

Books Titles Sorting

with Merge and Quick Sorting Algorithms

Members:

5748106 Serhii Bielik

Problem Statement

The problem of the project is to sort large amounts of books titles. Those titles were parsed from various websites. Titles don’t have fixed format. This type of research can be useful in books listing system. Moreover, the sorting speed must be fast enough.

Methodology

To solve this problem, I have chosen two popular sorting algorithms to compare Merge Sort and Quick Sort. Both merge sort and quicksort employ a common algorithmic paradigm based on recursion. This paradigm, divide-and-conquer, breaks a problem into subproblems that are similar to the original problem, recursively solves the subproblems, and finally combines the solutions to the subproblems to solve the original problem. Because divide-and-conquer solves subproblems recursively, each subproblem must be smaller than the original problem, and there must be a base case for subproblems.

So I expect to see a high performance sorting speed with Θ(n log(n))

Results

Merge Sort showed stable result in every case Ω(n log(n)), Θ(n log(n)), O(n log(n))

However, Quick Sort shoved worse result Ω(n^2), Θ(n log(n)), O(n^2)

All graphs based on the program output:

C:\Python27\python.exe D:/YaDisk/Uni/algorithms/proj/packet\_run.py

=== Merge Sort [ 500 ] ===

Best: 0.0033055123212

Average: 0.00414121573769

Worst: 0.00332416299359

=== Quick Sort [ 500 ] ===

Best: 0.0328727043066

Average: 0.00298308562856

Worst: 0.0452659483688

==============================

=== Merge Sort [ 1000 ] ===

Best: 0.00700703206996

Average: 0.00902922483718

Worst: 0.00703104800428

=== Quick Sort [ 1000 ] ===

Best: 0.0882391414763

Average: 0.00966283672128

Worst: 0.166470025559

==============================

=== Merge Sort [ 2500 ] ===

Best: 0.0190591987671

Average: 0.0312161158161

Worst: 0.0193021684856

=== Quick Sort [ 2500 ] ===

Best: 0.628150813936

Average: 0.0182464893303

Worst: 0.587899341555

==============================

=== Merge Sort [ 5000 ] ===

Best: 0.0395018686441

Average: 0.0598609937283

Worst: 0.0417345840696

=== Quick Sort [ 5000 ] ===

Best: 1.5834137271

Average: 0.0406794158916

Worst: 4.17549822844

==============================

=== Merge Sort [ 7000 ] ===

Best: 0.0580917347374

Average: 0.0785512668661

Worst: 0.0587419533846

=== Quick Sort [ 7000 ] ===

Best: 5.15881839559

Average: 0.0615551390523

Worst: 3.1970928967

==============================

=== Merge Sort [ 10000 ] ===

Best: 0.0844200969426

Average: 0.116536299318

Worst: 0.0869080455424

=== Quick Sort [ 10000 ] ===

Best: 8.27800079099

Average: 0.0904958728361

Worst: 8.21051039481

==============================

=== Merge Sort [ 13000 ] ===

Best: 0.111955898569

Average: 0.15473543127

Worst: 0.114788501375

=== Quick Sort [ 13000 ] ===

Best: 10.6285107975

Average: 0.130373565303

Worst: 10.9150426104

==============================

=== Merge Sort [ 15000 ] ===

Best: 0.130191657375

Average: 0.179194638417

Worst: 0.133227884646

=== Quick Sort [ 15000 ] ===

Best: 8.89626725954

Average: 0.147964470725

Worst: 9.91940201346

==============================

=== Merge Sort [ 17000 ] ===

Best: 0.154376725187

Average: 0.212586751176

Worst: 0.150756961811

=== Quick Sort [ 17000 ] ===

Best: 13.2860489394

Average: 0.169674619858

Worst: 13.2096233382

==============================

=== Merge Sort [ 20000 ] ===

Best: 0.178274623742

Average: 0.248051387968

Worst: 0.184811812161

=== Quick Sort [ 20000 ] ===

Best: 21.8897860231

Average: 0.191584816615

Worst: 29.8796914106

==============================

Process finished with exit code 0

Analysis

Merge Sort showed expected result without any surprises. However, the Quick Sort showed Ω(n^2) instead of expected Ω(n log(n)) which was quite interesting. I think, the reason is hiding in quality of data and different sorting technique of Excel which was used for best and worst case preparation. The situation with O(n^2) is normal for the Quick Sort but this kind of situation is rare in real life.

Summary

In conclusion I decide to use Merge Sort as a more stable in every situation; even the speed of Quick Sort is faster in the average cases. The Quick Sort will be better in case of really huge amounts of random data.