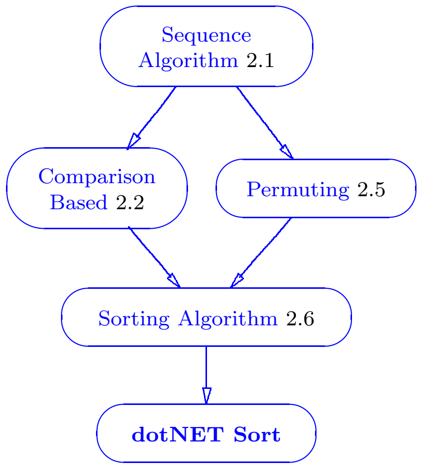
2.6.1 dotnetsort

Section authors: Alan Damon, Charlie Mathis, John Haggerty



Refinement of: Sequence Sorting Algorithm (§2.6), therefore of Comparison Based (§2.2), Permuting (§2.5), Sequence Algorithm (§2.1).

Prototype:
```
template<class RandomAccessIterator>
void sort(RandomAccessIterator first,
          RandomAccessIterator last)
```

Input/Output: The input/output of the sort algorithm is defined in the more abstract level of Sequence Sorting Algorithm (§2.6).

Effects: Standard effects of a Sequence Sorting Algorithm (§2.6). In brief: the elements in [first, last) after execution are a permutation of the original elements in the range, and they are in nondecreasing order according the comparison operator.

Asymptotic complexity: Let $N = \text{last} - \text{first}$.

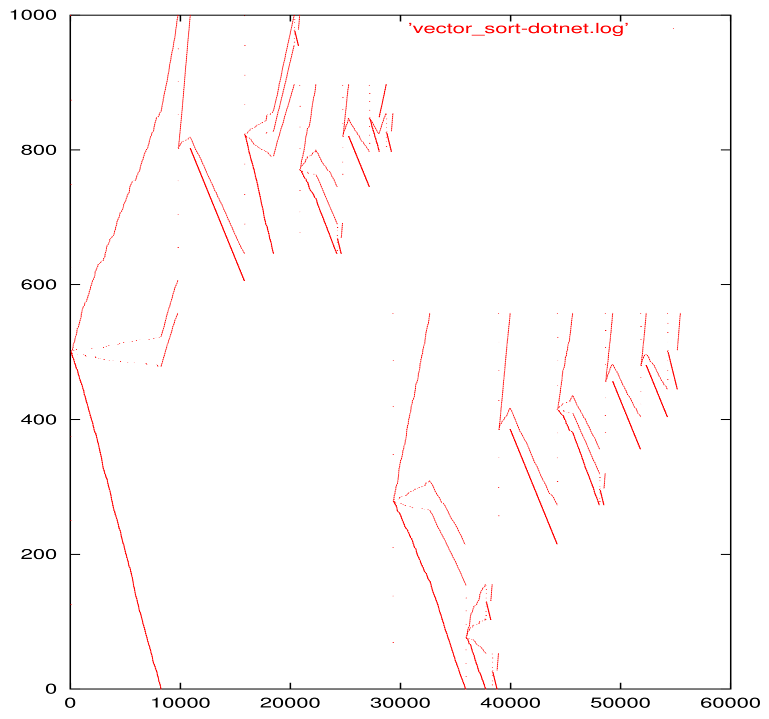
- Average case (random data): $O(N \log N)$

- Worst case: $O(N \log N)$

Operation Counts: (x1000)

Size	Version	Assign	Comp	Total
4.096	6.0	30	58	89
	.NET	52	74	127
8.192	6.0	65	123	189
	.NET	116	164	281
16.384	6.0	136	274	410
	.NET	261	362	624
32.768	6.0	291	580	871
	.NET	502	754	1257
49.152	6.0	445	902	1327
	.NET	796	1171	1967
65.536	6.0	600	1232	1832
	.NET	1135	1621	2757

.NET Sort iterator trace plot: (Vector; Size: 1000)



Algorithm Timing Summary: (.NET to 6.0)

Random input:	170% slower
Descending input:	215% slower
Ascending input:	30% faster

Input Times: (measured in seconds)

Size	Version	Random	Ascen	Desc
4096	.NET	0.007343	0.000781	0.000156
	6.0	0.004375	0.001250	0.005625
32756	.NET	0.063218	0.005484	0.001093
	6.0	0.362969	0.111094	0.050078
65536	.NET	0.132219	0.011890	0.001875
	6.0	0.955938	0.018765	0.109063

Whats up with .NET: There are a few differences between Visual Studio .NET and Visual Studio 6.0. The sort implementation was completely re-written durring the different versions which gives it the different characteristics. One of the bonus to the .NET version is that it is a lot easier to read than the 6.0 implementation.

Summary: (.NET vs. 6.0)

- Maximum Insertion Sort is 32 (vs. 16)
  1. Slight degradation to descending input
  2. Improvement on ascending input
- Uses a Median of 9 (vs. Median of 3)
  1. Improvement over Median of 3
- Partition Implementation is more complex
  1. 54 lines of code (vs. 10 lines of code)
  2. Most degradation