Sorting Algorithms Estimations – CMPE 242 HOMEWORK

I coded a tester class that we can choose how many array we want to sort, how many element they could have and which algorithms we want to use for sorting. In that way, we don't have to waste our time and compare easily one algorithm to the other ones. There are four pre-defined structure for arrays' elements. Randomly generated, ascending ordered, descending ordered and a list of same integers. I did it because I know sometimes we can't exactly say what algorithm is running based on only sorting randomly generated arrays. So we can look at descending or ascending and understand which one it is. In my opinion, it is better to run tester class with multiple arrays like ten and program will be print the average of them so we can get more precise data.

1) According the data I collect, we can easily see that algorithm 3 and algorithm 4 are faster than the others. So they should be merge sort and quick sort. Let's see which one is which.
2) Although, they are so close each other for randomly generated inputs we can clarify that algorithm 3 faster than algorithm 4 but we can't get the correct answer still. So let's look at a list which has a full of same integer. This would be worst case of quick-sort, merge sort is faster than quick sort this type of list.
3) That's why **algorithm 3 is Quick Sort**
4) **Algorithm 4 is Merge Sort**
5) Now let's look at algorithm 1, 2, 5. Well, when we look at the graphs we can't understand directly but there is a hint for us. Look at the ascending ordered graphs. Algorithm 1 is worst among others. Although it is already ordered, selection sort would iterate through array for each item to find minimum. That's why its time complexity is $O(n^2)$.
6) We can easily say that **Algorithm 1 is selection sort.**
7) Finally, we have algorithm 2, 5 and algorithm. Insertion sort algorithm finds the smallest one iterating through arrays and puts them into the correct index. For the best case, if the array is already sorted ascending, its time complexity will be just N due to iterating once through the array. For the worst-case which is ordered but descending. Putting the correct element into ith index we need to iterate N-i times. So $N+N-1+N-2\ldots = O(N^2)$
8) As for bubble sort, it will check each pair in the best case and its time complexity will be N. What will be the array is ordered descending. It changes pair elements until the ith element is at the correct place. It will be $O(N^2)$ which is the same with insertion sort but the bubble sort algorithm changes pairs too much so it will take longer time for larger inputs.
9) So when we compare algorithm 2 and 5 results we can understand that **Algorithm 2 is insertion sort** which is faster and **algorithm 5 is bubble sort.**

Mathematical Expressions for algorithm 1,2 and 5

Algo1 → Time ratio = $\frac{26054}{6008}=4.34$ → b = $\log(\frac{260}{4.34})=1.77$ → $a*N^{1.98}$ → $O(n^2)$

Algo2 → Time ratio = $\frac{246359}{34102}=7.22$ → b = $\log(\frac{246}{7.22})=1.53$ → $a*N^{1.53}$ → $O(n^2)$

Algo5 → Time ratio = $\frac{760144}{138607}=5.48$ → b = $\log(\frac{76}{5.48})=2.14$ → $a*N^{2.14}$ → $O(n^2)$

You can find the graphs and tables below.

Uğur Kellecioğlu

CMPE242 01

Randomly Generated Arrays' Sorting Times by Each Algorithm

Size	Algo1 Time(ms)	Algo2 Time(ms)	Algo5 Time(ms)
8000	90	86	373
16000	435.0	338	1631
32000	1443	1821	7100
64000	6008	7319	33358
128000	26054	34102	138607
256000	101844	246359	760144

Size	Algo3 Time(ms)	Algo4 Time(ms)
8000	4	7
16000	18	5
32000	11	9
64000	18	27
128000	39	48
256000	90	164
512000	197	353
1024000	441	770
2048000	974	1688
4096000	2240	3795
8192000	4965	8149
16384000	10869	17743
32768000	23674	38335

Generated in Ascending Order Arrays' Sorting Times by Each Algorithm

Size	Algo1 Time(ms)	Algo2 Time(ms)	Algo5 Time(ms)
8000	100	124	271
16000	365	496	1275
32000	1491	1979	5512
64000	5439	7559	22437
128000	22150	34432	98152
256000	93041	448340	871220

Size	Algo3 Time(ms)	Algo4 Time(ms)
8000	1	8
16000	9	3
32000	1	5
64000	3	12
128000	6	26
256000	25	139
512000	38	287
1024000	86	609
2048000	194	1246
4096000	398	2608
8192000	794	5406
16384000	1678	11316
32768000	3393	23406

Generated in Descending Order Arrays' Sorting Times by Each Algorithm

Size	Algo1 Time(ms)	Algo2 Time(ms)	Algo5 Time(ms)
8000	100	124	271
16000	365	496	1275
32000	1491	1979	5512
64000	5439	7559	22437
128000	22150	34432	98152
256000	93041	448340	871220

Size	Algo3 Time(ms)	Algo4 Time(ms)
8000	1	1
16000	10	3
32000	1	5
64000	3	12
128000	7	26
256000	28	147
512000	58	188
1024000	106	591
2048000	226	1242
4096000	451	2610
8192000	923	5396
16384000	1879	11270
32768000	3942	23357

Linear Graphics of Tables