectors.

```
public sealed class BitSet
```

## Inheritance

[object](#) ← BitSet

## Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

# Constructors

## BitSet(int)

Provides a mutable bit set for efficiently building large bit vectors.

```
public BitSet(int arrayAlignment = 65536)
```

## Parameters

**arrayAlignment** [int](#)

The alignment size for the internal buffer chunks. Must be a power of 2. Defaults to 65536.

# Properties

## ArrayAlignment

Gets the alignment size used by the internal buffer.

```
public int ArrayAlignment { get; }
```

Property Value

[int](#)

## Methods

### Add(bool)

Adds a bit to the end of the bit set.

```
public void Add(bool bit)
```

Parameters

[bit](#) [bool](#)

The bit to add ([true](#) for 1, [false](#) for 0).

### Get(int)

Gets the bit value at the specified index.

```
public bool Get(int index)
```

Parameters

[index](#) [int](#)

The zero-based index of the bit to get.

Returns

[bool](#)

The bit value ([true](#) for 1, [false](#) for 0) at the specified index.

## Exceptions

### [ArgumentOutOfRangeException](#)

`index` is less than 0 or greater than or equal to the number of bits in the set.

## ToImmutable()

Creates an immutable, read-only version of this bit set.

```
public ImmutableBitSet ToImmutable()
```

Returns

### [ImmutableBitSet](#)

A new [ImmutableBitSet](#) containing the current data.

## ToImmutable(BitSet)

Creates an immutable, read-only version of a specified [BitSet](#).

```
public static ImmutableBitSet ToImmutable(BitSet bitSet)
```

Parameters

### `bitSet` [BitSet](#)

The mutable bit set to convert.

Returns

### [ImmutableBitSet](#)

A new [ImmutableBitSet](#) containing the data from the provided bit set.

## Exceptions

### [ArgumentNullException](#)

`bitSet` is null.

## ToRankSelect(BitSet, bool)

Creates an immutable version of a specified [BitSet](#) optimized for high-performance Rank and Select operations.

```
public static RankSelectBitSet ToRankSelect(BitSet bitSet, bool createAuxDir = true)
```

### Parameters

`bitSet` [BitSet](#)

The mutable bit set to convert.

`createAuxDir` [bool](#) ↗

A value indicating whether to immediately create the auxiliary directories required for fast Rank and Select operations.

### Returns

[RankSelectBitSet](#)

A new [RankSelectBitSet](#) containing the data from the provided bit set and pre-calculated auxiliary indexes.

### Remarks

If `createAuxDir` is set to `false`, this method is very fast, but the returned [RankSelectBitSet](#) will not be ready for Rank/Select operations until its auxiliary directories are created and set via the [SetAuxDir\(\)](#) method. If set to `true` (the default), the auxiliary directories are created during this call, which takes more processing time but results in a fully initialized and ready-to-use object.

### Exceptions

[ArgumentNullException](#) ↗

`bitSet` is null.

## ToRankSelect(bool)

Creates an immutable version of a specified [BitSet](#), optimized for high-performance Rank and Select operations.

```
public RankSelectBitSet ToRankSelect(bool createAuxDir = true)
```

### Parameters

[createAuxDir](#) [bool](#) ↗

A value indicating whether to immediately create the auxiliary directories required for fast Rank and Select operations.

### Returns

[RankSelectBitSet](#)

A new [RankSelectBitSet](#) containing the data from the provided bit set and pre-calculated auxiliary indexes.

### Remarks

If [createAuxDir](#) is set to [false](#), this method is very fast, but the returned [RankSelectBitSet](#) will not be ready for Rank/Select operations until its auxiliary directories are created and set via the [SetAuxDir\(\)](#) method. If set to [true](#) (the default), the auxiliary directories are created during this call, which takes more processing time but results in a fully initialized and ready-to-use object.

# Class BurrowsWheelerTransform

Namespace: [BelNytheraSeiche.WaveletMatrix](#)

Assembly: BelNytheraSeiche.WaveletMatrix.dll

Provides a static method for performing the Burrows-Wheeler Transform (BWT).

```
public static class BurrowsWheelerTransform
```

## Inheritance

[object](#) ← BurrowsWheelerTransform

## Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

## Remarks

The BWT rearranges a string into runs of similar characters, which is useful for compression and indexing. This implementation uses a [SuffixArray](#) to efficiently compute the transform.

## Methods

### Transform(ReadOnlyMemory<char>)

Performs the Burrows-Wheeler Transform on the given text.

```
public static BurrowsWheelerTransform.BwtResult Transform(ReadOnlyMemory<char> text)
```

#### Parameters

**text** [ReadOnlyMemory](#)<[char](#)>

The input text to transform. For a correct and reversible transform, it is highly recommended that the text ends with a unique, lexicographically smallest character (terminator), such as '¥0'.

#### Returns

[BurrowsWheelerTransform.BwtResult](#)

A [BurrowsWheelerTransform.BwtResult](#) record containing the generated Suffix Array, the BWT string (L-column), and the zero-based index of the original string in the sorted rotation matrix.

# Class BurrowsWheelerTransform.BwtResult

Namespace: [BelNytheraSeiche.WaveletMatrix](#)

Assembly: BelNytheraSeiche.WaveletMatrix.dll

Encapsulates the result of a Burrows-Wheeler Transform operation.

```
public record BurrowsWheelerTransform.BwtResult :  
IEquatable<BurrowsWheelerTransform.BwtResult>
```

## Inheritance

[object](#) ← BurrowsWheelerTransform.BwtResult

## Implements

[IEquatable](#)<[BurrowsWheelerTransform.BwtResult](#)>

## Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

## Constructors

### BwtResult(SuffixArray, string, int)

Encapsulates the result of a Burrows-Wheeler Transform operation.

```
public BwtResult(SuffixArray SA, string BwtString, int OriginalIndex)
```

## Parameters

### SA [SuffixArray](#)

The Suffix Array generated during the transform.

### BwtString [string](#)

The resulting BWT string (often called the L-column).

### OriginalIndex [int](#)

The zero-based index of the original string in the conceptual sorted matrix of rotations. This is required for the inverse transform.

## Properties

### BwtString

The resulting BWT string (often called the L-column).

```
public string BwtString { get; init; }
```

Property Value

[string](#) ↗

### OriginalIndex

The zero-based index of the original string in the conceptual sorted matrix of rotations. This is required for the inverse transform.

```
public int OriginalIndex { get; init; }
```

Property Value

[int](#) ↗

### SA

The Suffix Array generated during the transform.

```
public SuffixArray SA { get; init; }
```

Property Value

[SuffixArray](#)

# Class FMIndex

Namespace: [BelNytheraSeiche.WaveletMatrix](#)

Assembly: BelNytheraSeiche.WaveletMatrix.dll

Provides a high-performance full-text search index based on the FM-Index algorithm. This implementation supports both a full Suffix Array mode for fast locating and a memory-efficient sampling mode.

```
public sealed class FMIndex
```

## Inheritance

[object](#) ← FMIndex

## Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

## Remarks

The FM-Index is a compressed full-text index that allows for fast counting ([Count\(ReadOnlySpan<char>\)](#)) and locating ([Locate\(ReadOnlySpan<char>, SortOrder\)](#)) of patterns within a large text. It is constructed using the Burrows-Wheeler Transform (BWT), a Wavelet Matrix, and a Suffix Array.

## Properties

### IsSampled

Gets a value indicating whether the index is using Suffix Array sampling.

```
public bool IsSampled { get; }
```

### Property Value

[bool](#)

### SA

Gets the underlying Suffix Array instance used by this index. This property is only available when the index is built in full Suffix Array mode (i.e., when [IsSampled](#) is [false](#)).

```
public SuffixArray SA { get; }
```

## Property Value

[SuffixArray](#)

## Exceptions

[InvalidOperationException](#)

Thrown if the index is in sampling mode.

## Text

Gets the original source text used to build the index, excluding the internal terminator character. This property is only available when the index is built in full Suffix Array mode (i.e., when [IsSampled](#) is [false](#)).

```
public ReadOnlyMemory<char> Text { get; }
```

## Property Value

[ReadOnlyMemory](#)<[char](#)>

## Exceptions

[InvalidOperationException](#)

Thrown if the index is in sampling mode.

## WM

Gets the underlying Wavelet Matrix instance used by this index.

```
public WaveletMatrixGeneric<char> WM { get; }
```

## Property Value

[WaveletMatrixGeneric<char>](#)

## Methods

### Count(ReadOnlySpan<char>)

Counts the number of occurrences of a pattern within the text. This operation is extremely fast, with performance proportional to the pattern length, not the text length.

```
public int Count(ReadOnlySpan<char> pattern)
```

#### Parameters

**pattern** [ReadOnlySpan<char>](#)

The pattern to search for.

#### Returns

[int](#)

The total number of non-overlapping occurrences of the pattern.

### Create(ReadOnlyMemory<char>, int)

Creates a new instance of the [FMIndex](#) from a read-only memory segment of characters, with an optional Suffix Array sampling rate. An internal terminator character ('\0') is appended to the text for correctness.

```
public static FMIndex Create(ReadOnlyMemory<char> text, int sampleRate = 0)
```

#### Parameters

**text** [ReadOnlyMemory<char>](#)

The text to be indexed.

## sampleRate [int](#)

The sampling rate for the Suffix Array. If 0 (default), the full Suffix Array is stored for fast locating. If greater than 1, only every N-th entry of the Suffix Array is stored, saving memory at the cost of slower locate operations.

## Returns

### [FMIndex](#)

A new, fully initialized [FMIndex](#) instance.

## Exceptions

### [ArgumentOutOfRangeException](#)

Thrown if `sampleRate` is less than 0.

## Create(string, int)

Creates a new instance of the [FMIndex](#) from a string, with an optional Suffix Array sampling rate.

```
public static FMIndex Create(string text, int sampleRate = 0)
```

## Parameters

### `text` [string](#)

The text to be indexed.

### `sampleRate` [int](#)

The sampling rate for the Suffix Array. If 0 (default), the full Suffix Array is stored for fast locating. If greater than 1, only every N-th entry of the Suffix Array is stored to save memory, at the cost of slower locate operations.

## Returns

### [FMIndex](#)

A new, fully initialized [FMIndex](#) instance.

## Exceptions

### [ArgumentNullException](#)

Thrown if `text` is null.

### [ArgumentOutOfRangeException](#)

Thrown if `sampleRate` is less than 0.

## Deserialize(byte[])

Deserializes a byte array into a [FMIndex](#) instance.

```
public static FMIndex Deserialize(byte[] data)
```

## Parameters

### `data` [byte](#)[]

The byte array containing the serialized data.

## Returns

### [FMIndex](#)

A new, deserialized instance of [FMIndex](#).

## Exceptions

### [ArgumentNullException](#)

Thrown if `data` is null.

## Deserialize(Stream)

Deserializes a [FMIndex](#) instance from a stream. It verifies the file format, type identifier, and checksum.

```
public static FMIndex Deserialize(Stream stream)
```

## Parameters

### `stream Stream` ↗

The stream to read the serialized data from.

## Returns

### [FMIndex](#)

A new, deserialized instance.

## Exceptions

### [ArgumentNullException](#) ↗

Thrown if `stream` is null.

### [InvalidDataException](#) ↗

Thrown if the data format is unsupported, the type is incompatible, or the data is corrupt.

## Deserialize(string)

Deserializes a [FMIndex](#) instance from the specified file.

```
public static FMIndex Deserialize(string file)
```

## Parameters

### `file string` ↗

The path of the file to read from.

## Returns

### [FMIndex](#)

A new, deserialized instance.

## Exceptions

## [ArgumentNullException](#)

Thrown if `file` is null.

## GetSnippet(int, int, int, double)

Calculates the location and optionally extracts the content of a text snippet surrounding a specified position.

```
public FMIndex.Snippet GetSnippet(int position, int keyLength, int totalLength, double leadingRatio = 0.5)
```

### Parameters

`position` [int](#)

The zero-based starting position of the keyword in the original text.

`keyLength` [int](#)

The length of the keyword.

`totalLength` [int](#)

The desired total length of the snippet.

`leadingRatio` [double](#)

The desired proportion (0.0 to 1.0) of the context text to appear before the keyword. Defaults to 0.5 for centering the keyword.

### Returns

[FMIndex.Snippet](#)

A [FMIndex.Snippet](#) record containing the location, length, and (in non-sampling mode) the text content of the snippet.

### Remarks

In full Suffix Array mode, the [Text](#) property of the returned record will contain the extracted string. In sampling mode, the [Text](#) property will be `null`, but the location properties ([Index](#), [Length](#), [KeyPosition](#)) will be correctly calculated, allowing the caller to extract the snippet from their own copy of the source

text. The specified `leadingRatio` is a guideline. The method prioritizes returning a snippet of approximately `totalLength`. If the keyword is near the start or end of the text, the actual ratio of leading/trailing context will be adjusted to fill the requested length.

## Exceptions

### [ArgumentOutOfRangeException](#)

Thrown if `position`, `keyLength`, `totalLength` are invalid, or if `leadingRatio` is not between 0.0 and 1.0.

### [ArgumentException](#)

Thrown if `position + keyLength` is greater than the length of the text.

## Locate(ReadOnlySpan<char>, SortOrder)

Finds all starting positions of a pattern within the text.

```
public IEnumerable<int> Locate(ReadOnlySpan<char> pattern, SortOrder sortOrder  
= SortOrder.Ascending)
```

## Parameters

### `pattern` [ReadOnlySpan](#)<`char`>

The pattern to search for.

### `sortOrder` [SortOrder](#)

Specifies the order of the returned positions. Defaults to [Ascending](#).

## Returns

### [IEnumerable](#)<`int`>

An enumerable collection of the zero-based starting positions of all occurrences.

## Remarks

In full Suffix Array mode, this operation is very fast. In sampling mode, locating each position requires additional computation (LF-mapping steps), making it slower but significantly more memory-efficient.

Specifying [Ascending](#) or [Descending](#) may require collecting all results and sorting them, which incurs a performance cost. For the highest performance where order is not important, use [Unordered](#).

## RestoreSourceText()

Reconstructs the original source text from the compressed index by inverting the Burrows-Wheeler Transform.

```
public string RestoreSourceText()
```

Returns

[string](#)

The original text that was used to create the index.

## Serialize(FMIndex, SerializationOptions?)

Serializes the [FMIndex](#) instance into a byte array.

```
public static byte[] Serialize(FMIndex obj, SerializationOptions? options = null)
```

Parameters

**obj** [FMIndex](#)

The instance to serialize.

**options** [SerializationOptions](#)

Serialization options, such as the compression level. If null, [Default](#) will be used.

Returns

[byte](#)[]

A byte array containing the serialized data.

Exceptions

## [ArgumentNullException](#)

Thrown if `obj` is null.

## Serialize(FMIndex, Stream, SerializationOptions?)

Serializes the [FMIndex](#) instance to a stream. The data is compressed using Brotli and includes a checksum for integrity verification.

```
public static void Serialize(FMIndex obj, Stream stream, SerializationOptions? options = null)
```

### Parameters

#### `obj` [FMIndex](#)

The instance to serialize.

#### `stream` [Stream](#)

The stream to write the serialized data to.

#### `options` [SerializationOptions](#)

Serialization options, such as the compression level. If null, [Default](#) will be used.

### Exceptions

## [ArgumentNullException](#)

Thrown if `obj` or `stream` is null.

## Serialize(FMIndex, string, SerializationOptions?)

Serializes the [FMIndex](#) instance to the specified file.

```
public static void Serialize(FMIndex obj, string file, SerializationOptions? options = null)
```

### Parameters

**obj** [FMIIndex](#)

The instance to serialize.

**file** [string](#)

The path of the file to write to.

**options** [SerializationOptions](#)

Serialization options, such as the compression level. If null, [Default](#) will be used.

## Exceptions

[ArgumentNullException](#)

Thrown if **obj** or **file** is null.

# Class FMIndex.Snippet

Namespace: [BelNytheraSeiche.WaveletMatrix](#)

Assembly: BelNytheraSeiche.WaveletMatrix.dll

Represents a snippet of text, including its content and position.

```
public record FMIndex.Snippet : IEquatable<FMIndex.Snippet>
```

## Inheritance

[object](#) ← FMIndex.Snippet

## Implements

[IEquatable](#)<[FMIndex.Snippet](#)>

## Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

# Constructors

## Snippet(string?, int, int, int)

Represents a snippet of text, including its content and position.

```
public Snippet(string? Text, int Index, int Length, int KeyPosition)
```

## Parameters

### Text [string](#)

The text content of the snippet. This will be [null](#) if the [FMIndex](#) was created in sampling mode.

### Index [int](#)

The zero-based starting position of the snippet within the original source text.

### Length [int](#)

The total length of the snippet.

## KeyPosition [int ↗](#)

The zero-based starting position of the keyword within this snippet's text.

# Properties

## Index

The zero-based starting position of the snippet within the original source text.

```
public int Index { get; init; }
```

### Property Value

[int ↗](#)

## KeyPosition

The zero-based starting position of the keyword within this snippet's text.

```
public int KeyPosition { get; init; }
```

### Property Value

[int ↗](#)

## Length

The total length of the snippet.

```
public int Length { get; init; }
```

### Property Value

[int ↗](#)

## Text

The text content of the snippet. This will be `null` if the [FMIIndex](#) was created in sampling mode.

```
public string? Text { get; init; }
```

Property Value

[string](#) ↗

# Class FischerHeunSparseTable<T>

Namespace: [BelNytheraSeiche.WaveletMatrix](#)

Assembly: BelNytheraSeiche.WaveletMatrix.dll

Provides a data structure for answering range queries on a static array in  $O(1)$  time after an  $O(N)$  preprocessing step, based on the Fischer-Heun structure. This implementation is the state-of-the-art solution for idempotent operations like finding the minimum or maximum.

```
public sealed class FischerHeunSparseTable<T> where T : IComparable<T>
```

## Type Parameters

T

The type of elements in the array. Must be comparable.

## Inheritance

[object](#) ← FischerHeunSparseTable<T>

## Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

## Remarks

This advanced algorithm achieves  $O(1)$  query time with  $O(N)$  space by dividing the input array into small blocks. It precomputes answers for all query patterns within the small blocks and uses a standard [SparseTable<T>](#) to answer queries across blocks. This makes it one of the asymptotically fastest solutions for the static Range Minimum/Maximum Query (RMQ) problem.

## Constructors

FischerHeunSparseTable(ReadOnlyMemory<T>, Func<T, T, bool>)

Initializes a new instance of the [FischerHeunSparseTable<T>](#) class. The constructor performs preprocessing which takes  $O(N)$  time and space.

```
public FischerHeunSparseTable(ReadOnlyMemory<T> memory, Func<T, T, bool> comparer)
```

## Parameters

`memory` [ReadOnlyMemory](#)<T>

The static array of data to be queried, represented as a `ReadOnlyMemory`.

`comparer` [Func](#)<T, T, bool>

A function that compares two elements. It should return `true` if the first argument is considered "better" than the second (e.g., for a minimum query, the comparer would be `(a, b) => a.CompareTo(b) < 0`).

## Exceptions

[ArgumentNullException](#)

Thrown if `comparer` is null.

# Properties

## Size

Gets the total number of elements in the sequence.

```
public int Size { get; }
```

## Property Value

[int](#)

# Methods

## Query()

Queries to find the best value according to the comparer and its original index. This operation takes O(1) time.

```
public SparseTable<T>.ValueWithIndex Query()
```

Returns

[SparseTable<T>.ValueWithIndex](#)

A [SparseTable<T>.ValueWithIndex](#) record containing the best value and its original index in the input array.

Remarks

This method is a shorthand for calling `Query(0, Size)`.

## Query(int, int)

Queries the range [start, end) to find the best value and its original index. This operation takes O(1) time.

```
public SparseTable<T>.ValueWithIndex Query(int start, int end)
```

Parameters

`start` [int](#)

The inclusive, zero-based start of the range.

`end` [int](#)

The exclusive, zero-based end of the range.

Returns

[SparseTable<T>.ValueWithIndex](#)

A [SparseTable<T>.ValueWithIndex](#) record containing the best value and its original index in the input array.

Exceptions

[ArgumentOutOfRangeException](#)

Thrown if `start` or `end` are outside the valid bounds.

## [ArgumentException](#)

Thrown if `start` is greater than or equal to `end`.

# Class ImmutableBitSet

Namespace: [BelNytheraSeiche.WaveletMatrix](#)

Assembly: BelNytheraSeiche.WaveletMatrix.dll

Represents a read-only, immutable bit set.

```
public class ImmutableBitSet
```

## Inheritance

[object](#) ← ImmutableBitSet

## Derived

[RankSelectBitSet](#)

## Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

# Constructors

## ImmutableBitSet(ulong[], int)

Represents a read-only, immutable bit set.

```
public ImmutableBitSet(ulong[] buffer, int count)
```

## Parameters

**buffer** [ulong](#)[]

The underlying ulong array that stores the bits.

**count** [int](#)

The total number of bits in the set.

# Properties

## Buffer

Gets the underlying ulong array that stores the bits.

```
public ulong[] Buffer { get; }
```

Property Value

[ulong](#)[]

## Count

Gets the total number of bits in the set.

```
public int Count { get; }
```

Property Value

[int](#)[]

## this[int]

Gets the bit value at the specified index.

```
public bool this[int index] { get; }
```

Parameters

[index](#) [int](#)[]

The zero-based index of the bit to get.

Property Value

[bool](#)[]

The bit value ([true](#) for 1, [false](#) for 0).

# Exceptions

## [ArgumentOutOfRangeException](#)

`index` is out of range.

# Methods

## At(int)

Gets the bit value at the specified index.

```
public bool At(int index)
```

### Parameters

#### `index` [int](#)

The zero-based index of the bit to get.

### Returns

#### [bool](#)

The bit value (`true` for 1, `false` for 0).

## ToRankSelect(ImmutableBitSet, bool)

Creates a new [RankSelectBitSet](#) from the specified [ImmutableBitSet](#).

```
public static RankSelectBitSet ToRankSelect(ImmutableBitSet bitSet, bool createAuxDir  
= true)
```

### Parameters

#### `bitSet` [ImmutableBitSet](#)

The source immutable bit set to convert.

#### `createAuxDir` [bool](#)

A value indicating whether to immediately create the auxiliary directories required for fast Rank and Select operations.

## Returns

### [RankSelectBitSet](#)

A new [RankSelectBitSet](#) instance.

## Remarks

If `createAuxDir` is set to `false`, this method is very fast, but the returned [RankSelectBitSet](#) will not be ready for Rank/Select operations until its auxiliary directories are created and set via the [SetAuxDir\(\)](#) method. If set to `true` (the default), the auxiliary directories are created during this call, which takes more processing time but results in a fully initialized and ready-to-use object.

## Exceptions

### [ArgumentNullException](#)

`bitSet` is null.

## ToRankSelect(bool)

Creates a new [RankSelectBitSet](#) from the specified [ImmutableBitSet](#).

```
public RankSelectBitSet ToRankSelect(bool createAuxDir = true)
```

## Parameters

### `createAuxDir` [bool](#)

A value indicating whether to immediately create the auxiliary directories required for fast Rank and Select operations.

## Returns

### [RankSelectBitSet](#)

A new [RankSelectBitSet](#) instance.

## Remarks

If `createAuxDir` is set to `false`, this method is very fast, but the returned [RankSelectBitSet](#) will not be ready for Rank/Select operations until its auxiliary directories are created and set via the [SetAuxDir\(\)](#) method. If set to `true` (the default), the auxiliary directories are created during this call, which takes more processing time but results in a fully initialized and ready-to-use object.

# Class LcpIndex

Namespace: [BelNytheraSeiche.WaveletMatrix](#)

Assembly: BelNytheraSeiche.WaveletMatrix.dll

Provides an index structure over a Suffix Array's LCP (Longest Common Prefix) array to answer advanced stringology queries efficiently.

```
public sealed class LcpIndex
```

## Inheritance

[object](#) ← LcpIndex

## Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

## Remarks

This class builds a [FischerHeunSparseTable<T>](#) on the LCP array to enable O(1) LCP queries between any two suffixes after an O(N) preprocessing step. It is a powerful tool for complex string analysis, such as finding repeated substrings or calculating string complexity.

## Properties

### SA

Gets the original source Suffix Array.

```
public SuffixArray SA { get; }
```

### Property Value

[SuffixArray](#)

### Text

Gets the original source text in Suffix Array.

```
public ReadOnlyMemory<char> Text { get; }
```

## Property Value

[ReadOnlyMemory](#)<char>

## Methods

### CalculateZivLempelComplexity()

Calculates the Ziv-Lempel 78 (LZ78) complexity of the text. This value represents the number of phrases in the LZ78 parsing of the string, indicating its compressibility.

```
public int CalculateZivLempelComplexity()
```

## Returns

[int](#)

The LZ78 complexity of the text.

### CountOccurrences(ReadOnlySpan<char>)

Efficiently counts the occurrences of a specified pattern within the text. It leverages binary search over the suffix array to quickly find the range, ensuring high performance even with large texts.

```
public int CountOccurrences(ReadOnlySpan<char> pattern)
```

## Parameters

**pattern** [ReadOnlySpan](#)<char>

The string pattern to search for.

## Returns

[int](#)

The number of times the pattern appears in the text.

## CountUniqueSubstrings()

Counts the total number of unique substrings within the text. This is calculated in O(N) time using the suffix and LCP arrays.

```
public long CountUniqueSubstrings()
```

Returns

[long](#)

The total count of unique substrings.

## Create(SuffixArray)

Creates a new instance of the [LcpIndex](#) for a given [SuffixArray](#).

```
public static LcpIndex Create(SuffixArray sa)
```

Parameters

[sa SuffixArray](#)

The Suffix Array to build the LCP index upon.

Returns

[LcpIndex](#)

A new, fully initialized [LcpIndex](#) instance.

Exceptions

[ArgumentNullException](#)

Thrown if [sa](#) is null.

## CreateSimilarityMatcher(string, string)

Creates a new [LcpIndex.SimilarityMatcher](#) instance to find common substrings between two texts. This is achieved by concatenating the two texts with a unique separator and building a single LCP index on the result.

```
public static LcpIndex.SimilarityMatcher CreateSimilarityMatcher(string text1, string text2)
```

### Parameters

**text1** [string](#)

The first text to compare.

**text2** [string](#)

The second text to compare.

### Returns

[LcpIndex.SimilarityMatcher](#)

A new [LcpIndex.SimilarityMatcher](#) instance ready for finding matches.

### Exceptions

[ArgumentNullException](#)

Thrown if **text1** or **text2** is null.

## CreateSimilarityMatcherForPalindrome(string)

Creates a new [LcpIndex.SimilarityMatcher](#) instance specifically configured to find palindromic substrings within a single text. This is achieved by concatenating the text with its reverse, separated by a unique character, and building an LCP index on the combined result.

```
public static LcpIndex.SimilarityMatcher CreateSimilarityMatcherForPalindrome(string text)
```

### Parameters

**text** [string](#)

The text in which to find palindromes.

Returns

[LcpIndex.SimilarityMatcher](#)

A new [LcpIndex.SimilarityMatcher](#) instance ready for palindrome detection.

Remarks

The returned matcher will have its [Text1](#) property set to the original **text** and its [Text2](#) property set to the reversed version of the text.

Exceptions

[ArgumentNullException](#)

Thrown if **text** is null.

## Deserialize(byte[])

Deserializes a byte array into a [LcpIndex](#) instance.

```
public static LcpIndex Deserialize(byte[] data)
```

Parameters

**data** [byte](#)[]

The byte array containing the serialized data.

Returns

[LcpIndex](#)

A new, deserialized instance of [LcpIndex](#).

Exceptions

[ArgumentNullException](#)

Thrown if `data` is null.

## Deserialize(Stream)

Deserializes a [LcpIndex](#) instance from a stream. It verifies the file format and checksum, and reconstructs the internal Sparse Table and Suffix Array.

```
public static LcpIndex Deserialize(Stream stream)
```

### Parameters

`stream` [Stream](#)

The stream to read the serialized data from.

### Returns

[LcpIndex](#)

A new, deserialized instance.

### Exceptions

[ArgumentNullException](#)

Thrown if `stream` is null.

[InvalidDataException](#)

Thrown if the data format is unsupported, the type is incompatible, or the data is corrupt.

## Deserialize(string)

Deserializes a [LcpIndex](#) instance from the specified file.

```
public static LcpIndex Deserialize(string file)
```

### Parameters

`file` `string` ↗

The path of the file to read from.

Returns

[LcpIndex](#)

A new, deserialized instance.

Exceptions

[ArgumentNullException](#) ↗

Thrown if `file` is null.

## FindAllShortestUniqueSubstrings()

Finds all shortest substrings that appear only once in the text. There can be multiple unique substrings of the same shortest length.

```
public IEnumerable<LcpIndex.TextWithPosition> FindAllShortestUniqueSubstrings()
```

Returns

[IEnumerable](#) ↗ <[LcpIndex.TextWithPosition](#)>

An enumerable collection of [LcpIndex.TextWithPosition](#) records, each representing one of the shortest unique substrings found. Returns an empty collection if no unique substring exists.

## FindLongestRepeats()

Finds the longest substring that appears at least twice in the text. This is achieved by finding the maximum value in the LCP array. If no substring is repeated, returns null.

```
public LcpIndex.Repeat? FindLongestRepeats()
```

Returns

## [LcpIndex.Repeat](#)

A [LcpIndex.Repeat](#) record representing the longest repeated substring and its positions, or null if no repeats exist.

## FindRepeats(int)

Finds all unique substrings that are repeated anywhere in the text (adjacent or non-adjacent) and meet a minimum length requirement.

```
public IEnumerable<LcpIndex.Repeat> FindRepeats(int minLength = 2)
```

### Parameters

**minLength** [int](#)

The minimum length of the repeated substrings to find.

### Returns

[IEnumerable](#) <[LcpIndex.Repeat](#)>

An enumerable collection of [LcpIndex.Repeat](#) records.

### Exceptions

[ArgumentOutOfRangeException](#)

Thrown if **minLength** is less than or equal 0.

## FindShortestUniqueSubstring()

Finds the shortest substring that appears only once in the text. This is determined by analyzing the LCP (Longest Common Prefix) values between adjacent suffixes in the suffix array.

```
public LcpIndex.TextWithPosition? FindShortestUniqueSubstring()
```

### Returns

## [LcpIndex.TextWithPosition](#)

A [LcpIndex.TextWithPosition](#) record containing the shortest unique substring and its starting position. Returns null if no unique substring exists (e.g., if the text is empty or consists of a single repeating character).

## FindTandemRepeats()

Finds all maximal tandem repeats ( substrings of the form `sss...` ) within the text.

```
public IEnumerable<LcpIndex.TandemRepeat> FindTandemRepeats()
```

Returns

[IEnumerable](#) <[LcpIndex.TandemRepeat](#)>

An enumerable collection of [LcpIndex.TandemRepeat](#) records, ordered by position and length.

## GetLcp(int, int)

Gets the length of the Longest Common Prefix (LCP) between the suffixes starting at `index1` and `index2`. This operation is performed in O(1) time.

```
public int GetLcp(int index1, int index2)
```

Parameters

`index1` [int](#)

The zero-based starting position of the first suffix in the original text.

`index2` [int](#)

The zero-based starting position of the second suffix in the original text.

Returns

[int](#)

The length of the longest common prefix.

## Exceptions

### [ArgumentException](#)

Thrown if `index1` or `index2` are out of the valid range.

## Locate(ReadOnlySpan<char>, SortOrder)

Finds all starting positions where the specified pattern occurs in the text.

```
public IEnumerable<int> Locate(ReadOnlySpan<char> pattern, SortOrder sortOrder  
= SortOrder.Ascending)
```

## Parameters

### `pattern` [ReadOnlySpan](#)<char>

The string pattern to search for.

### `sortOrder` [SortOrder](#)

Specifies the sort order for the resulting positions. The default is [Ascending](#).

## Returns

### [IEnumerable](#)<int>

An [IEnumerable](#)<T> that enumerates the starting positions (0-based indices) of the pattern's occurrences. Returns an empty sequence if the pattern is not found.

## Remarks

Specifying [Unordered](#) provides the fastest result by skipping any sorting. This option is recommended when the order of occurrences is not important.

## Exceptions

### [ArgumentException](#)

Thrown if an invalid `sortOrder` is provided.

## Serialize(LcpIndex, SerializationOptions?)

Serializes the [LcpIndex](#) instance into a byte array.

```
public static byte[] Serialize(LcpIndex obj, SerializationOptions? options = null)
```

Parameters

**obj** [LcpIndex](#)

The instance to serialize.

**options** [SerializationOptions](#)

Serialization options, such as the compression level. If null, [Default](#) will be used.

Returns

[byte](#)[]

A byte array containing the serialized data.

Exceptions

[ArgumentNullException](#)

Thrown if **obj** is null.

## Serialize(LcpIndex, Stream, SerializationOptions?)

Serializes the [LcpIndex](#) instance to a stream. This method saves the pre-computed Sparse Table and the associated Suffix Array. The data is compressed using Brotli and includes a checksum for integrity verification.

```
public static void Serialize(LcpIndex obj, Stream stream, SerializationOptions? options  
= null)
```

Parameters

**obj** [LcpIndex](#)

The instance to serialize.

#### **stream** [Stream](#)

The stream to write the serialized data to.

#### **options** [SerializationOptions](#)

Serialization options, such as the compression level. If null, [Default](#) will be used.

## Exceptions

#### [ArgumentNullException](#)

Thrown if **obj** or **stream** is null.

## Serialize(LcpIndex, string, SerializationOptions?)

Serializes the [LcpIndex](#) instance to the specified file.

```
public static void Serialize(LcpIndex obj, string file, SerializationOptions? options = null)
```

## Parameters

#### **obj** [LcpIndex](#)

The instance to serialize.

#### **file** [string](#)

The path of the file to write to.

#### **options** [SerializationOptions](#)

Serialization options, such as the compression level. If null, [Default](#) will be used.

## Exceptions

#### [ArgumentNullException](#)

Thrown if **obj** or **file** is null.

# Class LcpIndex.Repeat

Namespace: [BelNytheraSeiche.WaveletMatrix](#)

Assembly: BelNytheraSeiche.WaveletMatrix.dll

Represents a repeated substring and all its occurrences.

```
public record LcpIndex.Repeat : IEquatable<LcpIndex.Repeat>
```

## Inheritance

[object](#) ← LcpIndex.Repeat

## Implements

[IEquatable](#)<[LcpIndex.Repeat](#)>

## Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

# Constructors

## Repeat(string, int[])

Represents a repeated substring and all its occurrences.

```
public Repeat(string Text, int[] Positions)
```

## Parameters

**Text** [string](#)

The repeating unit string.

**Positions** [int](#)[]

An array of all 0-based starting positions where the substring occurs.

# Properties

## Positions

An array of all 0-based starting positions where the substring occurs.

```
public int[] Positions { get; init; }
```

Property Value

[int](#)[]

## Text

The repeating unit string.

```
public string Text { get; init; }
```

Property Value

[string](#)

# Class LcpIndex.SimilarityMatcher

Namespace: [BelNytheraSeiche.WaveletMatrix](#)

Assembly: BelNytheraSeiche.WaveletMatrix.dll

Provides functionality to find all common substrings between two texts using a combined [LcpIndex](#). An instance of this class is created via the [CreateSimilarityMatcher\(string, string\)](#) factory method.

```
public sealed class LcpIndex.SimilarityMatcher
```

## Inheritance

[object](#) ← LcpIndex.SimilarityMatcher

## Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

## Constructors

### SimilarityMatcher(LcpIndex, string, string)

Provides functionality to find all common substrings between two texts using a combined [LcpIndex](#). An instance of this class is created via the [CreateSimilarityMatcher\(string, string\)](#) factory method.

```
public SimilarityMatcher(LcpIndex lcpIndex, string text1, string text2)
```

## Parameters

### [lcpIndex](#) [LcpIndex](#)

The LcpIndex built on the combined text.

### [text1](#) [string](#)

The first original text.

### [text2](#) [string](#)

The second original text.

# Properties

## LcpIndex

Gets the underlying [LcpIndex](#) built on the combined text.

```
public LcpIndex LcpIndex { get; }
```

### Property Value

[LcpIndex](#)

## Text1

Gets the first original text used for the comparison.

```
public string Text1 { get; }
```

### Property Value

[string](#)

## Text2

Gets the second original text used for the comparison.

```
public string Text2 { get; }
```

### Property Value

[string](#)

# Methods

## FindPalindrome()

Finds the longest palindromic substring within the original text. This method requires the [LcpIndex.SimilarityMatcher](#) to be created via [CreateSimilarityMatcherForPalindrome\(string\)](#). It operates by finding the longest common substring between the text and its reverse.

```
public LcpIndex.SimilarityMatcher.Palindrome? FindPalindrome()
```

Returns

[LcpIndex.SimilarityMatcher.Palindrome](#)

A [LcpIndex.SimilarityMatcher.Palindrome](#) record for the longest palindrome, or null if no palindrome of length 2 or more exists.

## LongestMatch()

Finds the longest common substring between the two texts.

```
public LcpIndex.SimilarityMatcher.Match? LongestMatch()
```

Returns

[LcpIndex.SimilarityMatcher.Match](#)

A [LcpIndex.SimilarityMatcher.Match](#) record representing the longest common substring, or null if no common part exists.

## Matches(int)

Finds all common substrings between the two texts that are at least a specified minimum length. This method works by scanning the LCP array of the combined text and identifying adjacent suffixes that originate from different source texts.

```
public IEnumerable<LcpIndex.SimilarityMatcher.Match> Matches(int minLength = 2)
```

Parameters

`minLength` [int](#)

The minimum length for a substring to be reported as a match. Defaults to 2.

## Returns

[IEnumerable](#) <[LcpIndex.SimilarityMatcher.Match](#)>

An enumerable collection of [LcpIndex.SimilarityMatcher.Match](#) records, each representing a common substring.

## Exceptions

[ArgumentOutOfRangeException](#)

Thrown if `minLength` is less than 0.

# Class LcpIndex.SimilarityMatcher.Match

Namespace: [BelNytheraSeiche.WaveletMatrix](#)

Assembly: BelNytheraSeiche.WaveletMatrix.dll

Represents a common substring found between two texts.

```
public record LcpIndex.SimilarityMatcher.Match :  
IEquatable<LcpIndex.SimilarityMatcher.Match>
```

## Inheritance

[object](#) ← LcpIndex.SimilarityMatcher.Match

## Implements

[IEquatable](#)<[LcpIndex.SimilarityMatcher.Match](#)>

## Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

## Constructors

### Match(int, int, int)

Represents a common substring found between two texts.

```
public Match(int Position1, int Position2, int Length)
```

#### Parameters

**Position1** [int](#)

The zero-based starting position of the match in the first text.

**Position2** [int](#)

The zero-based starting position of the match in the second text.

**Length** [int](#)

The length of the common substring.

## Properties

### Length

The length of the common substring.

```
public int Length { get; init; }
```

Property Value

[int↗](#)

### Position1

The zero-based starting position of the match in the first text.

```
public int Position1 { get; init; }
```

Property Value

[int↗](#)

### Position2

The zero-based starting position of the match in the second text.

```
public int Position2 { get; init; }
```

Property Value

[int↗](#)

# Class LcpIndex.SimilarityMatcher.Palindrome

Namespace: [BelNytheraSeiche.WaveletMatrix](#)

Assembly: BelNytheraSeiche.WaveletMatrix.dll

Represents a palindromic substring found in the text.

```
public record LcpIndex.SimilarityMatcher.Palindrome :  
IEquatable<LcpIndex.SimilarityMatcher.Palindrome>
```

## Inheritance

[object](#) ← LcpIndex.SimilarityMatcher.Palindrome

## Implements

[IEquatable](#)<[LcpIndex.SimilarityMatcher.Palindrome](#)>

## Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

## Constructors

### Palindrome(int, int)

Represents a palindromic substring found in the text.

```
public Palindrome(int Position, int Length)
```

#### Parameters

##### Position [int](#)

The 0-based starting position of the palindrome in the original text.

##### Length [int](#)

The length of the palindrome.

# Properties

## Length

The length of the palindrome.

```
public int Length { get; init; }
```

Property Value

[int ↗](#)

## Position

The 0-based starting position of the palindrome in the original text.

```
public int Position { get; init; }
```

Property Value

[int ↗](#)

# Class LcpIndex.TandemRepeat

Namespace: [BelNytheraSeiche.WaveletMatrix](#)

Assembly: BelNytheraSeiche.WaveletMatrix.dll

Represents a tandem repeat found in the text.

```
public record LcpIndex.TandemRepeat : IEquatable<LcpIndex.TandemRepeat>
```

## Inheritance

[object](#) ← LcpIndex.TandemRepeat

## Implements

[IEquatable](#)<[LcpIndex.TandemRepeat](#)>

## Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

## Constructors

### TandemRepeat(int, int, int)

Represents a tandem repeat found in the text.

```
public TandemRepeat(int Position, int Length, int Count)
```

## Parameters

### Position [int](#)

The 0-based starting position of the entire repeat block.

### Length [int](#)

The length of the repeating unit string.

### Count [int](#)

The number of times the unit repeats (will be 2 or more).

# Properties

## Count

The number of times the unit repeats (will be 2 or more).

```
public int Count { get; init; }
```

### Property Value

[int↗](#)

## Length

The length of the repeating unit string.

```
public int Length { get; init; }
```

### Property Value

[int↗](#)

## Position

The 0-based starting position of the entire repeat block.

```
public int Position { get; init; }
```

### Property Value

[int↗](#)

# Class LcpIndex.TextWithPosition

Namespace: [BelNytheraSeiche.WaveletMatrix](#)

Assembly: BelNytheraSeiche.WaveletMatrix.dll

Represents a substring with its starting position in the original text.

```
public record LcpIndex.TextWithPosition : IEquatable<LcpIndex.TextWithPosition>
```

## Inheritance

[object](#) ← LcpIndex.TextWithPosition

## Implements

[IEquatable](#)<[LcpIndex.TextWithPosition](#)>

## Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

# Constructors

## TextWithPosition(string, int)

Represents a substring with its starting position in the original text.

```
public TextWithPosition(string Text, int Position)
```

## Parameters

**Text** [string](#)

The content of the substring.

**Position** [int](#)

The 0-based starting position of the substring in the original text.

# Properties

## Position

The 0-based starting position of the substring in the original text.

```
public int Position { get; init; }
```

Property Value

[int](#)

## Text

The content of the substring.

```
public string Text { get; init; }
```

Property Value

[string](#)

# Class RankSelectBitSet

Namespace: [BelNytheraSeiche.WaveletMatrix](#)

Assembly: BelNytheraSeiche.WaveletMatrix.dll

An immutable bit set optimized for high-performance Rank and Select operations.

```
public sealed class RankSelectBitSet : ImmutableBitSet
```

## Inheritance

[object](#) ← [ImmutableBitSet](#) ← RankSelectBitSet

## Inherited Members

[ImmutableBitSet.Buffer](#), [ImmutableBitSet.Count](#), [ImmutableBitSet.this\[int\]](#), [ImmutableBitSet.At\(int\)](#),  
[ImmutableBitSet.ToRankSelect\(bool\)](#), [ImmutableBitSet.ToRankSelect\(ImmutableBitSet, bool\)](#),  
[object.Equals\(object\)](#), [object.Equals\(object, object\)](#), [object.GetHashCode\(\)](#), [object.GetType\(\)](#),  
[object.ReferenceEquals\(object, object\)](#), [object.ToString\(\)](#)

## Remarks

This data structure uses auxiliary indexes (lookup tables) to perform Rank (counting bits) and Select (finding the n-th bit) operations in near-constant time, making it suitable for succinct data structures like LOUDS Tries.

## Constructors

### RankSelectBitSet(ulong[], int)

An immutable bit set optimized for high-performance Rank and Select operations.

```
public RankSelectBitSet(ulong[] buffer, int count)
```

#### Parameters

**buffer** [ulong](#)[]

The underlying ulong array that stores the bits.

**count** [int](#)

The total number of bits in the set.

## Remarks

This data structure uses auxiliary indexes (lookup tables) to perform Rank (counting bits) and Select (finding the n-th bit) operations in near-constant time, making it suitable for succinct data structures like LOUDS Tries.

# Methods

## CreateAuxDir(ImmutableBitSet)

Creates the auxiliary directories required for fast Rank and Select operations from a given bit set.

```
public static (int[], short[], int[]) CreateAuxDir(ImmutableBitSet bitSet)
```

### Parameters

**bitSet** [ImmutableBitSet](#)

The immutable bit set to process.

### Returns

[\(int\[\], short\[\], int\[\]\)](#)

A tuple containing the generated directories for rank and select.

## GetAuxDir()

Gets the pre-calculated auxiliary directories for Rank and Select operations.

```
public (int[], short[], int[]) GetAuxDir()
```

### Returns

[\(int\[\], short\[\], int\[\]\)](#)

A tuple containing the rank primary directory, rank secondary directory, and the select directory.

## Rank0(int)

Calculates the rank of a bit '0' before a specified position (exclusive). Rank0 is the total number of unset bits (0s) in the range [0, k).

```
public int Rank0(int k)
```

### Parameters

k [int](#)

The one-based rank of the unset bit to find (e.g., k=1 for the first '0').

### Returns

[int](#)

The zero-based index of the k-th unset bit, or -1 if not found.

## Rank1(int)

Calculates the rank of a bit '1' before a specified position (exclusive). Rank1 is the total number of set bits (1s) in the range [0, k).

```
public int Rank1(int k)
```

### Parameters

k [int](#)

The zero-based exclusive end of the range.

### Returns

[int](#)

The number of set bits before position k.

## Select0(int)

Finds the position of the k-th unset bit (0).

```
public int Select0(int k)
```

Parameters

k [int](#)

The one-based rank of the unset bit to find (e.g., k=1 for the first '0').

Returns

[int](#)

The zero-based index of the k-th unset bit, or -1 if not found.

## Select1(int)

Finds the position of the k-th set bit (1).

```
public int Select1(int k)
```

Parameters

k [int](#)

The one-based rank of the set bit to find (e.g., k=1 for the first '1').

Returns

[int](#)

The zero-based index of the k-th set bit, or -1 if not found.

## SetAuxDir((int[], short[], int[]))

Sets the pre-calculated auxiliary directories for Rank and Select operations.

```
public void SetAuxDir((int[], short[], int[]) auxDir)
```

## Parameters

**auxDir** ([int](#)[], [short](#)[], [int](#)[])

A tuple containing the rank primary directory, rank secondary directory, and the select directory.

# Class SerializationOptions

Namespace: [BelNytheraSeiche.WaveletMatrix](#)

Assembly: BelNytheraSeiche.WaveletMatrix.dll

Represents options for serialization.

```
public sealed class SerializationOptions
```

## Inheritance

[object](#) ← SerializationOptions

## Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

## Properties

### CompressionLevel

Gets or sets the compression level for serialization. Defaults to [Fastest](#).

```
public CompressionLevel CompressionLevel { get; set; }
```

### Property Value

[CompressionLevel](#)

### Default

Gets the default serialization options.

```
public static SerializationOptions Default { get; }
```

### Property Value

## SerializationOptions

# Enum SortOrder

Namespace: [BelNytheraSeiche.WaveletMatrix](#)

Assembly: BelNytheraSeiche.WaveletMatrix.dll

Specifies the sort order for the results of a Locate operation.

```
public enum SortOrder
```

## Fields

**Ascending = 1**

The positions are returned in ascending numerical order. This requires an internal sort and is less performant than Unordered.

**Descending = 2**

The positions are returned in descending numerical order. This requires an internal sort and is less performant than Unordered.

**Unordered = 0**

The positions are returned in an unsorted, arbitrary order. This is the most performant option.

# Class SparseTable<T>

Namespace: [BelNytheraSeiche.WaveletMatrix](#)

Assembly: BelNytheraSeiche.WaveletMatrix.dll

Provides a data structure for answering range queries on a static array in O(1) time after an O(N log N) preprocessing step. This version is optimized for **idempotent** operations like finding the minimum or maximum, and it also tracks the index of the result.

```
public sealed class SparseTable<T> where T : IComparable<T>
```

## Type Parameters

T

The type of elements in the array. Must be comparable.

## Inheritance

[object](#) ← SparseTable<T>

## Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

## Remarks

An operation is idempotent if applying it multiple times to the same value does not change the result (e.g.,  $\min(x, x) = x$ ). This implementation is a classic and efficient solution for the static Range Minimum/Maximum Query (RMQ) problem. For an O(N) preprocessing solution, see [FischerHeunSparseTable<T>](#).

## Constructors

### SparseTable(ReadOnlySpan<T>, Func<T, T, bool>)

Initializes a new instance of the [SparseTable<T>](#) class. The constructor performs preprocessing which takes O(N log N) time and O(N log N) space.

```
public SparseTable(ReadOnlySpan<T> array, Func<T, T, bool> comparer)
```

## Parameters

`array ReadOnlySpan<T>`

The static array of data to be queried. The array is not modified.

`comparer Func<T, T, bool>`

A function that compares two elements. It should return `true` if the first argument is considered "better" than the second (e.g., for a minimum query, the comparer would be `(a, b) => a.CompareTo(b) < 0`).

## Exceptions

[ArgumentNullException](#)

Thrown if `comparer` is null.

## Properties

### Size

Gets the total number of elements in the sequence.

```
public int Size { get; }
```

## Property Value

[int](#)

## Methods

### Query()

Queries to find the best value according to the comparer and its original index. This operation takes O(1) time.

```
public SparseTable<T>.ValueWithIndex Query()
```

## Returns

[SparseTable<T>.ValueWithIndex](#)

A [SparseTable<T>.ValueWithIndex](#) record containing the best value and its original index in the input array.

## Remarks

This method is a shorthand for calling `Query(0, Size)`.

## Query(int, int)

Queries the range [start, end) to find the best value according to the comparer and its original index. This operation takes O(1) time.

```
public SparseTable<T>.ValueWithIndex Query(int start, int end)
```

## Parameters

**start** [int](#)

The inclusive, zero-based start of the range.

**end** [int](#)

The exclusive, zero-based end of the range.

## Returns

[SparseTable<T>.ValueWithIndex](#)

A [SparseTable<T>.ValueWithIndex](#) record containing the best value and its original index in the input array.

## Exceptions

[ArgumentOutOfRangeException](#)

Thrown if **start** or **end** are outside the valid bounds.

[ArgumentException](#)

Thrown if `start` is greater than or equal `end`.

# Struct SparseTable<T>.ValueWithIndex

Namespace: [BelNytheraSeiche.WaveletMatrix](#)

Assembly: BelNytheraSeiche.WaveletMatrix.dll

Represents a value and its original index in the input array.

```
public record struct SparseTable<T>.ValueWithIndex :  
IEquatable<SparseTable<T>.ValueWithIndex>
```

Implements

[IEquatable](#)<[SparseTable](#)<T>.ValueWithIndex>

Inherited Members

[ValueType.Equals\(object\)](#) , [ValueType.GetHashCode\(\)](#) , [ValueType.ToString\(\)](#) ,  
[object.Equals\(object, object\)](#) , [object.GetType\(\)](#) , [object.ReferenceEquals\(object, object\)](#)

## Constructors

### ValueWithIndex(T, int)

Represents a value and its original index in the input array.

```
public ValueWithIndex(T Value, int Index)
```

Parameters

**Value** T

The element's value.

**Index** [int](#)

The original zero-based index of the element.

## Properties

# Index

The original zero-based index of the element.

```
public int Index { readonly get; set; }
```

Property Value

[int↗](#)

# Value

The element's value.

```
public T Value { readonly get; set; }
```

Property Value

T

# Class SuffixArray

Namespace: [BelNytheraSeiche.WaveletMatrix](#)

Assembly: BelNytheraSeiche.WaveletMatrix.dll

Provides a powerful string searching data structure using a Suffix Array and LCP Array.

```
public sealed class SuffixArray
```

## Inheritance

[object](#) ← SuffixArray

## Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

## Remarks

This class builds an index from a given string to perform various advanced searches with high performance. Construction is based on the SA-IS algorithm for the suffix array and Kasai's algorithm for the LCP array. While the initial construction can be resource-intensive, subsequent search operations are extremely fast.

## Properties

### Lcp

Gets the LCP (Longest Common Prefix) array. [Lcp\[i\]](#) stores the length of the longest common prefix between the suffixes starting at [SA\[i-1\]](#) and [SA\[i\]](#).

```
public ReadOnlyMemory<int> Lcp { get; }
```

### Property Value

[ReadOnlyMemory](#)<[int](#)>

### Rank

Gets the Rank array (also known as the Inverse Suffix Array). `Rank[i]` gives the rank (the index in the suffix array) of the suffix starting at position `i` of the original text.

```
public ReadOnlyMemory<int> Rank { get; }
```

## Property Value

[ReadOnlyMemory](#)<int>

## SA

Gets the suffix array. This is a sorted array of all suffixes of the text.

```
public ReadOnlyMemory<int> SA { get; }
```

## Property Value

[ReadOnlyMemory](#)<int>

## Remarks

The 0-th element of the internal array is skipped as it corresponds to the sentinel character used by the SA-IS algorithm.

## Text

Gets the original text used to build the suffix array.

```
public ReadOnlyMemory<char> Text { get; }
```

## Property Value

[ReadOnlyMemory](#)<char>

## Methods

## Create(ReadOnlyMemory<char>)

Creates a new instance of the [SuffixArray](#) class by building the suffix and LCP arrays for the specified text.

```
public static SuffixArray Create(ReadOnlyMemory<char> text)
```

### Parameters

**text** [ReadOnlyMemory](#)<[char](#)>

The input string to be indexed. The string will be held in memory.

### Returns

[SuffixArray](#)

### Exceptions

[ArgumentNullException](#)

Thrown if **text** is null.

[ArgumentOutOfRangeException](#)

Thrown if **text** length exceeds the maximum supported length of 100,000,000 characters.

## Create(string)

Creates a new instance of the [SuffixArray](#) class by building the suffix and LCP arrays for the specified text.

```
public static SuffixArray Create(string text)
```

### Parameters

**text** [string](#)

The input string to be indexed. The string will be held in memory.

### Returns

[SuffixArray](#)

## Exceptions

### [ArgumentNullException](#)

Thrown if `text` is null.

### [ArgumentOutOfRangeException](#)

Thrown if `text` length exceeds the maximum supported length of 100,000,000 characters.

## Deserialize(byte[])

Deserializes a byte array into a [SuffixArray](#) instance.

```
public static SuffixArray Deserialize(byte[] data)
```

## Parameters

### `data` [byte](#)[]

The byte array containing the serialized data.

## Returns

### [SuffixArray](#)

A new, deserialized instance of [SuffixArray](#).

## Exceptions

### [ArgumentNullException](#)

Thrown if `data` is null.

## Deserialize(Stream)

Deserializes a [SuffixArray](#) instance from a stream. It verifies the file format, type identifier, and checksum.

```
public static SuffixArray Deserialize(Stream stream)
```

## Parameters

### `stream Stream` ↗

The stream to read the serialized data from.

## Returns

### [SuffixArray](#)

A new, deserialized instance.

## Exceptions

### [ArgumentNullException](#) ↗

Thrown if `stream` is null.

### [InvalidDataException](#) ↗

Thrown if the data format is unsupported, the type is incompatible, or the data is corrupt.

## Deserialize(string)

Deserializes a [SuffixArray](#) instance from the specified file.

```
public static SuffixArray Deserialize(string file)
```

## Parameters

### `file string` ↗

The path of the file to read from.

## Returns

### [SuffixArray](#)

A new, deserialized instance.

## Exceptions

## [ArgumentNullException](#)

Thrown if `file` is null.

## Search(ReadOnlySpan<char>)

Finds all starting positions of a specified substring. The search is case-sensitive.

```
public IEnumerable<int> Search(ReadOnlySpan<char> text)
```

### Parameters

`text` [ReadOnlySpan<char>](#)

The substring to search for.

### Returns

[IEnumerable<int>](#)

An enumerable collection of 0-based starting positions where the substring is found, sorted in ascending order.

## SearchCommon(ReadOnlyMemory<char>, int)

```
public IEnumerable<SuffixArray.MatchGroup> SearchCommon(ReadOnlyMemory<char> text, int minLength = 2)
```

### Parameters

`text` [ReadOnlyMemory<char>](#)

`minLength` [int](#)

### Returns

[IEnumerable<SuffixArray.MatchGroup>](#)

## SearchCommon(ReadOnlyMemory<char>, ReadOnlyMemory<char>, int)

```
public static IEnumerable<SuffixArray.MatchGroup> SearchCommon(ReadOnlyMemory<char> text1,  
    ReadOnlyMemory<char> text2, int minLength = 2)
```

### Parameters

`text1` [ReadOnlyMemory<char>](#)

`text2` [ReadOnlyMemory<char>](#)

`minLength` [int](#)

### Returns

[IEnumerable<SuffixArray.MatchGroup>](#)

## SearchCommon(string, int)

Finds all common substrings between the text of this instance and another specified text that meet a minimum length requirement.

```
public IEnumerable<SuffixArray.MatchGroup> SearchCommon(string text, int minLength = 2)
```

### Parameters

`text` [string](#)

The other text to compare against.

`minLength` [int](#)

The minimum length of the common substrings to find. Must be 2 or greater.

### Returns

[IEnumerable<SuffixArray.MatchGroup>](#)

An enumerable collection of [SuffixArray.MatchGroup](#) records for each unique common substring found.

## Exceptions

### [ArgumentNullException](#)

Thrown if `text` is null.

### [ArgumentOutOfRangeException](#)

Thrown if `minLength` is less than 2.

## SearchCommon(string, string, int)

Finds all common substrings between two specified texts that meet a minimum length requirement.

```
public static IEnumerable<SuffixArray.MatchGroup> SearchCommon(string text1, string text2, int minLength = 2)
```

## Parameters

### `text1` [string](#)

The first text.

### `text2` [string](#)

The second text.

### `minLength` [int](#)

The minimum length of the common substrings to find. Must be 2 or greater.

## Returns

### [IEnumerable](#) <[SuffixArray.MatchGroup](#)>

An enumerable collection of [SuffixArray.MatchGroup](#) records for each unique common substring found.

## Exceptions

## [ArgumentNullException](#)

Thrown if `text1` or `text2` is null.

## [ArgumentOutOfRangeException](#)

Thrown if `minLength` is less than 2.

# SearchLongestCommon(ReadOnlyMemory<char>)

```
public IEnumerable<SuffixArray.MatchGroup> SearchLongestCommon(ReadOnlyMemory<char> text)
```

## Parameters

`text` [ReadOnlyMemory](#)<`char`>

## Returns

[IEnumerable](#)<[SuffixArray.MatchGroup](#)>

# SearchLongestCommon(ReadOnlyMemory<char>, ReadOnlyMemory<char>)

```
public static IEnumerable<SuffixArray.MatchGroup> SearchLongestCommon(ReadOnlyMemory<char>  
text1, ReadOnlyMemory<char> text2)
```

## Parameters

`text1` [ReadOnlyMemory](#)<`char`>

`text2` [ReadOnlyMemory](#)<`char`>

## Returns

[IEnumerable](#)<[SuffixArray.MatchGroup](#)>

# SearchLongestCommon(string)

Finds the longest common substring(s) between the text of this instance and another specified text.

```
public IEnumerable<SuffixArray.MatchGroup> SearchLongestCommon(string text)
```

## Parameters

**text** [string](#)

The other text to compare against.

## Returns

[IEnumerable](#) <[SuffixArray.MatchGroup](#)>

An enumerable collection of [SuffixArray.MatchGroup](#) records for each longest common substring found.

## Exceptions

[ArgumentNullException](#)

Thrown if **text** is null.

## SearchLongestCommon(string, string)

Finds the longest common substring(s) between two specified texts.

```
public static IEnumerable<SuffixArray.MatchGroup> SearchLongestCommon(string text1,  
    string text2)
```

## Parameters

**text1** [string](#)

The first text.

**text2** [string](#)

The second text.

Returns

#### [IEnumerable<SuffixArray.MatchGroup>](#)

An enumerable collection of [SuffixArray.MatchGroup](#) records for each longest common substring found.

Exceptions

#### [ArgumentNullException](#)

Thrown if `text1` or `text2` is null.

## SearchLongestRepeated()

Finds the longest substring(s) that appear more than once in the text.

```
public IEnumerable<SuffixArray.MatchRepeated> SearchLongestRepeated()
```

Returns

#### [IEnumerable<SuffixArray.MatchRepeated>](#)

An enumerable collection of [SuffixArray.MatchRepeated](#) records, one for each longest repeated substring found.

## SearchRepeated(int)

Finds all unique substrings that are repeated in the text and meet a minimum length requirement.

```
public IEnumerable<SuffixArray.MatchRepeated> SearchRepeated(int minLength = 2)
```

Parameters

#### `minLength` [int](#)

The minimum length of the repeated substrings to find. Must be 2 or greater.

Returns

## [IEnumerable](#) <[SuffixArray.MatchRepeated](#)>

An enumerable collection of [SuffixArray.MatchRepeated](#) records for each unique repeated substring found.

## Exceptions

### [ArgumentOutOfRangeException](#)

Thrown if `minLength` is less than 2.

## SearchWildcard(ReadOnlySpan<char>, WildcardOptions?)

Finds all substrings that match a given pattern containing wildcards.

```
public IEnumerable<SuffixArray.Match> SearchWildcard(ReadOnlySpan<char> expr,  
SuffixArray.WildcardOptions? options = null)
```

## Parameters

### `expr` [ReadOnlySpan](#) <[char](#)>

The search pattern. Use '\*' for a variable-length wildcard and '?' for a single-character wildcard.

### `options` [SuffixArray.WildcardOptions](#)

The options that customize the wildcard search behavior. If null, the default options specified by [Default](#) will be used.

## Returns

## [IEnumerable](#) <[SuffixArray.Match](#)>

An enumerable collection of [SuffixArray.Match](#) records for all found occurrences.

## Remarks

The search is anchored by the non-wildcard parts of the pattern. Therefore, leading and trailing asterisks are trimmed as they do not provide additional constraints for the search and have no effect on the result.

## Exceptions

## [ArgumentNullException](#)

Thrown if `expr` is null.

## [ArgumentException](#)

Thrown if and are the same character.

## [ArgumentException](#)

Thrown if or are found in .

## [ArgumentException](#)

Thrown if is greater than .

# Serialize(SuffixArray, SerializationOptions?)

Serializes the [SuffixArray](#) instance into a byte array.

```
public static byte[] Serialize(SuffixArray obj, SerializationOptions? options = null)
```

## Parameters

### `obj` [SuffixArray](#)

The instance to serialize.

### `options` [SerializationOptions](#)

Serialization options, such as the compression level. If null, [Default](#) will be used.

## Returns

### `byte`[]

A byte array containing the serialized data.

## Exceptions

### [ArgumentNullException](#)

Thrown if `obj` is null.

## Serialize(SuffixArray, Stream, SerializationOptions?)

Serializes the [SuffixArray](#) instance to a stream. The data is compressed using Brotli and includes a checksum for integrity verification.

```
public static void Serialize(SuffixArray obj, Stream stream, SerializationOptions? options  
= null)
```

### Parameters

**obj** [SuffixArray](#)

The instance to serialize.

**stream** [Stream](#)

The stream to write the serialized data to.

**options** [SerializationOptions](#)

Serialization options, such as the compression level. If null, [Default](#) will be used.

### Exceptions

[ArgumentNullException](#)

Thrown if **obj** or **stream** is null.

## Serialize(SuffixArray, string, SerializationOptions?)

Serializes the [SuffixArray](#) instance to the specified file.

```
public static void Serialize(SuffixArray obj, string file, SerializationOptions? options  
= null)
```

### Parameters

**obj** [SuffixArray](#)

The instance to serialize.

`file` [string](#)

The path of the file to write to.

`options` [SerializationOptions](#)

Serialization options, such as the compression level. If null, [Default](#) will be used.

## Exceptions

[ArgumentNullException](#)

Thrown if `obj` or `file` is null.

# Struct SuffixArray.Match

Namespace: [BelNytheraSeiche.WaveletMatrix](#)

Assembly: BelNytheraSeiche.WaveletMatrix.dll

Provides a single match found in a text.

```
public record struct SuffixArray.Match : IEquatable<SuffixArray.Match>
```

Implements

[IEquatable](#) <[SuffixArray.Match](#)>

Inherited Members

[ValueType.Equals\(object\)](#) , [ValueType.GetHashCode\(\)](#) , [ValueType.ToString\(\)](#) ,  
[object.Equals\(object, object\)](#) , [object.GetType\(\)](#) , [object.ReferenceEquals\(object, object\)](#)

## Constructors

### Match(int, int)

Provides a single match found in a text.

```
public Match(int Position, int Length)
```

Parameters

**Position** [int](#)

The 0-based starting position of the match.

**Length** [int](#)

The length of the match.

## Properties

### Length

The length of the match.

```
public int Length { readonly get; set; }
```

Property Value

[int↗](#)

## Position

The 0-based starting position of the match.

```
public int Position { readonly get; set; }
```

Property Value

[int↗](#)

# Struct SuffixArray.MatchGroup

Namespace: [BelNytheraSeiche.WaveletMatrix](#)

Assembly: BelNytheraSeiche.WaveletMatrix.dll

Provides a common substring found between two texts.

```
public record struct SuffixArray.MatchGroup : IEquatable<SuffixArray.MatchGroup>
```

## Implements

[IEquatable](#) <[SuffixArray.MatchGroup](#)>

## Inherited Members

[ValueType.Equals\(object\)](#) , [ValueType.GetHashCode\(\)](#) , [ValueType.ToString\(\)](#) ,  
[object.Equals\(object, object\)](#) , [object.GetType\(\)](#) , [object.ReferenceEquals\(object, object\)](#)

## Constructors

### MatchGroup(string, int[], int[])

Provides a common substring found between two texts.

```
public MatchGroup(string Text, int[] Positions1, int[] Positions2)
```

#### Parameters

**Text** [string](#)

The common substring.

**Positions1** [int](#)[]

An array of all 0-based starting positions in the first text.

**Positions2** [int](#)[]

An array of all 0-based starting positions in the second text.

# Properties

## Positions1

An array of all 0-based starting positions in the first text.

```
public int[] Positions1 { readonly get; set; }
```

Property Value

[int](#)[]

## Positions2

An array of all 0-based starting positions in the second text.

```
public int[] Positions2 { readonly get; set; }
```

Property Value

[int](#)[]

## Text

The common substring.

```
public string Text { readonly get; set; }
```

Property Value

[string](#)

# Struct SuffixArray.MatchRepeated

Namespace: [BelNytheraSeiche.WaveletMatrix](#)

Assembly: BelNytheraSeiche.WaveletMatrix.dll

Provides a repeated substring and all its occurrences.

```
public record struct SuffixArray.MatchRepeated : IEquatable<SuffixArray.MatchRepeated>
```

Implements

[IEquatable](#) <[SuffixArray.MatchRepeated](#)>

Inherited Members

[ValueType.Equals\(object\)](#) , [ValueType.GetHashCode\(\)](#) , [ValueType.ToString\(\)](#) ,  
[object.Equals\(object, object\)](#) , [object.GetType\(\)](#) , [object.ReferenceEquals\(object, object\)](#)

## Constructors

### MatchRepeated(string, int[])

Provides a repeated substring and all its occurrences.

```
public MatchRepeated(string Text, int[] Positions)
```

Parameters

**Text** [string](#)

The repeated substring.

**Positions** [int](#)[]

An array of all 0-based starting positions where the substring occurs.

## Properties

### Positions

An array of all 0-based starting positions where the substring occurs.

```
public int[] Positions { readonly get; set; }
```

Property Value

[int](#)[]

Text

The repeated substring.

```
public string Text { readonly get; set; }
```

Property Value

[string](#)

# Class SuffixArray.WildcardOptions

Namespace: [BelNytheraSeiche.WaveletMatrix](#)

Assembly: BelNytheraSeiche.WaveletMatrix.dll

Provides options to customize the behavior of a wildcard search.

```
public sealed class SuffixArray.WildcardOptions
```

## Inheritance

[object](#) ← SuffixArray.WildcardOptions

## Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

## Properties

### Asterisk

The character to be treated as a variable-length wildcard (\*).

```
public char Asterisk { get; set; }
```

#### Property Value

[char](#)

### AsteriskMaxLength

Gets or sets the maximum number of characters the wildcard can match. Defaults to Int32.MaxValue.

```
public int AsteriskMaxLength { get; set; }
```

#### Property Value

[int](#)

## AsteriskMinLength

Gets or sets the minimum number of characters the wildcard can match. Defaults to 0.

```
public int AsteriskMinLength { get; set; }
```

### Property Value

[int](#)

## Default

Gets a singleton instance of [SuffixArray.WildcardOptions](#) with default values.

```
public static SuffixArray.WildcardOptions Default { get; }
```

### Property Value

[SuffixArray.WildcardOptions](#)

## Question

The character to be treated as a single-character wildcard (?).

```
public char Question { get; set; }
```

### Property Value

[char](#)

## StopCharacters

Gets or sets a set of characters that will stop the match. If the match encounters any of these characters, it will end.

```
public char[]? StopCharacters { get; set; }
```

Property Value

[char\[\]](#)

# Class ValueBuffer<T>

Namespace: [BelNytheraSeiche.WaveletMatrix](#)

Assembly: BelNytheraSeiche.WaveletMatrix.dll

Represents a buffer for value types, implemented using a list of array chunks.

```
public sealed class ValueBuffer<T> : ICollection<T>, IEnumerable<T>, IEnumerable where T : struct
```

## Type Parameters

T

The type of element, which must be a struct.

## Inheritance

[object](#) ← ValueBuffer<T>

## Implements

[ICollection](#)<T>, [IEnumerable](#)<T>, [IEnumerable](#)

## Inherited Members

[object.Equals\(object\)](#), [object.Equals\(object, object\)](#), [object.GetHashCode\(\)](#), [object.GetType\(\)](#),  
[object.ReferenceEquals\(object, object\)](#), [object.ToString\(\)](#)

## Remarks

This class is designed to manage large collections of value types without allocating on the Large Object Heap (LOH). It achieves this by storing elements in a series of smaller array "chunks". The size of these chunks, specified by [ElementsPerChunk](#), must be a power of 2.

## Constructors

### ValueBuffer(int, bool)

Initializes a new instance of the [ValueBuffer<T>](#) class.

```
public ValueBuffer(int elementsPerChunk, bool extendAutomatically = false)
```

## Parameters

`elementsPerChunk` [int](#)

The size of each internal array chunk. This value must be a power of 2.

`extendAutomatically` [bool](#)

A value indicating whether to automatically extend the buffer's capacity when an out-of-bounds index is accessed.

## Exceptions

[ArgumentOutOfRangeException](#)

`elementsPerChunk` is not positive.

[ArgumentException](#)

`elementsPerChunk` is not a power of 2.

## Properties

### Capacity

Gets the total number of elements that can be stored across all currently allocated chunks.

```
public int Capacity { get; }
```

### Property Value

[int](#)

### Count

Gets the number of elements contained in the [ValueBuffer<T>](#). This property is an implementation for the [ICollection<T>](#) interface and is an alias for the [Used](#) property.

```
public int Count { get; }
```

Property Value

[int ↗](#)

## ElementsPerChunk

Gets the size of each internal array chunk.

```
public int ElementsPerChunk { get; }
```

Property Value

[int ↗](#)

## ExtendAutomatically

Gets a value indicating whether the buffer will automatically allocate new chunks when an index outside the current capacity is accessed.

```
public bool ExtendAutomatically { get; }
```

Property Value

[bool ↗](#)

## IsReadOnly

Gets a value indicating whether the [ValueBuffer<T>](#) is read-only.

```
public bool IsReadOnly { get; }
```

Property Value

[bool ↗](#)

## this[int]

Gets or sets the element at the specified index.

```
public ref T this[int index] { get; }
```

### Parameters

**index** [int](#)

The zero-based index of the element to get or set.

### Property Value

T

A reference to the element at the specified index.

### Remarks

Accessing an index greater than or equal to the current [Used](#) count will update the count to [index + 1](#). If [ExtendAutomatically](#) is true, accessing an index beyond the current [Capacity](#) will allocate new chunks.

### Exceptions

[ArgumentOutOfRangeException](#)

Thrown if [index](#) is negative, or if [index](#) is greater than or equal to [Capacity](#) and [ExtendAutomatically](#) is false.

## NumberOfChunks

Gets the number of internal array chunks that are currently allocated.

```
public int NumberOfChunks { get; }
```

### Property Value

[int](#)

## Used

Gets the number of elements that are considered to be in use within the buffer. This value is updated when an element is written to an index greater than the current used count.

```
public int Used { get; }
```

## Property Value

[int ↗](#)

## Methods

### Add(T)

Adds an element to the end of the [ValueBuffer<T>](#).

```
public void Add(T value)
```

#### Parameters

**value T**

The element to add.

### AppendChunkDirectly(T[])

Appends a pre-allocated array chunk directly to the internal list of buffers, increasing the capacity.

```
public void AppendChunkDirectly(T[] chunk)
```

#### Parameters

**chunk T[]**

The array to append. Its length must be exactly equal to [ElementsPerChunk](#).

## Remarks

This is an advanced optimization method that avoids an array allocation and copy by using a caller-provided array.

## Exceptions

### [ArgumentNullException](#)

`chunk` is null.

### [ArgumentException](#)

The length of `chunk` is not equal to [ElementsPerChunk](#).

## Clear()

Removes all elements from the [ValueBuffer<T>](#) and releases all internal array chunks.

```
public void Clear()
```

## Contains(T)

Determines whether the [ValueBuffer<T>](#) contains a specific value.

```
public bool Contains(T item)
```

## Parameters

`item` T

The object to locate in the [ValueBuffer<T>](#).

## Returns

[bool](#)

true if item is found in the [ValueBuffer<T>](#); otherwise, false.

## CopyTo(T[], int)

Copies the elements of the [ValueBuffer<T>](#) to an [Array](#), starting at a particular [Array](#) index.

```
public void CopyTo(T[] array, int arrayIndex)
```

## Parameters

**array** `T[]`

The one-dimensional [Array](#) that is the destination of the elements copied from [ValueBuffer<T>](#).

**arrayIndex** `int`

The zero-based index in **array** at which copying begins.

## Exceptions

[ArgumentNullException](#)

**array** is null.

[ArgumentOutOfRangeException](#)

**arrayIndex** is less than 0.

[ArgumentException](#)

The number of elements in the source buffer is greater than the available space from **arrayIndex** to the end of the destination **array**.

## EnumerateChunks()

Returns an enumerable collection of the internal array chunks.

```
public IEnumerable<T[]> EnumerateChunks()
```

## Returns

[IEnumerable](#)<`T[]`>

An [IEnumerable](#)<`T`> (`T` is an array) that allows iteration over the raw internal buffers.

## Remarks

This method is intended for advanced scenarios where direct read-only access to the underlying chunks is required.

## ExtendCapacity(int)

Ensures that the capacity of this buffer is at least the specified value.

```
public int ExtendCapacity(int minCapacity)
```

### Parameters

**minCapacity** [int](#)

The minimum required capacity.

### Returns

[int](#)

The new capacity of the buffer, which will be a multiple of [ElementsPerChunk](#).

### Exceptions

[ArgumentOutOfRangeException](#)

**minCapacity** is negative.

## Fill(T, int, int)

Fills a range of elements in the buffer with a specified value.

```
public void Fill(T value, int index, int count)
```

### Parameters

**value** T

The value to assign to each element in the range.

`index` [int](#)

The zero-based starting index of the range to fill.

`count` [int](#)

The number of elements in the range to fill.

## Exceptions

[ArgumentOutOfRangeException](#)

`index` or `count` is negative.

[ArgumentException](#)

The sum of `index` and `count` is greater than the buffer's capacity and [ExtendAutomatically](#) is false.

## GetEnumerator()

Returns an enumerator that iterates through the used elements in the buffer.

```
public IEnumrator<T> GetEnumerator()
```

## Returns

[IEnumrator](#)<T>

An enumerator for the [ValueBuffer<T>](#).

## Remove(T)

Finds the first occurrence of an item and replaces it with the default value of type `T`.

```
public bool Remove(T item)
```

## Parameters

`item` `T`

The item to locate and replace.

Returns

[bool](#)

true if the item was found and replaced; otherwise, false.

Remarks

This method does not actually remove the element in a way that would shift subsequent elements. It only overwrites the found element with `default(T)`. The [Count](#) and [Used](#) properties are not changed by this operation.

## SetUsedDirectly(int, bool)

Directly sets the number of used elements and optionally shrinks the buffer's capacity.

```
public void SetUsedDirectly(int used, bool shrinkAutomatically = false)
```

Parameters

`used` [int](#)

The new count of used elements. This value must be between 0 and [Capacity](#).

`shrinkAutomatically` [bool](#)

If true, deallocates unused array chunks from the end of the buffer to fit the new used count.

Remarks

This is an advanced method and should be used with caution. Setting an incorrect value can lead to an inconsistent state or data corruption.

Exceptions

[ArgumentOutOfRangeException](#)

`used` is negative or greater than [Capacity](#).

## ToArray()

Copies the used elements from the [ValueBuffer<T>](#) to a new, single contiguous array.

```
public T[] ToArray()
```

Returns

T[]

A new array containing the elements from the buffer.

# Class WaveletMatrixCore

Namespace: [BelNytheraSeiche.WaveletMatrix](#)

Assembly: BelNytheraSeiche.WaveletMatrix.dll

Provides the core, high-performance implementation of a Wavelet Matrix for integer sequences. This class is not generic and operates directly on coordinate-compressed integer arrays.

```
public sealed class WaveletMatrixCore
```

## Inheritance

[object](#) ← WaveletMatrixCore

## Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

## Remarks

This class forms the engine for the generic [WaveletMatrixGeneric<T>](#). It encapsulates the bit-level data structures and algorithms, such as Rank, Select, and Quantile.

## Properties

### Size

Gets the total number of elements in the sequence.

```
public int Size { get; }
```

## Property Value

[int](#)

## Methods

### Access(int)

Gets the value of the element at the specified zero-based position.

```
public int Access(int k)
```

Parameters

**k** [int](#)

The zero-based index of the element to retrieve.

Returns

[int](#)

The integer value at the specified position.

Exceptions

[ArgumentOutOfRangeException](#)

Thrown if **k** is out of range.

## Create(IEnumerable<int>)

Creates a new instance of the [WaveletMatrixCore](#) from a sequence of non-negative integers.

```
public static WaveletMatrixCore Create(IEnumerable<int> sequence)
```

Parameters

**sequence** [IEnumerable](#)<[int](#)>

The sequence of integers to build the Wavelet Matrix from. All values must be non-negative.

Returns

[WaveletMatrixCore](#)

A new, fully initialized [WaveletMatrixCore](#) instance.

## Exceptions

### [ArgumentNullException](#)

Thrown if `sequence` is null.

### [ArgumentException](#)

Thrown if `sequence` contains negative values.

## Deserialize(byte[])

Deserializes a byte array into a [WaveletMatrixCore](#) instance.

```
public static WaveletMatrixCore Deserialize(byte[] data)
```

### Parameters

#### `data` [byte](#)[]

The byte array containing the serialized data.

### Returns

#### [WaveletMatrixCore](#)

A new, deserialized instance of [WaveletMatrixCore](#).

## Exceptions

### [ArgumentNullException](#)

Thrown if `data` is null.

## Deserialize(Stream)

Deserializes a [WaveletMatrixCore](#) instance from a stream. It verifies the file format, type identifier, and checksum.

```
public static WaveletMatrixCore Deserialize(Stream stream)
```

## Parameters

### `stream Stream` ↗

The stream to read the serialized data from.

## Returns

### [WaveletMatrixCore](#)

A new, deserialized instance.

## Exceptions

### [ArgumentNullException](#) ↗

Thrown if `stream` is null.

### [InvalidOperationException](#) ↗

Thrown if the data format is unsupported, the type is incompatible, or the data is corrupt.

## Deserialize(string)

Deserializes a [WaveletMatrixCore](#) instance from the specified file.

```
public static WaveletMatrixCore Deserialize(string file)
```

## Parameters

### `file string` ↗

The path of the file to read from.

## Returns

### [WaveletMatrixCore](#)

A new, deserialized instance.

## Exceptions

## [ArgumentNullException](#)

Thrown if `file` is null.

## LargerValue(int)

Finds the successor of a given value. The successor is the smallest value that is strictly larger than the given `value`.

```
public int? LargerValue(int value)
```

### Parameters

`value` [int](#)

The reference value.

### Returns

[int](#)?

The successor value, or null if no such value is found.

### Remarks

This method is a shorthand for calling `LargerValue(0, Size, value)`.

## LargerValue(int, int, int)

Finds the successor of a given value in the specified range [start, end). The successor is the smallest value in the range that is strictly larger than the given `value`.

```
public int? LargerValue(int start, int end, int value)
```

### Parameters

`start` [int](#)

The inclusive start of the range.

`end int`

The exclusive end of the range.

`value int`

The reference value.

Returns

`int?`

The successor value, or null if no such value is found.

Exceptions

[ArgumentOutOfRangeException](#)

Thrown if `start` or `end` is out of range.

[ArgumentException](#)

Thrown if `start` is greater than or equal `end`.

## Quantile(int)

Finds the k-th smallest value.

```
public int Quantile(int k)
```

Parameters

`k int`

The zero-based rank of the value to find (e.g., `k=0` for the smallest value).

Returns

`int`

The k-th smallest value.

## Remarks

This method is a shorthand for calling Quantile(0, [Size](#), [k](#)).

## Exceptions

### [ArgumentException](#)

Thrown if [k](#) is out of the valid range.

### [ArgumentOutOfRangeException](#)

Thrown if [k](#) is out of the valid range [0, [Size](#)).

## Quantile(int, int, int)

Finds the k-th smallest value within the specified range [start, end).

```
public int Quantile(int start, int end, int k)
```

## Parameters

### [start](#) [int](#)

The inclusive start of the range.

### [end](#) [int](#)

The exclusive end of the range.

### [k](#) [int](#)

The zero-based rank of the value to find (e.g., k=0 for the smallest value).

## Returns

### [int](#)

The k-th smallest value in the range.

## Exceptions

### [ArgumentOutOfRangeException](#)

Thrown if `start`, `end` or `k` is out of range.

## [ArgumentException](#)

Thrown if `start` is greater than or equal `end`.

## [ArgumentException](#)

Thrown if `k` is out of the valid range [0, `end`-`start`).

# RangeCount(int)

Counts the number of occurrences of a specified value.

```
public int RangeCount(int value)
```

## Parameters

### `value` [int](#)

The value to count.

## Returns

### [int](#)

The number of occurrences of the value.

## Remarks

This method is a shorthand for calling `RangeCount(0, Size, value)`.

# RangeCount(int, int, int)

Counts the number of occurrences of a specified value within the range [start, end).

```
public int RangeCount(int start, int end, int value)
```

## Parameters

**start** [int ↗](#)

The inclusive start of the range.

**end** [int ↗](#)

The exclusive end of the range.

**value** [int ↗](#)

The value to count.

Returns

[int ↗](#)

The number of occurrences of the value in the range.

Exceptions

[ArgumentOutOfRangeException ↗](#)

Thrown if **start** or **end** is out of range.

[ArgumentException ↗](#)

Thrown if **start** is greater than or equal **end**.

## RangeFreq(int, int)

Counts the number of elements whose values are within the value range [minValue, maxValue).

```
public int RangeFreq(int minValue, int maxValue)
```

Parameters

**minValue** [int ↗](#)

The inclusive start of the value range.

**maxValue** [int ↗](#)

The exclusive end of the value range.

## Returns

[int ↗](#)

The count of elements matching the criteria.

## Remarks

This method is a shorthand for calling RangeFreq(0, [Size](#), [minValue](#), [maxValue](#)).

## Exceptions

[ArgumentException ↗](#)

Thrown if [minValue](#) is greater than or equal [maxValue](#).

## RangeFreq(int, int, int, int)

Counts the number of elements in a position range [start, end) whose values are within the value range [[minValue](#), [maxValue](#)).

```
public int RangeFreq(int start, int end, int minValue, int maxValue)
```

## Parameters

[start](#) [int ↗](#)

The inclusive start of the position range.

[end](#) [int ↗](#)

The exclusive end of the position range.

[minValue](#) [int ↗](#)

The inclusive start of the value range.

[maxValue](#) [int ↗](#)

The exclusive end of the value range.

## Returns

## [int](#)

The count of elements matching the criteria.

## Exceptions

### [ArgumentOutOfRangeException](#)

Thrown if `start` or `end` is out of range.

### [ArgumentException](#)

Thrown if `start` is greater than or equal `end`.

### [ArgumentException](#)

Thrown if `minValue` is greater than or equal `maxValue`.

## RangeMode()

Finds the most frequent value (the mode) and its frequency.

```
public WaveletMatrixCore.ValueWithFrequency RangeMode()
```

## Returns

### [WaveletMatrixCore.ValueWithFrequency](#)

A [WaveletMatrixCore.ValueWithFrequency](#) record containing the mode and its count.

## Remarks

This method is a shorthand for calling `RangeMode(0, Size)`.

## RangeMode(int, int)

Finds the most frequent value (the mode) and its frequency within the specified range [start, end].

```
public WaveletMatrixCore.ValueWithFrequency RangeMode(int start, int end)
```

## Parameters

**start** [int](#)

The inclusive start of the range.

**end** [int](#)

The exclusive end of the range.

## Returns

[WaveletMatrixCore.ValueWithFrequency](#)

A [WaveletMatrixCore.ValueWithFrequency](#) record containing the mode and its count.

## Exceptions

[ArgumentOutOfRangeException](#)

Thrown if **start** or **end** is out of range.

[ArgumentException](#)

Thrown if **start** is greater than or equal **end**.

## Rank(int, int)

Counts the number of occurrences of a specified value in the prefix of the sequence [0, k).

```
public int Rank(int k, int value)
```

## Parameters

**k** [int](#)

The exclusive end of the range [0, k).

**value** [int](#)

The value to count.

Returns

[int ↗](#)

The number of times the value appears in the specified prefix.

Exceptions

[ArgumentOutOfRangeException ↗](#)

Thrown if **k** is out of range.

## Select(int, int)

Finds the zero-based position of the k-th occurrence of a specified value.

`public int Select(int k, int value)`

Parameters

**k** [int ↗](#)

The one-based rank of the occurrence to find (e.g., k=1 for the first occurrence).

**value** [int ↗](#)

The value to search for.

Returns

[int ↗](#)

The zero-based index of the k-th occurrence of the value, or -1 if not found.

Exceptions

[ArgumentOutOfRangeException ↗](#)

Thrown if **k** is out of range.

## Serialize(WaveletMatrixCore, SerializationOptions?)

Serializes the [WaveletMatrixCore](#) instance into a byte array.

```
public static byte[] Serialize(WaveletMatrixCore obj, SerializationOptions? options = null)
```

### Parameters

**obj** [WaveletMatrixCore](#)

The instance to serialize.

**options** [SerializationOptions](#)

Serialization options, such as the compression level. If null, [Default](#) will be used.

### Returns

[byte](#)[]

A byte array containing the serialized data.

### Exceptions

[ArgumentNullException](#)

Thrown if **obj** is null.

## Serialize(WaveletMatrixCore, Stream, SerializationOptions?)

Serializes the [WaveletMatrixCore](#) instance to a stream. The data is compressed using Brotli and includes a checksum for integrity verification.

```
public static void Serialize(WaveletMatrixCore obj, Stream stream, SerializationOptions? options = null)
```

### Parameters

**obj** [WaveletMatrixCore](#)

The instance to serialize.

## `stream` [Stream](#)

The stream to write the serialized data to.

## `options` [SerializationOptions](#)

Serialization options, such as the compression level. If null, [Default](#) will be used.

## Exceptions

### [ArgumentNullException](#)

Thrown if `obj` or `stream` is null.

## Serialize(WaveletMatrixCore, string, SerializationOptions?)

Serializes the [WaveletMatrixCore](#) instance to the specified file.

```
public static void Serialize(WaveletMatrixCore obj, string file, SerializationOptions?  
options = null)
```

## Parameters

### `obj` [WaveletMatrixCore](#)

The instance to serialize.

### `file` [string](#)

The path of the file to write to.

### `options` [SerializationOptions](#)

Serialization options, such as the compression level. If null, [Default](#) will be used.

## Exceptions

### [ArgumentNullException](#)

Thrown if `obj` or `file` is null.

## SmallerValue(int)

Finds the predecessor of a given value. The predecessor is the largest value that is strictly smaller than the given **value**.

```
public int? SmallerValue(int value)
```

### Parameters

**value** [int](#)

The reference value.

### Returns

[int](#)?

The predecessor value, or null if no such value is found.

### Remarks

This method is a shorthand for calling SmallerValue(0, [Size](#), **value**).

## SmallerValue(int, int, int)

Finds the predecessor of a given value in the specified range [start, end). The predecessor is the largest value in the range that is strictly smaller than the given **value**.

```
public int? SmallerValue(int start, int end, int value)
```

### Parameters

**start** [int](#)

The inclusive start of the range.

**end** [int](#)

The exclusive end of the range.

**value** [int](#)

The reference value.

Returns

[int](#)?

The predecessor value, or null if no such value is found.

Exceptions

[ArgumentOutOfRangeException](#)

Thrown if `start` or `end` is out of range.

[ArgumentException](#)

Thrown if `start` is greater than or equal `end`.

## TopK(int)

Finds the top K most frequent values and their frequencies.

```
public IEnumerable<WaveletMatrixCore.ValueWithFrequency> TopK(int k)
```

Parameters

`k` [int](#)

The number of top elements to retrieve.

Returns

[IEnumerable](#) <[WaveletMatrixCore.ValueWithFrequency](#)>

An enumerable collection of [WaveletMatrixCore.ValueWithFrequency](#) records, ordered by frequency in descending order.

Remarks

This method is a shorthand for calling `TopK(0, Size, k)`.

Exceptions

## [ArgumentOutOfRangeException](#)

Thrown if `k` is out of range.

## TopK(int, int, int)

Finds the top K most frequent values and their frequencies within the specified range [start, end).

```
public IEnumerable<WaveletMatrixCore.ValueWithFrequency> TopK(int start, int end, int k)
```

### Parameters

`start` [int](#)

The inclusive start of the range.

`end` [int](#)

The exclusive end of the range.

`k` [int](#)

The number of top elements to retrieve.

### Returns

[IEnumerable](#) <[WaveletMatrixCore.ValueWithFrequency](#)>

An enumerable collection of [WaveletMatrixCore.ValueWithFrequency](#) records, ordered by frequency in descending order.

### Exceptions

#### [ArgumentOutOfRangeException](#)

Thrown if `start`, `end` or `k` is out of range.

#### [ArgumentException](#)

Thrown if `start` is greater than or equal `end`.

# Class WaveletMatrixCore.ValueWithFrequency

Namespace: [BelNytheraSeiche.WaveletMatrix](#)

Assembly: BelNytheraSeiche.WaveletMatrix.dll

Represents a value and its frequency of occurrence.

```
public record WaveletMatrixCore.ValueWithFrequency :  
IEquatable<WaveletMatrixCore.ValueWithFrequency>
```

## Inheritance

[object](#) ← WaveletMatrixCore.ValueWithFrequency

## Implements

[IEquatable](#)<[WaveletMatrixCore.ValueWithFrequency](#)>

## Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

## Constructors

### ValueWithFrequency(int, int)

Represents a value and its frequency of occurrence.

```
public ValueWithFrequency(int Value, int Frequency)
```

## Parameters

**Value** [int](#)

The element's value.

**Frequency** [int](#)

The number of times the value occurred.

# Properties

## Frequency

The number of times the value occurred.

```
public int Frequency { get; init; }
```

Property Value

[int ↗](#)

## Value

The element's value.

```
public int Value { get; init; }
```

Property Value

[int ↗](#)

# Class WaveletMatrixGeneric<T>

Namespace: [BelNytheraSeiche.WaveletMatrix](#)

Assembly: BelNytheraSeiche.WaveletMatrix.dll

A generic, compressed data structure for sequences of any comparable type. It provides fast Rank, Select, Quantile, and other advanced queries. This class uses coordinate compression internally, making it highly memory-efficient for data with a small alphabet size.

```
public sealed class WaveletMatrixGeneric<T> where T : IComparable<T>
```

## Type Parameters

T

The type of elements in the sequence. Must implement [IComparable<T>](#).

## Inheritance

[object](#) ← WaveletMatrixGeneric<T>

## Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

## Examples

```
var data = new[] { "apple", "banana", "apple", "orange" };
var wm = WaveletMatrixGeneric<string>.Create(data);
int rank = wm.Rank(3, "apple"); // Counts "apple" in the range [0, 3). Result: 2
string value = wm.Access(1);    // Gets the element at index 1. Result: "banana"
```

## Properties

### Size

Gets the total number of elements in the sequence.

```
public int Size { get; }
```

Property Value

[int ↗](#)

## Methods

### Access(int)

Gets the value of the element at the specified zero-based position.

```
public T Access(int k)
```

Parameters

[k int ↗](#)

The zero-based index of the element to retrieve.

Returns

T

The integer value at the specified position.

Exceptions

[ArgumentOutOfRangeException ↗](#)

Thrown if [k](#) is out of range.

### Create(IEnumerable<T>)

Creates a new instance of the [WaveletMatrixGeneric<T>](#) from a sequence of elements. This factory method performs coordinate compression on the input sequence before building the core data structure.

```
public static WaveletMatrixGeneric<T> Create(IEnumerable<T> sequence)
```

Parameters

`sequence` [IEnumerable](#) <T>

The input sequence of data to be indexed.

Returns

[WaveletMatrixGeneric](#)<T>

A new, fully initialized [WaveletMatrixGeneric](#)<T> instance.

Exceptions

[ArgumentNullException](#)

Thrown if `sequence` is null.

## Deserialize(byte[], [IGenericSerializer](#)<T>?)

Deserializes a byte array into a [WaveletMatrixGeneric](#)<T> instance.

```
public static WaveletMatrixGeneric<T> Deserialize(byte[] data,  
WaveletMatrixGeneric<T>.IGenericSerializer<T>? genericSerializer = null)
```

Parameters

`data` [byte](#)[]

The byte array containing the serialized data.

`genericSerializer` [WaveletMatrixGeneric](#)<T>.IGenericSerializer<T>

An optional serializer for handling the generic type T. If null, a default serializer for [byte](#), [int](#), or [char](#) will be used if applicable. If a default serializer is not available for T, an exception will be thrown.

Returns

[WaveletMatrixGeneric](#)<T>

A new, deserialized instance of [WaveletMatrixGeneric](#)<T>.

Exceptions

## [ArgumentNullException](#)

Thrown if `data` is null.

## [ArgumentException](#)

Thrown if `genericSerializer` is null and a default serializer for type `T` cannot be found.

# Deserialize(Stream, IGenericSerializer<T>?)

Deserializes a [WaveletMatrixGeneric<T>](#) instance from a stream. It verifies the file format, type identifier, and checksum.

```
public static WaveletMatrixGeneric<T> Deserialize(Stream stream,  
    WaveletMatrixGeneric<T>.IGenericSerializer<T>? genericSerializer = null)
```

## Parameters

### `stream` [Stream](#)

The stream to read the serialized data from.

### `genericSerializer` [WaveletMatrixGeneric<T>.IGenericSerializer<T>](#)

An optional serializer for handling the generic type `T`. If null, a default serializer for [byte](#), [int](#), or [char](#) will be used if applicable. If a default serializer is not available for `T`, an exception will be thrown.

## Returns

### [WaveletMatrixGeneric<T>](#)

A new, deserialized instance.

## Exceptions

### [ArgumentException](#)

Thrown if `genericSerializer` is null and a default serializer for type `T` cannot be found.

### [InvalidDataException](#)

Thrown if the data format is unsupported, the type is incompatible, or the data is corrupt.

## Deserialize(string, IGenericSerializer<T>?)

Deserializes a [WaveletMatrixGeneric<T>](#) instance from the specified file.

```
public static WaveletMatrixGeneric<T> Deserialize(string file,  
WaveletMatrixGeneric<T>.IGenericSerializer<T>? genericSerializer = null)
```

### Parameters

#### `file` [string](#)

The path of the file to read from.

#### `genericSerializer` [WaveletMatrixGeneric<T>.IGenericSerializer<T>](#)

An optional serializer for handling the generic type `T`. If null, a default serializer for [byte](#), [int](#), or [char](#) will be used if applicable. If a default serializer is not available for `T`, an exception will be thrown.

### Returns

#### [WaveletMatrixGeneric<T>](#)

A new, deserialized instance.

### Exceptions

#### [ArgumentNullException](#)

Thrown if `file` is null.

#### [ArgumentException](#)

Thrown if `genericSerializer` is null and a default serializer for type `T` cannot be found.

## LargerValue(int, int, T, T)

Finds the successor of a given value in the specified range [start, end). The successor is the smallest value in the range that is strictly larger than the given `value`.

```
public T LargerValue(int start, int end, T value, T defaultValue = default)
```

## Parameters

**start** [int](#)

The inclusive start of the range.

**end** [int](#)

The exclusive end of the range.

**value** T

The reference value.

**defaultValue** T

The value to return if no successor is found.

## Returns

T

The successor value, or **defaultValue** if no such value is found.

## Exceptions

[ArgumentOutOfRangeException](#)

Thrown if **start** or **end** is out of range.

[ArgumentException](#)

Thrown if **start** is greater than or equal **end**.

## LargerValue(T, T)

Finds the successor of a given value. The successor is the smallest value in the range that is strictly larger than the given **value**.

```
public T LargerValue(T value, T defaultValue = default)
```

## Parameters

**value** T

The reference value.

**defaultValue** T

The value to return if no successor is found.

Returns

T

The successor value, or **defaultValue** if no such value is found.

Remarks

This method is a shorthand for calling LargerValue(0, [Size](#), **value**, **defaultValue**).

## Quantile(int)

Finds the k-th smallest value in the entire sequence.

```
public T Quantile(int k)
```

Parameters

**k** [int](#)

The zero-based rank of the value to find (e.g., k=0 for the smallest value).

Returns

T

The k-th smallest value.

Remarks

This method is a shorthand for calling Quantile(0, [Size](#), **k**).

Exceptions

## [ArgumentOutOfRangeException](#)

Thrown if `k` is out of the valid range [0, Size).

## Quantile(int, int, int)

Finds the k-th smallest value within the specified range [start, end).

```
public T Quantile(int start, int end, int k)
```

### Parameters

`start` [int](#)

The inclusive start of the range.

`end` [int](#)

The exclusive end of the range.

`k` [int](#)

The zero-based rank of the value to find (e.g., k=0 for the smallest value).

### Returns

T

The k-th smallest value in the range.

### Exceptions

#### [ArgumentOutOfRangeException](#)

Thrown if `start`, `end` or `k` is out of range.

#### [ArgumentException](#)

Thrown if `start` is greater than or equal `end`.

#### [ArgumentException](#)

Thrown if `k` is out of the valid range [0, end-start).

## RangeCount(int, int, T)

Counts the number of occurrences of a specified value within the range [start, end).

```
public int RangeCount(int start, int end, T value)
```

### Parameters

**start** [int](#)

The inclusive start of the range.

**end** [int](#)

The exclusive end of the range.

**value** T

The value to count.

### Returns

[int](#)

The number of occurrences of the value in the range.

### Exceptions

[ArgumentOutOfRangeException](#)

Thrown if **start** or **end** is out of range.

[ArgumentException](#)

Thrown if **start** is greater than or equal **end**.

## RangeCount(T)

Counts the number of occurrences of a specified value.

```
public int RangeCount(T value)
```

## Parameters

**value** T

The value to count.

## Returns

[int ↗](#)

The number of occurrences of the value.

## Remarks

This method is a shorthand for calling RangeCount(0, [Size](#), **value**).

## RangeFreq(int, int, T, T)

Counts the number of elements in a position range [start, end) whose values are within the value range [minValue, maxValue).

```
public int RangeFreq(int start, int end, T minValue, T maxValue)
```

## Parameters

**start** [int ↗](#)

The inclusive start of the position range.

**end** [int ↗](#)

The exclusive end of the position range.

**minValue** T

The inclusive start of the value range.

**maxValue** T

The exclusive end of the value range.

## Returns

## [int](#)

The count of elements matching the criteria.

## Exceptions

### [ArgumentOutOfRangeException](#)

Thrown if `start` or `end` is out of range.

### [ArgumentException](#)

Thrown if `start` is greater than or equal `end`.

### [ArgumentException](#)

Thrown if `minValue` is greater than or equal `maxValue`.

## RangeFreq(T, T)

Counts the number of elements whose values are within the value range [minValue, maxValue).

```
public int RangeFreq(T minValue, T maxValue)
```

## Parameters

### `minValue` T

The inclusive start of the value range.

### `maxValue` T

The exclusive end of the value range.

## Returns

### [int](#)

The count of elements matching the criteria.

## Remarks

This method is a shorthand for calling `RangeFreq(0, Size, minValue, maxValue)`.

## Exceptions

### [ArgumentException](#)

Thrown if `minValue` is greater than or equal `maxValue`.

## RangeMode()

Finds the most frequent value (the mode) and its frequency.

```
public WaveletMatrixGeneric<T>.ValueWithFrequency<T> RangeMode()
```

Returns

### [WaveletMatrixGeneric<T>.ValueWithFrequency<T>](#)

A [WaveletMatrixGeneric<T>.ValueWithFrequency<Tx>](#) record containing the mode and its count.

Remarks

This method is a shorthand for calling `RangeMode(0, Size)`.

## RangeMode(int, int)

Finds the most frequent value (the mode) and its frequency within the specified range [start, end).

```
public WaveletMatrixGeneric<T>.ValueWithFrequency<T> RangeMode(int start, int end)
```

Parameters

### `start` [int](#)

The inclusive start of the range.

### `end` [int](#)

The exclusive end of the range.

Returns

## WaveletMatrixGeneric<T>.ValueWithFrequency<T>

A [WaveletMatrixGeneric<T>.ValueWithFrequency<Tx>](#) record containing the mode and its count.

## Exceptions

### [ArgumentOutOfRangeException](#)

Thrown if `start` or `end` is out of range.

### [ArgumentException](#)

Thrown if `start` is greater than or equal `end`.

## Rank(int, T)

Counts the number of occurrences of a specified value in the prefix of the sequence [0, k).

```
public int Rank(int k, T value)
```

## Parameters

### `k` [int](#)

The exclusive end of the range [0, k).

### `value` `T`

The value to count.

## Returns

### [int](#)

The number of times the value appears in the specified prefix.

## Exceptions

### [ArgumentOutOfRangeException](#)

Thrown if `k` is out of range.

## Select(int, T)

Finds the zero-based position of the k-th occurrence of a specified value.

```
public int Select(int k, T value)
```

### Parameters

**k** [int](#)

The one-based rank of the occurrence to find (e.g., k=1 for the first occurrence).

**value** [T](#)

The value to search for.

### Returns

[int](#)

The zero-based index of the k-th occurrence of the value, or -1 if not found.

### Exceptions

[ArgumentOutOfRangeException](#)

Thrown if **k** is out of range.

## Serialize(WaveletMatrixGeneric<T>, IGenericSerializer<T>?, SerializationOptions?)

Serializes the [WaveletMatrixGeneric<T>](#) instance into a byte array.

```
public static byte[] Serialize(WaveletMatrixGeneric<T> obj,  
WaveletMatrixGeneric<T>.IGenericSerializer<T>? genericSerializer = null,  
SerializationOptions? options = null)
```

### Parameters

**obj** [WaveletMatrixGeneric<T>](#)

The instance to serialize.

`genericSerializer` [WaveletMatrixGeneric<T>.IGenericSerializer<T>](#)

An optional serializer for handling the generic type `T`. If null, a default serializer for `byte`, `int`, or `char` will be used if applicable. If a default serializer is not available for `T`, an exception will be thrown.

`options` [SerializationOptions](#)

Serialization options, such as the compression level. If null, [Default](#) will be used.

Returns

`byte[]`

A byte array containing the serialized data.

Exceptions

[ArgumentNullException](#)

Thrown if `obj` is null.

[ArgumentException](#)

Thrown if `genericSerializer` is null and a default serializer for type `T` cannot be found.

## Serialize(WaveletMatrixGeneric<T>, Stream, IGenericSerializer<T>?, SerializationOptions?)

Serializes the [WaveletMatrixGeneric<T>](#) instance to a stream. The data is compressed using Brotli and includes a checksum for integrity verification.

```
public static void Serialize(WaveletMatrixGeneric<T> obj, Stream stream,
    WaveletMatrixGeneric<T>.IGenericSerializer<T>? genericSerializer = null,
    SerializationOptions? options = null)
```

Parameters

`obj` [WaveletMatrixGeneric<T>](#)

The instance to serialize.

## stream [Stream](#)

The stream to write the serialized data to.

## genericSerializer [WaveletMatrixGeneric<T>.IGenericSerializer<T>](#)

An optional serializer for handling the generic type **T**. If null, a default serializer for [byte](#), [int](#), or [char](#) will be used if applicable. If a default serializer is not available for **T**, an exception will be thrown.

## options [SerializationOptions](#)

Serialization options, such as the compression level. If null, [Default](#) will be used.

## Exceptions

### [ArgumentNullException](#)

Thrown if **obj** or **stream** is null.

### [ArgumentException](#)

Thrown if **genericSerializer** is null and a default serializer for type **T** cannot be found.

## Serialize(WaveletMatrixGeneric<T>, string, IGenericSerializer<T>?, SerializationOptions?)

Serializes the [WaveletMatrixGeneric<T>](#) instance to the specified file.

```
public static void Serialize(WaveletMatrixGeneric<T> obj, string file,  
    WaveletMatrixGeneric<T>.IGenericSerializer<T>? genericSerializer = null,  
    SerializationOptions? options = null)
```

## Parameters

### **obj** [WaveletMatrixGeneric<T>](#)

The instance to serialize.

### **file** [string](#)

The path of the file to write to.

## genericSerializer [WaveletMatrixGeneric<T>.IGenericSerializer<T>](#)

An optional serializer for handling the generic type `T`. If null, a default serializer for `byte`, `int`, or `char` will be used if applicable. If a default serializer is not available for `T`, an exception will be thrown.

## options [SerializationOptions](#)

Serialization options, such as the compression level. If null, `Default` will be used.

## Exceptions

### [ArgumentNullException](#)

Thrown if `obj` or `file` is null.

### [ArgumentException](#)

Thrown if `genericSerializer` is null and a default serializer for type `T` cannot be found.

## SmallerValue(int, int, T, T)

Finds the predecessor of a given value in the specified range [start, end). The predecessor is the largest value in the range that is strictly smaller than the given `value`.

```
public T SmallerValue(int start, int end, T value, T defaultValue = default)
```

## Parameters

### `start` [int](#)

The inclusive start of the range.

### `end` [int](#)

The exclusive end of the range.

### `value` `T`

The reference value.

### `defaultValue` `T`

The value to return if no predecessor is found.

## Returns

T

The predecessor value, or `defaultValue` if no such value is found.

## Exceptions

### [ArgumentOutOfRangeException](#)

Thrown if `start` or `end` is out of range.

### [ArgumentException](#)

Thrown if `start` is greater than or equal `end`.

## SmallerValue(T, T)

Finds the predecessor of a given value. The predecessor is the largest value in the range that is strictly smaller than the given `value`.

```
public T SmallerValue(T value, T defaultValue = default)
```

## Parameters

`value` T

The reference value.

`defaultValue` T

The value to return if no predecessor is found.

## Returns

T

The predecessor value, or `defaultValue` if no such value is found.

## Remarks

This method is a shorthand for calling `SmallerValue(0, Size, value, defaultValue)`.

## TopK(int)

Finds the top K most frequent values and their frequencies.

```
public IEnumerable<WaveletMatrixGeneric<T>.ValueWithFrequency<T>> TopK(int k)
```

### Parameters

[k \[int\]\(#\)](#)

The number of top elements to retrieve.

### Returns

[IEnumerable](#) <[WaveletMatrixGeneric](#)<T>.ValueWithFrequency<T>>

An enumerable collection of [WaveletMatrixGeneric<T>.ValueWithFrequency<Tx>](#) records, ordered by frequency in descending order.

### Remarks

This method is a shorthand for calling TopK(0, [Size](#), [k](#)).

### Exceptions

[ArgumentOutOfRangeException](#)

Thrown if [k](#) is out of range.

## TopK(int, int, int)

Finds the top K most frequent values and their frequencies within the specified range [start, end).

```
public IEnumerable<WaveletMatrixGeneric<T>.ValueWithFrequency<T>> TopK(int start, int end,  
int k)
```

### Parameters

[start \[int\]\(#\)](#)

The inclusive start of the range.

end [int](#)

The exclusive end of the range.

k [int](#)

The number of top elements to retrieve.

Returns

[IEnumerable](#)<[WaveletMatrixGeneric](#)<T>.ValueWithFrequency<T>>

An enumerable collection of [WaveletMatrixGeneric](#)<T>.ValueWithFrequency<Tx> records, ordered by frequency in descending order.

Exceptions

[ArgumentOutOfRangeException](#)

Thrown if `start`, `end` or `k` is out of range.

[ArgumentException](#)

Thrown if `start` is greater than or equal `end`.

## TryLargerValue(int, int, T, out T)

Finds the successor of a given value in the specified range [start, end). The successor is the smallest value in the range that is strictly larger than the given `value`. This is the robust version that returns a boolean indicating success.

```
public bool TryLargerValue(int start, int end, T value, out T result)
```

Parameters

`start` [int](#)

The inclusive start of the range.

end [int](#)

The exclusive end of the range.

**value** T

The reference value.

**result** T

When this method returns, contains the successor value if found; otherwise, the default value for the type.

Returns

[bool](#)

`true` if a predecessor value was found; otherwise, `false`.

Exceptions

[ArgumentOutOfRangeException](#)

Thrown if `start` or `end` is out of range.

[ArgumentException](#)

Thrown if `start` is greater than or equal `end`.

## TryLargerValue(T, out T)

Finds the successor of a given value. The successor is the smallest value in the range that is strictly larger than the given `value`. This is the robust version that returns a boolean indicating success.

```
public bool TryLargerValue(T value, out T result)
```

Parameters

**value** T

The reference value.

**result** T

When this method returns, contains the successor value if found; otherwise, the default value for the type.

Returns

[bool](#)

`true` if a predecessor value was found; otherwise, `false`.

Remarks

This method is a shorthand for calling `TryLargerValue(0, Size, value, out result)`.

## TrySmallerValue(int, int, T, out T)

Finds the predecessor of a given value in the specified range [start, end). The predecessor is the largest value in the range that is strictly smaller than the given `value`. This is the robust version that returns a boolean indicating success.

```
public bool TrySmallerValue(int start, int end, T value, out T result)
```

Parameters

`start` [int](#)

The inclusive start of the range.

`end` [int](#)

The exclusive end of the range.

`value` T

The reference value.

`result` T

When this method returns, contains the successor value if found; otherwise, the default value for the type.

Returns

[bool](#)

`true` if a predecessor value was found; otherwise, `false`.

## Exceptions

### [ArgumentOutOfRangeException](#)

Thrown if `start` or `end` is out of range.

### [ArgumentException](#)

Thrown if `start` is greater than or equal `end`.

## TrySmallerValue(T, out T)

Finds the predecessor of a given value. The predecessor is the largest value in the range that is strictly smaller than the given `value`. This is the robust version that returns a boolean indicating success.

```
public bool TrySmallerValue(T value, out T result)
```

### Parameters

#### `value` T

The reference value.

#### `result` T

When this method returns, contains the successor value if found; otherwise, the default value for the type.

### Returns

#### [bool](#)

`true` if a predecessor value was found; otherwise, `false`.

### Remarks

This method is a shorthand for calling `TrySmallerValue(0, Size, value, out result)`.

# Class WaveletMatrixGeneric<T>.ByteSerializer

Namespace: [BelNytheraSeiche.WaveletMatrix](#)

Assembly: BelNytheraSeiche.WaveletMatrix.dll

Provides a default serializer for the [byte](#) type.

```
public sealed class WaveletMatrixGeneric<T>.ByteSerializer :  
    WaveletMatrixGeneric<T>.IGenericSerializer<byte>
```

## Inheritance

[object](#) ← WaveletMatrixGeneric<T>.ByteSerializer

## Implements

[WaveletMatrixGeneric<T>.IGenericSerializer<byte>](#)

## Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

# Properties

## Instance

Gets the singleton instance of the [WaveletMatrixGeneric<T>.ByteSerializer](#).

```
public static WaveletMatrixGeneric<T>.IGenericSerializer<byte> Instance { get; }
```

## Property Value

[WaveletMatrixGeneric<T>.IGenericSerializer<byte>](#)

## TypeIdentifier

Gets a unique 4-byte identifier for the type [byte](#). This identifier is written to the file header to ensure type safety during deserialization.

```
public byte[] TypeIdentifier { get; }
```

## Property Value

[byte](#)[]

## Examples

```
public byte[] TypeIdentifier => "INT_";
```

# Methods

## ReadResolveMap(Stream)

Reads the resolve map from the stream.

```
public byte[] ReadResolveMap(Stream stream)
```

### Parameters

[stream](#) [Stream](#)[]

The stream to read from. Typically a [BrotliStream](#) for decompression.

### Returns

[byte](#)[]

An array containing the deserialized unique values.

## WriteResolveMap(Stream, byte[])

Writes the resolve map (the dictionary of unique values) to the stream.

```
public void WriteResolveMap(Stream stream, byte[] resolveMap)
```

## Parameters

**stream** [Stream](#)

The stream to write to. Typically a [BrotliStream](#) for compression.

**resolveMap** [byte](#)[]

The array of unique values to serialize.

# Class WaveletMatrixGeneric<T>.CharSerializer

Namespace: [BelNytheraSeiche.WaveletMatrix](#)

Assembly: BelNytheraSeiche.WaveletMatrix.dll

Provides a default serializer for the [char](#) type.

```
public sealed class WaveletMatrixGeneric<T>.CharSerializer :  
    WaveletMatrixGeneric<T>.IGenericSerializer<char>
```

## Inheritance

[object](#) ← WaveletMatrixGeneric<T>.CharSerializer

## Implements

[WaveletMatrixGeneric<T>.IGenericSerializer<char>](#)

## Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

# Properties

## Instance

Gets the singleton instance of the [WaveletMatrixGeneric<T>.CharSerializer](#).

```
public static WaveletMatrixGeneric<T>.IGenericSerializer<char> Instance { get; }
```

## Property Value

[WaveletMatrixGeneric<T>.IGenericSerializer<char>](#)

## TypeIdentifier

Gets a unique 4-byte identifier for the type [char](#). This identifier is written to the file header to ensure type safety during deserialization.

```
public byte[] TypeIdentifier { get; }
```

## Property Value

[byte](#)[]

## Examples

```
public byte[] TypeIdentifier => "INT_";
```

# Methods

## ReadResolveMap(Stream)

Reads the resolve map from the stream.

```
public char[] ReadResolveMap(Stream stream)
```

### Parameters

[stream](#) [Stream](#)[]

The stream to read from. Typically a [BrotliStream](#) for decompression.

### Returns

[char](#)[][]

An array containing the deserialized unique values.

## WriteResolveMap(Stream, char[])

Writes the resolve map (the dictionary of unique values) to the stream.

```
public void WriteResolveMap(Stream stream, char[] resolveMap)
```

## Parameters

**stream** [Stream](#)

The stream to write to. Typically a [BrotliStream](#) for compression.

**resolveMap** [char](#)[]

The array of unique values to serialize.

# Interface

## WaveletMatrixGeneric<T>.IGenericSerializer<T X>

Namespace: [BelNytheraSeiche.WaveletMatrix](#)

Assembly: BelNytheraSeiche.WaveletMatrix.dll

Defines the contract for serializing and deserializing elements of type **Tx**.

```
public interface WaveletMatrixGeneric<T>.IGenericSerializer<Tx> where Tx : IComparable<Tx>
```

### Type Parameters

**Tx**

The type to be serialized.

## Properties

### TypeIdentifier

Gets a unique 4-byte identifier for the type **Tx**. This identifier is written to the file header to ensure type safety during deserialization.

```
byte[] TypeIdentifier { get; }
```

### Property Value

[byte](#)[]

### Examples

```
public byte[] TypeIdentifier => "INT_";
```

# Methods

## ReadResolveMap(Stream)

Reads the resolve map from the stream.

```
Tx[] ReadResolveMap(Stream stream)
```

Parameters

**stream** [Stream](#)

The stream to read from. Typically a [BrotliStream](#) for decompression.

Returns

```
Tx[]
```

An array containing the deserialized unique values.

## WriteResolveMap(Stream, Tx[])

Writes the resolve map (the dictionary of unique values) to the stream.

```
void WriteResolveMap(Stream stream, Tx[] resolveMap)
```

Parameters

**stream** [Stream](#)

The stream to write to. Typically a [BrotliStream](#) for compression.

**resolveMap** [Tx\[\]](#)

The array of unique values to serialize.

# Class WaveletMatrixGeneric<T>.Int32Serializer

Namespace: [BelNytheraSeiche.WaveletMatrix](#)

Assembly: BelNytheraSeiche.WaveletMatrix.dll

Provides a default serializer for the [int](#) type.

```
public sealed class WaveletMatrixGeneric<T>.Int32Serializer :  
    WaveletMatrixGeneric<T>.IGenericSerializer<int>
```

## Inheritance

[object](#) ← WaveletMatrixGeneric<T>.Int32Serializer

## Implements

[WaveletMatrixGeneric<T>.IGenericSerializer<int](#)

## Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

# Properties

## Instance

Gets the singleton instance of the [WaveletMatrixGeneric<T>.Int32Serializer](#).

```
public static WaveletMatrixGeneric<T>.IGenericSerializer<int> Instance { get; }
```

## Property Value

[WaveletMatrixGeneric<T>.IGenericSerializer<int](#)

## TypeIdentifier

Gets a unique 4-byte identifier for the type [int](#). This identifier is written to the file header to ensure type safety during deserialization.

```
public byte[] TypeIdentifier { get; }
```

## Property Value

[byte](#)[]

## Examples

```
public byte[] TypeIdentifier => "INT_";
```

# Methods

## ReadResolveMap(Stream)

Reads the resolve map from the stream.

```
public int[] ReadResolveMap(Stream stream)
```

### Parameters

[stream](#) [Stream](#)[]

The stream to read from. Typically a [BrotliStream](#) for decompression.

### Returns

[int](#)[]

An array containing the deserialized unique values.

## WriteResolveMap(Stream, int[])

Writes the resolve map (the dictionary of unique values) to the stream.

```
public void WriteResolveMap(Stream stream, int[] resolveMap)
```

## Parameters

**stream** [Stream](#)

The stream to write to. Typically a [BrotliStream](#) for compression.

**resolveMap** [int](#)[]

The array of unique values to serialize.

# Class

## WaveletMatrixGeneric<T>.ValueWithFrequency<Tx>

Namespace: [BelNytheraSeiche.WaveletMatrix](#)

Assembly: BelNytheraSeiche.WaveletMatrix.dll

Represents a value and its frequency of occurrence.

```
public record WaveletMatrixGeneric<T>.ValueWithFrequency<Tx> :  
IEquatable<WaveletMatrixGeneric<T>.ValueWithFrequency<Tx>>
```

### Type Parameters

Tx

#### Inheritance

[object](#) ← WaveletMatrixGeneric<T>.ValueWithFrequency<Tx>

#### Implements

[IEquatable](#)<[WaveletMatrixGeneric](#)<T>.ValueWithFrequency<Tx>>

#### Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

# Constructors

## ValueWithFrequency(Tx, int)

Represents a value and its frequency of occurrence.

```
public ValueWithFrequency(Tx Value, int Frequency)
```

### Parameters

Value Tx

The element's value.

### Frequency [int ↗](#)

The number of times the value occurred.

## Properties

### Frequency

The number of times the value occurred.

```
public int Frequency { get; init; }
```

### Property Value

[int ↗](#)

### Value

The element's value.

```
public Tx Value { get; init; }
```

### Property Value

Tx