|  |
| --- |
| [Type the company name] |
| P3-Andrew Reference |
| [Type the document subtitle] |

|  |
| --- |
| [Type the author name]  [Pick the date] |

Table of Contents

[Press ALT+A and F9 to update the TOC and other fields]

P3-Andrew Reference

Namespaces

[AlgorithmTesting](#topic_000000000000001C), [P3\_Andrew](#topic_0000000000000000), [P3\_Andrew.Sorting\_Algorithms](#topic_0000000000000003)

AlgorithmTesting Namespace

Classes

[UnitTest1](#topic_000000000000001D)

UnitTest1 Class

[System.Object](https://docs.microsoft.com/en-us/dotnet/api/system.object)

**AlgorithmTesting.UnitTest1**

|  |  |
| --- | --- |
| C# |  |
| [TestClass()] public class UnitTest1 | |

Requirements

**Namespace:**[AlgorithmTesting](#topic_000000000000001C)

**Assembly:** AlgorithmTesting (in AlgorithmTesting.dll)

Methods

[Average](#topic_0000000000000022), [BubbleSort](#topic_0000000000000023), [CombSort](#topic_0000000000000024), [Equals](https://docs.microsoft.com/en-us/dotnet/api/system.object.equals) (inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object)), [Finalize](https://docs.microsoft.com/en-us/dotnet/api/system.object.finalize) (inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object)), [GetHashCode](https://docs.microsoft.com/en-us/dotnet/api/system.object.gethashcode) (inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object)), [GetType](https://docs.microsoft.com/en-us/dotnet/api/system.object.gettype) (inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object)), [HeapSort](#topic_0000000000000025), [InsertionSort](#topic_0000000000000026), [MemberwiseClone](https://docs.microsoft.com/en-us/dotnet/api/system.object.memberwiseclone) (inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object)), [MergeSort](#topic_0000000000000027), [PrepTest](#topic_0000000000000021), [QuickSort](#topic_0000000000000028), [SelectionSort](#topic_0000000000000029), [ShellSort](#topic_000000000000002A), [ToString](https://docs.microsoft.com/en-us/dotnet/api/system.object.tostring) (inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object))

Fields

[AVERAGEVALUES](#topic_000000000000001F), [DATASIZE](#topic_000000000000001E), [testList](#topic_0000000000000020)

UnitTest1.Average Method

|  |  |
| --- | --- |
| C# |  |
| public [long](https://docs.microsoft.com/en-us/dotnet/api/system.int64) Average(  [long](https://docs.microsoft.com/en-us/dotnet/api/system.int64)[] *l* ) | |

Parameters

l

|  |
| --- |
|  |

Source code

|  |
| --- |
| public long Average(long[] l)  {  long sum = 0;  foreach(long val in l)  {  sum += val;  }    return sum / (long) AVERAGEVALUES;  } |

See Also

Applies to: [UnitTest1](#topic_000000000000001D)

UnitTest1.BubbleSort Method

|  |  |
| --- | --- |
| C# |  |
| [TestMethod()] public [void](https://docs.microsoft.com/en-us/dotnet/api/system.void) BubbleSort() | |

Source code

|  |
| --- |
| [TestMethod]  public void BubbleSort()  {  List<int> dataSet = new List<int>(testList);  long[] measures = new long[AVERAGEVALUES];    for (int i = 0; i < AVERAGEVALUES; i++)  {  Stopwatch stopwatch = new Stopwatch();  stopwatch.Start();    P3\_Andrew.Sorting\_Algorithms.BubbleSort.Sort(dataSet);    stopwatch.Stop();  TimeSpan ts = stopwatch.Elapsed;  measures[i] = ts.Ticks;  }  Debug.WriteLine("Average Runtime: " + Average(measures));  } |

See Also

Applies to: [UnitTest1](#topic_000000000000001D)

UnitTest1.CombSort Method

|  |  |
| --- | --- |
| C# |  |
| [TestMethod()] public [void](https://docs.microsoft.com/en-us/dotnet/api/system.void) CombSort() | |

Source code

|  |
| --- |
| [TestMethod]  public void CombSort()  {  List<int> dataSet = new List<int>(testList);  long[] measures = new long[AVERAGEVALUES];    for (int i = 0; i < AVERAGEVALUES; i++)  {  Stopwatch stopwatch = new Stopwatch();  stopwatch.Start();    P3\_Andrew.Sorting\_Algorithms.CombSort.Sort(dataSet);    stopwatch.Stop();  TimeSpan ts = stopwatch.Elapsed;  measures[i] = ts.Ticks;  }  Debug.WriteLine("Average Runtime: " + Average(measures));  } |

See Also

Applies to: [UnitTest1](#topic_000000000000001D)

UnitTest1.HeapSort Method

|  |  |
| --- | --- |
| C# |  |
| [TestMethod()] public [void](https://docs.microsoft.com/en-us/dotnet/api/system.void) HeapSort() | |

Source code

|  |
| --- |
| [TestMethod]  public void HeapSort()  {  List<int> dataSet = new List<int>(testList);  long[] measures = new long[AVERAGEVALUES];    for (int i = 0; i < AVERAGEVALUES; i++)  {  Stopwatch stopwatch = new Stopwatch();  stopwatch.Start();    P3\_Andrew.Sorting\_Algorithms.HeapSort.Sort(dataSet);    stopwatch.Stop();  TimeSpan ts = stopwatch.Elapsed;  measures[i] = ts.Ticks;  }  Debug.WriteLine("Average Runtime: " + Average(measures)); ;  } |

See Also

Applies to: [UnitTest1](#topic_000000000000001D)

UnitTest1.InsertionSort Method

|  |  |
| --- | --- |
| C# |  |
| [TestMethod()] public [void](https://docs.microsoft.com/en-us/dotnet/api/system.void) InsertionSort() | |

Source code

|  |
| --- |
| [TestMethod]  public void InsertionSort()  {  List<int> dataSet = new List<int>(testList);  long[] measures = new long[AVERAGEVALUES];    for (int i = 0; i < AVERAGEVALUES; i++)  {  Stopwatch stopwatch = new Stopwatch();  stopwatch.Start();    P3\_Andrew.Sorting\_Algorithms.InsertionSort.Sort(dataSet);    stopwatch.Stop();  TimeSpan ts = stopwatch.Elapsed;  measures[i] = ts.Ticks;  }  Debug.WriteLine("Average Runtime: " + Average(measures));  } |

See Also

Applies to: [UnitTest1](#topic_000000000000001D)

UnitTest1.MergeSort Method

|  |  |
| --- | --- |
| C# |  |
| [TestMethod()] public [void](https://docs.microsoft.com/en-us/dotnet/api/system.void) MergeSort() | |

Source code

|  |
| --- |
| [TestMethod]  public void MergeSort()  {  List<int> dataSet = new List<int>(testList);  long[] measures = new long[AVERAGEVALUES];    for (int i = 0; i < AVERAGEVALUES; i++)  {  Stopwatch stopwatch = new Stopwatch();  stopwatch.Start();    P3\_Andrew.Sorting\_Algorithms.MergeSort.Sort(dataSet);    stopwatch.Stop();  TimeSpan ts = stopwatch.Elapsed;  measures[i] = ts.Ticks;  }  Debug.WriteLine("Average Runtime: " + Average(measures));  } |

See Also

Applies to: [UnitTest1](#topic_000000000000001D)

UnitTest1.PrepTest Method

|  |  |
| --- | --- |
| C# |  |
| [TestInitialize()] public [void](https://docs.microsoft.com/en-us/dotnet/api/system.void) PrepTest() | |

Source code

|  |
| --- |
| [TestInitialize]  public void PrepTest()  {     BinaryFormatter formatter = new BinaryFormatter();  Stream stream = new FileStream(DATASIZE.ToString(), FileMode.Open, FileAccess.Read);  testList = (List<int>)formatter.Deserialize(stream);  stream.Close();  } |

See Also

Applies to: [UnitTest1](#topic_000000000000001D)

UnitTest1.QuickSort Method

|  |  |
| --- | --- |
| C# |  |
| public [void](https://docs.microsoft.com/en-us/dotnet/api/system.void) QuickSort() | |

Source code

|  |
| --- |
| public void QuickSort()  {  List<int> dataSet = new List<int>(testList);  long[] measures = new long[AVERAGEVALUES];    for (int i = 0; i < AVERAGEVALUES; i++)  {  Stopwatch stopwatch = new Stopwatch();  stopwatch.Start();    P3\_Andrew.Sorting\_Algorithms.QuickSort.Sort(dataSet);    stopwatch.Stop();  TimeSpan ts = stopwatch.Elapsed;  measures[i] = ts.Ticks;  }  Debug.WriteLine("Average Runtime: " + Average(measures));  } |

See Also

Applies to: [UnitTest1](#topic_000000000000001D)

UnitTest1.SelectionSort Method

|  |  |
| --- | --- |
| C# |  |
| [TestMethod()] public [void](https://docs.microsoft.com/en-us/dotnet/api/system.void) SelectionSort() | |

Source code

|  |
| --- |
| [TestMethod]  public void SelectionSort()  {  List<int> dataSet = new List<int>(testList);  long[] measures = new long[AVERAGEVALUES];    for (int i = 0; i < AVERAGEVALUES; i++)  {  Stopwatch stopwatch = new Stopwatch();  stopwatch.Start();    P3\_Andrew.Sorting\_Algorithms.SelectionSort.Sort(dataSet);    stopwatch.Stop();  TimeSpan ts = stopwatch.Elapsed;  measures[i] = ts.Ticks;  }  Debug.WriteLine("Average Runtime: " + Average(measures));  } |

See Also

Applies to: [UnitTest1](#topic_000000000000001D)

UnitTest1.ShellSort Method

|  |  |
| --- | --- |
| C# |  |
| [TestMethod()] public [void](https://docs.microsoft.com/en-us/dotnet/api/system.void) ShellSort() | |

Source code

|  |
| --- |
| [TestMethod]  public void ShellSort()  {  List<int> dataSet = new List<int>(testList);  long[] measures = new long[AVERAGEVALUES];    for (int i = 0; i < AVERAGEVALUES; i++)  {  Stopwatch stopwatch = new Stopwatch();  stopwatch.Start();    P3\_Andrew.Sorting\_Algorithms.SelectionSort.Sort(dataSet);    stopwatch.Stop();  TimeSpan ts = stopwatch.Elapsed;  measures[i] = ts.Ticks;  }  Debug.WriteLine("Average Runtime: " + Average(measures));  } |

See Also

Applies to: [UnitTest1](#topic_000000000000001D)

testList Field

|  |  |
| --- | --- |
| C# |  |
| new private [List](https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.list-1)<[int](https://docs.microsoft.com/en-us/dotnet/api/system.int32)> testList | |

Source code

|  |
| --- |
| List<int> testList = new List<int>(); |

See Also

Applies to: [UnitTest1](#topic_000000000000001D)

AVERAGEVALUES Field

|  |  |
| --- | --- |
| C# |  |
| private const [int](https://docs.microsoft.com/en-us/dotnet/api/system.int32) AVERAGEVALUES = 10 | |

Source code

|  |
| --- |
| const int AVERAGEVALUES = 10; |

See Also

Applies to: [UnitTest1](#topic_000000000000001D)

DATASIZE Field

|  |  |
| --- | --- |
| C# |  |
| private const [int](https://docs.microsoft.com/en-us/dotnet/api/system.int32) DATASIZE = 25000 | |

Source code

|  |
| --- |
| const int DATASIZE = 25000; |

See Also

Applies to: [UnitTest1](#topic_000000000000001D)

P3\_Andrew Namespace

Classes

[Program](#topic_0000000000000001)

Program Class

[System.Object](https://docs.microsoft.com/en-us/dotnet/api/system.object)

**P3\_Andrew.Program**

|  |  |
| --- | --- |
| C# |  |
| internal class Program | |

Requirements

**Namespace:**[P3\_Andrew](#topic_0000000000000000)

**Assembly:** P3-Andrew (in P3-Andrew.exe)

Methods

[Equals](https://docs.microsoft.com/en-us/dotnet/api/system.object.equals) (inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object)), [Finalize](https://docs.microsoft.com/en-us/dotnet/api/system.object.finalize) (inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object)), [GetHashCode](https://docs.microsoft.com/en-us/dotnet/api/system.object.gethashcode) (inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object)), [GetType](https://docs.microsoft.com/en-us/dotnet/api/system.object.gettype) (inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object)), [Main](#topic_0000000000000002), [MemberwiseClone](https://docs.microsoft.com/en-us/dotnet/api/system.object.memberwiseclone) (inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object)), [ToString](https://docs.microsoft.com/en-us/dotnet/api/system.object.tostring) (inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object))

Program.Main Method

|  |  |
| --- | --- |
| C# |  |
| private static [void](https://docs.microsoft.com/en-us/dotnet/api/system.void) Main(  [string](https://docs.microsoft.com/en-us/dotnet/api/system.string)[] *args* ) | |

Parameters

args

|  |
| --- |
|  |

Source code

|  |
| --- |
| static void Main(string[] args)  {  int[] vals = { 1000, 5000, 10000, 15000, 20000, 25000, 30000, 35000, 40000, 45000, 50000 };  int toBeGenerated = 0;    Random rand = new Random();    List<int> list = new List<int>();    foreach(int val in vals)  {  for (int i = 0; i < vals[toBeGenerated]; i++)  {  list.Add(rand.Next());  }    IFormatter formatter = new BinaryFormatter();  Stream stream = new FileStream(val.ToString(), FileMode.Create, FileAccess.Write);  formatter.Serialize(stream, list);  stream.Close();  list.Clear();    toBeGenerated++;  }        } |

See Also

Applies to: [Program](#topic_0000000000000001)

P3\_Andrew.Sorting\_Algorithms Namespace

Classes

[BubbleSort](#topic_0000000000000004), [CombSort](#topic_0000000000000007), [HeapSort](#topic_000000000000000A), [InsertionSort](#topic_000000000000000E), [MergeSort](#topic_0000000000000010), [QuickSort](#topic_0000000000000013), [SelectionSort](#topic_0000000000000017), [ShellSort](#topic_000000000000001A)

BubbleSort Class

A class contatining the bubble sort algorithm and supporting methods.
Algorithm adapted from: https://www.w3resource.com/csharp-exercises/searching-and-sorting-algorithm/searching-and-sorting-algorithm-exercise-3.php

[System.Object](https://docs.microsoft.com/en-us/dotnet/api/system.object)

**P3\_Andrew.Sorting\_Algorithms.BubbleSort**

|  |  |
| --- | --- |
| C# |  |
| public static class BubbleSort | |

Requirements

**Namespace:**[P3\_Andrew.Sorting\_Algorithms](#topic_0000000000000003)

**Assembly:** P3-Andrew (in P3-Andrew.exe)

Methods

[Equals](https://docs.microsoft.com/en-us/dotnet/api/system.object.equals) (inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object)), [Finalize](https://docs.microsoft.com/en-us/dotnet/api/system.object.finalize) (inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object)), [GetHashCode](https://docs.microsoft.com/en-us/dotnet/api/system.object.gethashcode) (inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object)), [GetType](https://docs.microsoft.com/en-us/dotnet/api/system.object.gettype) (inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object)), [MemberwiseClone](https://docs.microsoft.com/en-us/dotnet/api/system.object.memberwiseclone) (inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object)), [Sort<T>](#topic_0000000000000005), [Swap<T>](#topic_0000000000000006), [ToString](https://docs.microsoft.com/en-us/dotnet/api/system.object.tostring) (inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object))

BubbleSort.Sort<T> Method

The method that calls the bubble sort

|  |  |
| --- | --- |
| C# |  |
| public static [void](https://docs.microsoft.com/en-us/dotnet/api/system.void) Sort<T>(  [List](https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.list-1)<T> *l* )  where T : [IComparable](https://docs.microsoft.com/en-us/dotnet/api/system.icomparable) | |

Type Parameters

T

|  |
| --- |
| Any type derived from IComparable |

Parameters

l

|  |
| --- |
| A list of type T |

Source code

|  |
| --- |
| public static void Sort<T>(List<T> l) where T : IComparable  {  for(int i = 0; i <= l.Count - 2; i++)  {  for(int j = 0; j <= l.Count - 2; j++)  {  if(l[j].CompareTo(l[j + 1]) > 0)  {  Swap(l, j, j + 1);  }  }  }  } |

See Also

Applies to: [BubbleSort](#topic_0000000000000004)

BubbleSort.Swap<T> Method

Supporting method to swap to values in a list

|  |  |
| --- | --- |
| C# |  |
| private static [void](https://docs.microsoft.com/en-us/dotnet/api/system.void) Swap<T>(  [List](https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.list-1)<T> *l*,  [int](https://docs.microsoft.com/en-us/dotnet/api/system.int32) *a*,  [int](https://docs.microsoft.com/en-us/dotnet/api/system.int32) *b* )  where T : [IComparable](https://docs.microsoft.com/en-us/dotnet/api/system.icomparable) | |

Type Parameters

T

|  |
| --- |
| Any type derived from IComparable |

Parameters

l

|  |
| --- |
| List with values that need to be swapped |

a

|  |
| --- |
| First index to be swapped |

b

|  |
| --- |
| Second index to be swapped |

Source code

|  |
| --- |
| private static void Swap<T>(List<T> l, int a, int b) where T : IComparable  {  T temp = l[a];  l[a] = l[b];  l[b] = temp;  } |

See Also

Applies to: [BubbleSort](#topic_0000000000000004)

CombSort Class

A class contatining the comb sort algorithm and supporting methods.
Algorithm adapted from: https://www.csharpstar.com/comb-sort-program-csharp/

[System.Object](https://docs.microsoft.com/en-us/dotnet/api/system.object)

**P3\_Andrew.Sorting\_Algorithms.CombSort**

|  |  |
| --- | --- |
| C# |  |
| public static class CombSort | |

Requirements

**Namespace:**[P3\_Andrew.Sorting\_Algorithms](#topic_0000000000000003)

**Assembly:** P3-Andrew (in P3-Andrew.exe)

Methods

[Equals](https://docs.microsoft.com/en-us/dotnet/api/system.object.equals) (inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object)), [Finalize](https://docs.microsoft.com/en-us/dotnet/api/system.object.finalize) (inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object)), [GetHashCode](https://docs.microsoft.com/en-us/dotnet/api/system.object.gethashcode) (inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object)), [GetType](https://docs.microsoft.com/en-us/dotnet/api/system.object.gettype) (inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object)), [MemberwiseClone](https://docs.microsoft.com/en-us/dotnet/api/system.object.memberwiseclone) (inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object)), [Sort<T>](#topic_0000000000000008), [Swap<T>](#topic_0000000000000009), [ToString](https://docs.microsoft.com/en-us/dotnet/api/system.object.tostring) (inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object))

CombSort.Sort<T> Method

The method that calls the comb sort

|  |  |
| --- | --- |
| C# |  |
| public static [void](https://docs.microsoft.com/en-us/dotnet/api/system.void) Sort<T>(  [List](https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.list-1)<T> *l* )  where T : [IComparable](https://docs.microsoft.com/en-us/dotnet/api/system.icomparable) | |

Type Parameters

T

|  |
| --- |
| Any type derived from IComparable |

Parameters

l

|  |
| --- |
| A list of type T |

Source code

|  |
| --- |
| public static void Sort<T>(List<T> l) where T : IComparable  {  double gap = l.Count;  bool swaps = true;    while(gap > 1 || swaps)  {  gap /= 1.247330950103979;    if (gap < 1)  gap = 1;    int i = 0;  swaps = false;    while(i + gap < l.Count)  {  int igap = i + (int)gap;    if(l[i].CompareTo(l[igap]) > 0)  {  Swap(l, i, igap);  swaps = true;  }    i++;  }  }  } |

See Also

Applies to: [CombSort](#topic_0000000000000007)

CombSort.Swap<T> Method

Supporting method to swap to values in a list

|  |  |
| --- | --- |
| C# |  |
| private static [void](https://docs.microsoft.com/en-us/dotnet/api/system.void) Swap<T>(  [List](https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.list-1)<T> *l*,  [int](https://docs.microsoft.com/en-us/dotnet/api/system.int32) *a*,  [int](https://docs.microsoft.com/en-us/dotnet/api/system.int32) *b* )  where T : [IComparable](https://docs.microsoft.com/en-us/dotnet/api/system.icomparable) | |

Type Parameters

T

|  |
| --- |
| Any type derived from IComparable |

Parameters

l

|  |
| --- |
| List with values that need to be swapped |

a

|  |
| --- |
| First index to be swapped |

b

|  |
| --- |
| Second index to be swapped |

Source code

|  |
| --- |
| private static void Swap<T>(List<T> l, int a, int b) where T : IComparable  {  T temp = l[a];  l[a] = l[b];  l[b] = temp;  } |

See Also

Applies to: [CombSort](#topic_0000000000000007)

HeapSort Class

A class contatining the heap sort algorithm and supporting methods.
Algorithm adapted from: https://www.geeksforgeeks.org/heap-sort/

[System.Object](https://docs.microsoft.com/en-us/dotnet/api/system.object)

**P3\_Andrew.Sorting\_Algorithms.HeapSort**

|  |  |
| --- | --- |
| C# |  |
| public static class HeapSort | |

Requirements

**Namespace:**[P3\_Andrew.Sorting\_Algorithms](#topic_0000000000000003)

**Assembly:** P3-Andrew (in P3-Andrew.exe)

Methods

[Equals](https://docs.microsoft.com/en-us/dotnet/api/system.object.equals) (inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object)), [Finalize](https://docs.microsoft.com/en-us/dotnet/api/system.object.finalize) (inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object)), [GetHashCode](https://docs.microsoft.com/en-us/dotnet/api/system.object.gethashcode) (inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object)), [GetType](https://docs.microsoft.com/en-us/dotnet/api/system.object.gettype) (inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object)), [Heapify<T>](#topic_000000000000000C), [MemberwiseClone](https://docs.microsoft.com/en-us/dotnet/api/system.object.memberwiseclone) (inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object)), [Sort<T>](#topic_000000000000000B), [Swap<T>](#topic_000000000000000D), [ToString](https://docs.microsoft.com/en-us/dotnet/api/system.object.tostring) (inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object))

HeapSort.Heapify<T> Method

Arranges list into heap sorted order

|  |  |
| --- | --- |
| C# |  |
| private static [void](https://docs.microsoft.com/en-us/dotnet/api/system.void) Heapify<T>(  [List](https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.list-1)<T> *l*,  [int](https://docs.microsoft.com/en-us/dotnet/api/system.int32) *i* )  where T : [IComparable](https://docs.microsoft.com/en-us/dotnet/api/system.icomparable) | |

Type Parameters

T

|  |
| --- |
| Any type derived from IComparable |

Parameters

l

|  |
| --- |
| A list of type T |

i

|  |
| --- |
| The index to "heapify" |

Source code

|  |
| --- |
| private static void Heapify<T>(List<T> l, int i) where T : IComparable  {  int largest = i;  int left = 2 \* i + 1;  int right = 2 \* i + 2;    if (left < l.Count && l[left].CompareTo(l[largest]) < 0)  largest = left;    if (right < l.Count && l[right].CompareTo(l[largest]) < 0)  largest = right;    if (largest != i)  {  Swap(l, i, largest);  Heapify(l, largest);      }  } |

See Also

Applies to: [HeapSort](#topic_000000000000000A)

HeapSort.Sort<T> Method

The method that calls the Heap sort

|  |  |
| --- | --- |
| C# |  |
| public static [void](https://docs.microsoft.com/en-us/dotnet/api/system.void) Sort<T>(  [List](https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.list-1)<T> *l* )  where T : [IComparable](https://docs.microsoft.com/en-us/dotnet/api/system.icomparable) | |

Type Parameters

T

|  |
| --- |
| Any type derived from IComparable |

Parameters

l

|  |
| --- |
| A list of type T |

Source code

|  |
| --- |
| public static void Sort<T>(List<T> l) where T : IComparable  {  for (int i = l.Count / 2 - 1; i >= 0; i--)  Heapify(l, i);    for (int i = l.Count - 1; i >= 0; i--)  {  Swap(l, 0, i);  Heapify(l, i);  }  } |

See Also

Applies to: [HeapSort](#topic_000000000000000A)

HeapSort.Swap<T> Method

Supporting method to swap to values in a list

|  |  |
| --- | --- |
| C# |  |
| private static [void](https://docs.microsoft.com/en-us/dotnet/api/system.void) Swap<T>(  [List](https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.list-1)<T> *l*,  [int](https://docs.microsoft.com/en-us/dotnet/api/system.int32) *a*,  [int](https://docs.microsoft.com/en-us/dotnet/api/system.int32) *b* )  where T : [IComparable](https://docs.microsoft.com/en-us/dotnet/api/system.icomparable) | |

Type Parameters

T

|  |
| --- |
| Any type derived from IComparable |

Parameters

l

|  |
| --- |
| List with values that need to be swapped |

a

|  |
| --- |
| First index to be swapped |

b

|  |
| --- |
| Second index to be swapped |

Source code

|  |
| --- |
| private static void Swap<T>(List<T> l, int a, int b) where T : IComparable  {  T temp = l[a];  l[a] = l[b];  l[b] = temp;  } |

See Also

Applies to: [HeapSort](#topic_000000000000000A)

InsertionSort Class

A class contatining the insertion sort algorithm and supporting methods.
Algorithm adapted from: https://www.geeksforgeeks.org/insertion-sort/

[System.Object](https://docs.microsoft.com/en-us/dotnet/api/system.object)

**P3\_Andrew.Sorting\_Algorithms.InsertionSort**

|  |  |
| --- | --- |
| C# |  |
| public static class InsertionSort | |

Requirements

**Namespace:**[P3\_Andrew.Sorting\_Algorithms](#topic_0000000000000003)

**Assembly:** P3-Andrew (in P3-Andrew.exe)

Methods

[Equals](https://docs.microsoft.com/en-us/dotnet/api/system.object.equals) (inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object)), [Finalize](https://docs.microsoft.com/en-us/dotnet/api/system.object.finalize) (inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object)), [GetHashCode](https://docs.microsoft.com/en-us/dotnet/api/system.object.gethashcode) (inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object)), [GetType](https://docs.microsoft.com/en-us/dotnet/api/system.object.gettype) (inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object)), [MemberwiseClone](https://docs.microsoft.com/en-us/dotnet/api/system.object.memberwiseclone) (inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object)), [Sort<T>](#topic_000000000000000F), [ToString](https://docs.microsoft.com/en-us/dotnet/api/system.object.tostring) (inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object))

InsertionSort.Sort<T> Method

The method that calls the insertion sort

|  |  |
| --- | --- |
| C# |  |
| public static [void](https://docs.microsoft.com/en-us/dotnet/api/system.void) Sort<T>(  [List](https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.list-1)<T> *l* )  where T : [IComparable](https://docs.microsoft.com/en-us/dotnet/api/system.icomparable) | |

Type Parameters

T

|  |
| --- |
| Any type derived from IComparable |

Parameters

l

|  |
| --- |
| A list of type T |

Source code

|  |
| --- |
| public static void Sort<T>(List<T> l) where T : IComparable  {  for(int i = 1; i < l.Count; i++)  {  T key = l[i];    int j;    for(j = i - 1; j >= 0 && l[j].CompareTo(key) > 0; j--)  {  l[j + 1] = l[j];  }    l[j + 1] = key;  }  } |

See Also

Applies to: [InsertionSort](#topic_000000000000000E)

MergeSort Class

A class contatining the merge sort algorithm and supporting methods.
Algorithm adapted from: https://www.w3resource.com/csharp-exercises/searching-and-sorting-algorithm/searching-and-sorting-algorithm-exercise-7.php

[System.Object](https://docs.microsoft.com/en-us/dotnet/api/system.object)

**P3\_Andrew.Sorting\_Algorithms.MergeSort**

|  |  |
| --- | --- |
| C# |  |
| public static class MergeSort | |

Requirements

**Namespace:**[P3\_Andrew.Sorting\_Algorithms](#topic_0000000000000003)

**Assembly:** P3-Andrew (in P3-Andrew.exe)

Methods

[Equals](https://docs.microsoft.com/en-us/dotnet/api/system.object.equals) (inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object)), [Finalize](https://docs.microsoft.com/en-us/dotnet/api/system.object.finalize) (inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object)), [GetHashCode](https://docs.microsoft.com/en-us/dotnet/api/system.object.gethashcode) (inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object)), [GetType](https://docs.microsoft.com/en-us/dotnet/api/system.object.gettype) (inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object)), [MemberwiseClone](https://docs.microsoft.com/en-us/dotnet/api/system.object.memberwiseclone) (inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object)), [Merge<T>](#topic_0000000000000012), [Sort<T>](#topic_0000000000000011), [ToString](https://docs.microsoft.com/en-us/dotnet/api/system.object.tostring) (inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object))

MergeSort.Merge<T> Method

Combines the two lists, ensuring they are in descending order

|  |  |
| --- | --- |
| C# |  |
| private static [List](https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.list-1)<T> Merge<T>(  [List](https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.list-1)<T> *left*,  [List](https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.list-1)<T> *right* )  where T : [IComparable](https://docs.microsoft.com/en-us/dotnet/api/system.icomparable) | |

Type Parameters

T

|  |
| --- |
| Any type derived from IComparable |

Parameters

left

|  |
| --- |
| A list of type T |

right

|  |
| --- |
| A list of type T |

Returns

A merged list

Source code

|  |
| --- |
| private static List<T> Merge<T>(List<T> left, List<T> right) where T : IComparable  {  List<T> toReturn = new List<T>();    while (left.Count > 0 || right.Count > 0)  {  if (left.Count > 0 && right.Count > 0)  {  if (left.First().CompareTo(right.First()) <= 0) //Comparing First two elements to see which is smaller  {  toReturn.Add(left.First());  left.Remove(left.First()); //Rest of the list minus the first element  }  else  {  toReturn.Add(right.First());  right.Remove(right.First());  }  }  else if (left.Count > 0)  {  toReturn.Add(left.First());  left.Remove(left.First());  }  else if (right.Count > 0)  {  toReturn.Add(right.First());  right.Remove(right.First());  }  }  return toReturn;    } |

See Also

Applies to: [MergeSort](#topic_0000000000000010)

MergeSort.Sort<T> Method

The method that calls the merge sort

|  |  |
| --- | --- |
| C# |  |
| public static [List](https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.list-1)<T> Sort<T>(  [List](https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.list-1)<T> *l* )  where T : [IComparable](https://docs.microsoft.com/en-us/dotnet/api/system.icomparable) | |

Type Parameters

T

|  |
| --- |
| Any type derived from IComparable |

Parameters

l

|  |
| --- |
| A list of type T |

Source code

|  |
| --- |
| public static List<T> Sort<T>(List<T> l) where T : IComparable  {  if (l.Count <= 1)  return l;    List<T> left = new List<T>();  List<T> right = new List<T>();    int middle = l.Count() / 2;    for(int i = 0; i < middle; i++)  {  left.Add(l[i]);  }  for(int i = middle; i < l.Count; i++)  {  right.Add(l[i]);  }      left = Sort(left);  right = Sort(right);  return Merge(left, right);            } |

See Also

Applies to: [MergeSort](#topic_0000000000000010)

QuickSort Class

A class contatining the quick sort algorithm and supporting methods.
Algorithm adapted from: https://www.w3resource.com/csharp-exercises/searching-and-sorting-algorithm/searching-and-sorting-algorithm-exercise-9.php

[System.Object](https://docs.microsoft.com/en-us/dotnet/api/system.object)

**P3\_Andrew.Sorting\_Algorithms.QuickSort**

|  |  |
| --- | --- |
| C# |  |
| public static class QuickSort | |

Requirements

**Namespace:**[P3\_Andrew.Sorting\_Algorithms](#topic_0000000000000003)

**Assembly:** P3-Andrew (in P3-Andrew.exe)

Methods

[Equals](https://docs.microsoft.com/en-us/dotnet/api/system.object.equals) (inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object)), [Finalize](https://docs.microsoft.com/en-us/dotnet/api/system.object.finalize) (inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object)), [GetHashCode](https://docs.microsoft.com/en-us/dotnet/api/system.object.gethashcode) (inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object)), [GetType](https://docs.microsoft.com/en-us/dotnet/api/system.object.gettype) (inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object)), [MemberwiseClone](https://docs.microsoft.com/en-us/dotnet/api/system.object.memberwiseclone) (inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object)), [Partition<T>](#topic_0000000000000015), [Sort<T>](#topic_0000000000000014), [Swap<T>](#topic_0000000000000016), [ToString](https://docs.microsoft.com/en-us/dotnet/api/system.object.tostring) (inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object))

QuickSort.Partition<T> Method

Finds a point to split the list

|  |  |
| --- | --- |
| C# |  |
| private static [int](https://docs.microsoft.com/en-us/dotnet/api/system.int32) Partition<T>(  [List](https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.list-1)<T> *l*,  [int](https://docs.microsoft.com/en-us/dotnet/api/system.int32) *left*,  [int](https://docs.microsoft.com/en-us/dotnet/api/system.int32) *right* )  where T : [IComparable](https://docs.microsoft.com/en-us/dotnet/api/system.icomparable) | |

Type Parameters

T

|  |
| --- |
| Any type derived from IComparable |

Parameters

l

|  |
| --- |
| A list of type T |

left

|  |
| --- |
| Left most point to compare |

right

|  |
| --- |
| Right most point to compare |

Returns

A point to split the list

Source code

|  |
| --- |
| private static int Partition<T>(List<T> l, int left, int right) where T : IComparable  {  T pivot = l[left];  while (true)  {    while (l[left].CompareTo(pivot) < 0 )  {  left++;  }    while (l[right].CompareTo(pivot) > 0)  {  right--;  }    if (left < right)  {  if (l[left].CompareTo(l[right]) == 0) return right;    Swap(l, left, right);  }  else  {  return right;  }  }  } |

See Also

Applies to: [QuickSort](#topic_0000000000000013)

QuickSort.Sort<T> Method

The method that calls the quick sort

|  |  |
| --- | --- |
| C# |  |
| public static [void](https://docs.microsoft.com/en-us/dotnet/api/system.void) Sort<T>(  [List](https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.list-1)<T> *l*,  [int](https://docs.microsoft.com/en-us/dotnet/api/system.int32) *left* = default,  [int](https://docs.microsoft.com/en-us/dotnet/api/system.int32) *right* = default )  where T : [IComparable](https://docs.microsoft.com/en-us/dotnet/api/system.icomparable) | |

Type Parameters

T

|  |
| --- |
| Any type derived from IComparable |

Parameters

l

|  |
| --- |
| A list of type T |

left

|  |
| --- |
|  |

right

|  |
| --- |
|  |

Source code

|  |
| --- |
| public static void Sort<T>(List<T> l, int left = default, int right = default) where T : IComparable  {  if (left == default)  left = 0;  if (right == default)  right = l.Count - 1;    int pivot = Partition(l, left, right);      if (pivot > 1)  Sort(l, left, pivot - 1);  if (pivot + 1 < right)  {  Sort(l, pivot + 1, right);  }    } |

See Also

Applies to: [QuickSort](#topic_0000000000000013)

QuickSort.Swap<T> Method

Supporting method to swap to values in a list

|  |  |
| --- | --- |
| C# |  |
| private static [void](https://docs.microsoft.com/en-us/dotnet/api/system.void) Swap<T>(  [List](https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.list-1)<T> *l*,  [int](https://docs.microsoft.com/en-us/dotnet/api/system.int32) *a*,  [int](https://docs.microsoft.com/en-us/dotnet/api/system.int32) *b* )  where T : [IComparable](https://docs.microsoft.com/en-us/dotnet/api/system.icomparable) | |

Type Parameters

T

|  |
| --- |
| Any type derived from IComparable |

Parameters

l

|  |
| --- |
| List with values that need to be swapped |

a

|  |
| --- |
| First index to be swapped |

b

|  |
| --- |
| Second index to be swapped |

Source code

|  |
| --- |
| private static void Swap<T>(List<T> l, int a, int b) where T : IComparable  {  T temp = l[a];  l[a] = l[b];  l[b] = temp;  } |

See Also

Applies to: [QuickSort](#topic_0000000000000013)

SelectionSort Class

A class contatining the selection sort algorithm and supporting methods.
Algorithm adapted from: https://www.geeksforgeeks.org/selection-sort/

[System.Object](https://docs.microsoft.com/en-us/dotnet/api/system.object)

**P3\_Andrew.Sorting\_Algorithms.SelectionSort**

|  |  |
| --- | --- |
| C# |  |
| public static class SelectionSort | |

Requirements

**Namespace:**[P3\_Andrew.Sorting\_Algorithms](#topic_0000000000000003)

**Assembly:** P3-Andrew (in P3-Andrew.exe)

Methods

[Equals](https://docs.microsoft.com/en-us/dotnet/api/system.object.equals) (inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object)), [Finalize](https://docs.microsoft.com/en-us/dotnet/api/system.object.finalize) (inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object)), [GetHashCode](https://docs.microsoft.com/en-us/dotnet/api/system.object.gethashcode) (inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object)), [GetType](https://docs.microsoft.com/en-us/dotnet/api/system.object.gettype) (inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object)), [MemberwiseClone](https://docs.microsoft.com/en-us/dotnet/api/system.object.memberwiseclone) (inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object)), [Sort<T>](#topic_0000000000000018), [Swap<T>](#topic_0000000000000019), [ToString](https://docs.microsoft.com/en-us/dotnet/api/system.object.tostring) (inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object))

SelectionSort.Sort<T> Method

The method that calls the selection sort

|  |  |
| --- | --- |
| C# |  |
| public static [void](https://docs.microsoft.com/en-us/dotnet/api/system.void) Sort<T>(  [List](https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.list-1)<T> *l* )  where T : [IComparable](https://docs.microsoft.com/en-us/dotnet/api/system.icomparable) | |

Type Parameters

T

|  |
| --- |
| Any type derived from IComparable |

Parameters

l

|  |
| --- |
| A list of type T |

Source code

|  |
| --- |
| public static void Sort<T>(List<T> l) where T : IComparable  {  int min\_index;    for(int i = 0; i < l.Count - 1; i++)  {  min\_index = i;    for(int j = i + 1; j < l.Count; j++)  {    if(l[j].CompareTo(l[min\_index]) < 0)  {  min\_index = j;  }  }    Swap(l, min\_index, i);  }  } |

See Also

Applies to: [SelectionSort](#topic_0000000000000017)

SelectionSort.Swap<T> Method

Supporting method to swap to values in a list

|  |  |
| --- | --- |
| C# |  |
| private static [void](https://docs.microsoft.com/en-us/dotnet/api/system.void) Swap<T>(  [List](https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.list-1)<T> *l*,  [int](https://docs.microsoft.com/en-us/dotnet/api/system.int32) *a*,  [int](https://docs.microsoft.com/en-us/dotnet/api/system.int32) *b* )  where T : [IComparable](https://docs.microsoft.com/en-us/dotnet/api/system.icomparable) | |

Type Parameters

T

|  |
| --- |
| Any type derived from IComparable |

Parameters

l

|  |
| --- |
| List with values that need to be swapped |

a

|  |
| --- |
| First index to be swapped |

b

|  |
| --- |
| Second index to be swapped |

Source code

|  |
| --- |
| private static void Swap<T>(List<T> l, int a, int b) where T : IComparable  {  T temp = l[a];  l[a] = l[b];  l[b] = temp;  } |

See Also

Applies to: [SelectionSort](#topic_0000000000000017)

ShellSort Class

A class contatining the shell sort algorithm and supporting methods.
Algorithm adapted from: https://www.w3resource.com/csharp-exercises/searching-and-sorting-algorithm/searching-and-sorting-algorithm-exercise-1.php

[System.Object](https://docs.microsoft.com/en-us/dotnet/api/system.object)

**P3\_Andrew.Sorting\_Algorithms.ShellSort**

|  |  |
| --- | --- |
| C# |  |
| public static class ShellSort | |

Requirements

**Namespace:**[P3\_Andrew.Sorting\_Algorithms](#topic_0000000000000003)

**Assembly:** P3-Andrew (in P3-Andrew.exe)

Methods

[Equals](https://docs.microsoft.com/en-us/dotnet/api/system.object.equals) (inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object)), [Finalize](https://docs.microsoft.com/en-us/dotnet/api/system.object.finalize) (inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object)), [GetHashCode](https://docs.microsoft.com/en-us/dotnet/api/system.object.gethashcode) (inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object)), [GetType](https://docs.microsoft.com/en-us/dotnet/api/system.object.gettype) (inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object)), [MemberwiseClone](https://docs.microsoft.com/en-us/dotnet/api/system.object.memberwiseclone) (inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object)), [Sort<T>](#topic_000000000000001B), [ToString](https://docs.microsoft.com/en-us/dotnet/api/system.object.tostring) (inherited from [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object))

ShellSort.Sort<T> Method

The method that calls the shell sort

|  |  |
| --- | --- |
| C# |  |
| public static [void](https://docs.microsoft.com/en-us/dotnet/api/system.void) Sort<T>(  [List](https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.list-1)<T> *l* )  where T : [IComparable](https://docs.microsoft.com/en-us/dotnet/api/system.icomparable) | |

Type Parameters

T

|  |
| --- |
| Any type derived from IComparable |

Parameters

l

|  |
| --- |
| A list of type T |

Source code

|  |
| --- |
| public static void Sort<T>(List<T> l) where T : IComparable  {  int inc = 3;  int j;  T temp;    while(inc > 0)  {  for(int i = 0; i < l.Count; i++)  {  j = i;  temp = l[i];    while((j >= inc) && (l[j - inc].CompareTo(temp) > 0))  {  l[j] = l[j - inc];  j = j - inc;  }    l[j] = temp;  }    if (inc / 2 != 0)  inc = inc / 2;  else if (inc == 1)  inc = 0;  else  inc = 1;  }  } |

See Also

Applies to: [ShellSort](#topic_000000000000001A)

# Index

[AVERAGEVALUES Field](#topic_000000000000001F)

[AlgorithmTesting Namespace](#topic_000000000000001C)

[Average Method](#topic_0000000000000022)

[BubbleSort Class](#topic_0000000000000004)

[BubbleSort Method](#topic_0000000000000023)

[CombSort Class](#topic_0000000000000007)

[CombSort Method](#topic_0000000000000024)

[DATASIZE Field](#topic_000000000000001E)

[HeapSort Class](#topic_000000000000000A)

[HeapSort Method](#topic_0000000000000025)

[Heapify<T> Method](#topic_000000000000000C)

[InsertionSort Class](#topic_000000000000000E)

[InsertionSort Method](#topic_0000000000000026)

[Main Method](#topic_0000000000000002)

[Merge<T> Method](#topic_0000000000000012)

[MergeSort Class](#topic_0000000000000010)

[MergeSort Method](#topic_0000000000000027)

[P3-Andrew Reference](#topic_000000000000002B)

[P3\_Andrew Namespace](#topic_0000000000000000)

[P3\_Andrew.Sorting\_Algorithms Namespace](#topic_0000000000000003)

[Partition<T> Method](#topic_0000000000000015)

[PrepTest Method](#topic_0000000000000021)

[Program Class](#topic_0000000000000001)

[QuickSort Class](#topic_0000000000000013)

[QuickSort Method](#topic_0000000000000028)

[SelectionSort Class](#topic_0000000000000017)

[SelectionSort Method](#topic_0000000000000029)

[ShellSort Class](#topic_000000000000001A)

[ShellSort Method](#topic_000000000000002A)

[Sort<T> Method {P3\_Andrew.Sorting\_Algorithms.BubbleSort}](#topic_0000000000000005)

[Sort<T> Method {P3\_Andrew.Sorting\_Algorithms.CombSort}](#topic_0000000000000008)

[Sort<T> Method {P3\_Andrew.Sorting\_Algorithms.HeapSort}](#topic_000000000000000B)

[Sort<T> Method {P3\_Andrew.Sorting\_Algorithms.InsertionSort}](#topic_000000000000000F)

[Sort<T> Method {P3\_Andrew.Sorting\_Algorithms.MergeSort}](#topic_0000000000000011)

[Sort<T> Method {P3\_Andrew.Sorting\_Algorithms.QuickSort}](#topic_0000000000000014)

[Sort<T> Method {P3\_Andrew.Sorting\_Algorithms.SelectionSort}](#topic_0000000000000018)

[Sort<T> Method {P3\_Andrew.Sorting\_Algorithms.ShellSort}](#topic_000000000000001B)

[Swap<T> Method {P3\_Andrew.Sorting\_Algorithms.BubbleSort}](#topic_0000000000000006)

[Swap<T> Method {P3\_Andrew.Sorting\_Algorithms.CombSort}](#topic_0000000000000009)

[Swap<T> Method {P3\_Andrew.Sorting\_Algorithms.HeapSort}](#topic_000000000000000D)

[Swap<T> Method {P3\_Andrew.Sorting\_Algorithms.QuickSort}](#topic_0000000000000016)

[Swap<T> Method {P3\_Andrew.Sorting\_Algorithms.SelectionSort}](#topic_0000000000000019)

[UnitTest1 Class](#topic_000000000000001D)

[testList Field](#topic_0000000000000020)