***Sorting algorithms and their time complexity in C++***

# Presentation

* Standard in C++ (std::sort() known as IntroSort): O(nlogn);
* RadixSort (with base 10 or 2): O((n+b) \* logb(k)), b=base, k=nrmax;
* QuickSort: [Engineering a Sort Function](http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.14.8162&rep=rep1&type=pdf)-> 1.386 n log n average comparisons using a randomized pivot and 1.188 n log n average comparisons using a median of 3 pivot. It has O(n^2) worst-case runtime (when the pivot is fixed and the numbers are already sorted);
* BubbleSort: O(n^2) and O(n) for already sorted numbers;
* MergeSort: O(nlogn) but is slower than QuickSort in general terms.
* ShellSort (with gap reduced by half in every iteration): O(n^2)

and ShellSort2 (with gap calculated with the formula 2\*[N/22k+1]+1): O(N3/2)

***Top fastest algorithms for nrmax=10^6 and:***

|  |  |  |
| --- | --- | --- |
| N=10^3 | | N=10^5 |
| Random numbers | | |
| 1. RadixSort,QuickSort, Standard, ShellSort 2. MergeSort 3. BubbleSort | | 1. Standard 2. QuickSortMedian3 3. ShellSort 4. ShellSort2, QucikSortRandom 5. RadixSort10 6. RadixSort2 7. MergeSort 8. BubbleSort |
| Sorted numbers | | |
| 1. RadixSort10 ,QuickSort, Standard, ShellSort,Bubble 2. RadixSort2 3. MergeSort | 1. BubbleSort 2. ShellSorts 3. Standard 4. Radix10,QuickSortMedian3 5. QuickSortRandom 6. RadixSort2 7. MergeSort | |

# Charts