Homework Assignment I

CS 202-002

2022-23 Spring

Yassin Younis

22101310

CS

a) Show that $f(n) = 6n^4 + 9n^2 - 8$ is $O(n^4)$ by specifying the appropriate c and n_0 values in Big-O definition.

Proof: by the Big-Oh definition, f(n) is $O(n^4)$ if f(n) $\leq$ c·n^4 for some n $\geq n_0$:

$6n^4 + 9n^2 - 8 \leq$ c·n^4, dividing by n^4 we get:

$6 + 9/n^2 - 8/n^4 \leq$ c, therefore the statement above holds for: c=7 and n $\geq n_0$= 1

b) Trace the below mentioned sorting algorithms to sort the array [5, 3, 2, 6, 4, 1, 3, 7] in ascending order. Use the array implementation of the algorithms as described in the textbook and show all major steps (after each sort pass for instance).

i) Selection Sort

[5, 3, 2, 6, 4, 1, 3, 7]

[5, 3, 2, 3, 4, 1, 6, **7**]

[1, 3, 2, 3, 4, 5, **6, 7**]

[1, 3, 2, 3, 4, **5, 6, 7**]

[1, 3, 2, 3, **4, 5, 6, 7**]

[1, 2, 3, **3, 4, 5, 6, 7**]

[1, 2, **3, 3, 4, 5, 6, 7**]

[1, **2, 3, 3, 4, 5, 6, 7**]

ii) Merge Sort

[5, 3, 2, 6, 4, 1, 3, 7]

[5, 3, 2, 6] [4, 1, 3, 7]

[5, 3] [2, 6] [4, 1] [3, 7]

[5] [3] [2] [6] [4] [1] [3] [7]

[3, 5] [2, 6] [1, 4] [3, 7]

[2, 3, 5, 6] [1, 3, 4, 7]

[1, 2, 3, 3, 4, 5, 6, 7]

iii) Quick Sort – Assume the last element is chosen as a pivot.

[5, 3, 2, 6, 4, 1, 3, 7]

[5, 3, 2, 6, 4, 1, 3] [**7**]

[2, 1, 3, 6, 4, 3, 5] [**7**]

[2, 1, 3, 4, 3][5][6] [**7**]

[2, 1, 3][3][4][5][6] [**7**]

[2, 1] [3][3][4][5][6] [**7**]

[1] [2] [3][3][4][5][6] [7]

c) Find the asymptotic running times in big O notation of $T(n) = 2T(n - 1) + n^2$, where $T(1) = 1$ by using the repeated substitution method. Show your steps in detail.

$$T(n) = 2T(n - 1) + n^2$$

$$T(n) = 2(2T(n - 2) + (n - 1)^2) + n^2$$

$$T(n) = 2(2(2T(n - 3) + (n - 2)^2) + (n - 1)^2) + n^2$$

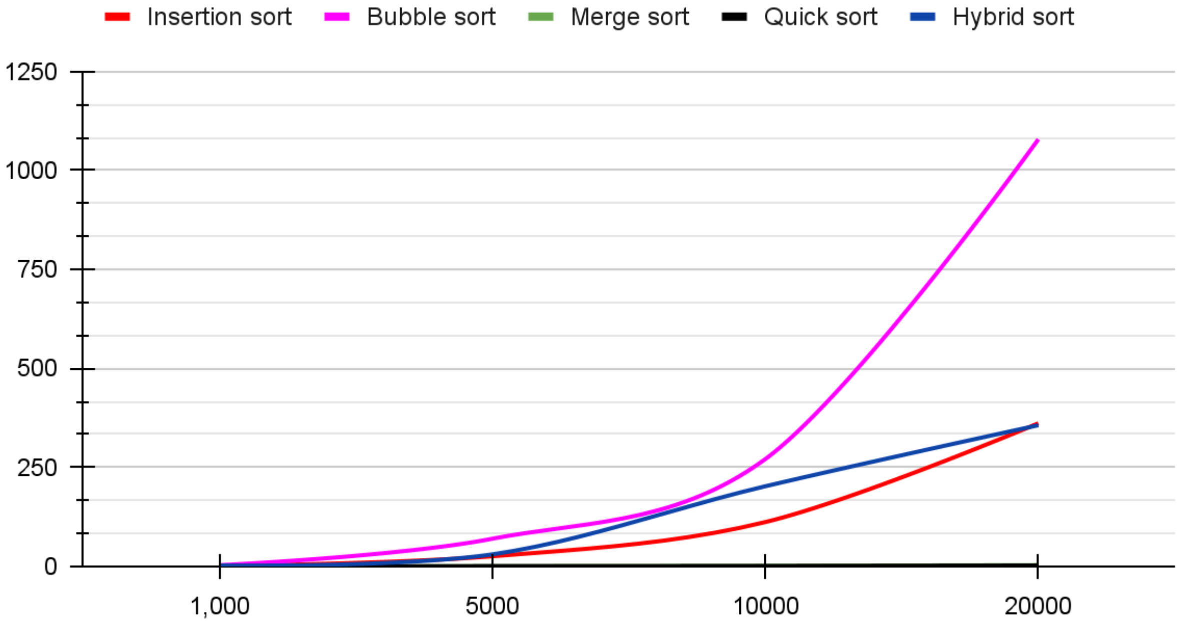
$$T(n) = 2^k T(n - k) + \sum_{0}^{k-1} (n - k)^2 \quad \text{for k=n-1}$$

O(n) for T(n) = 2^n

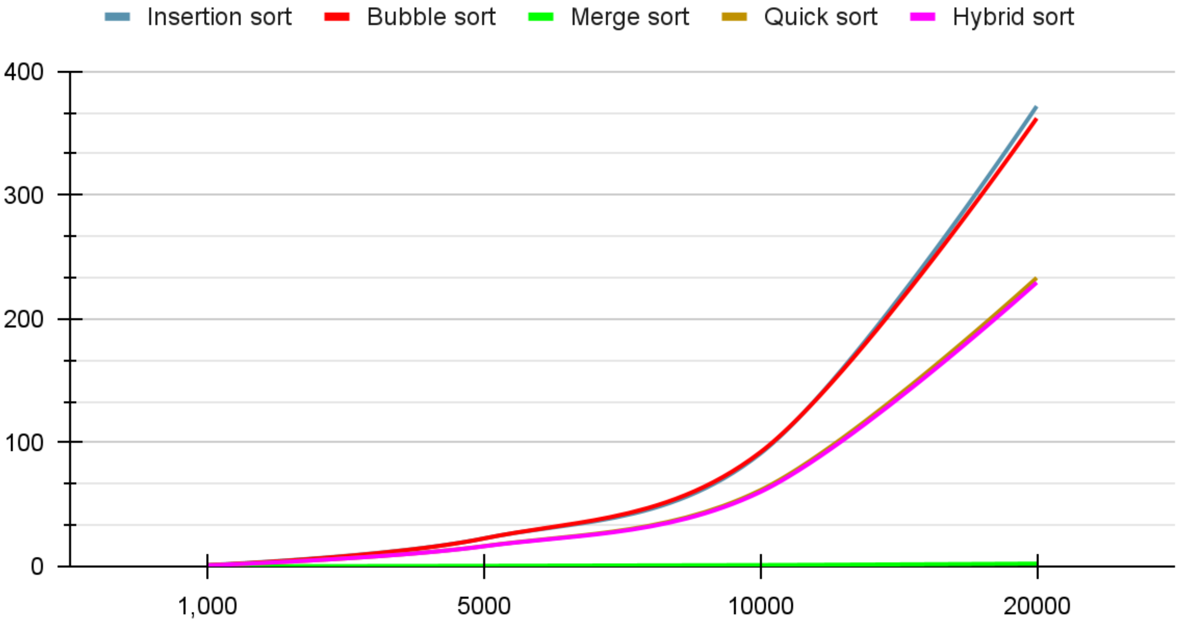
Question III Table

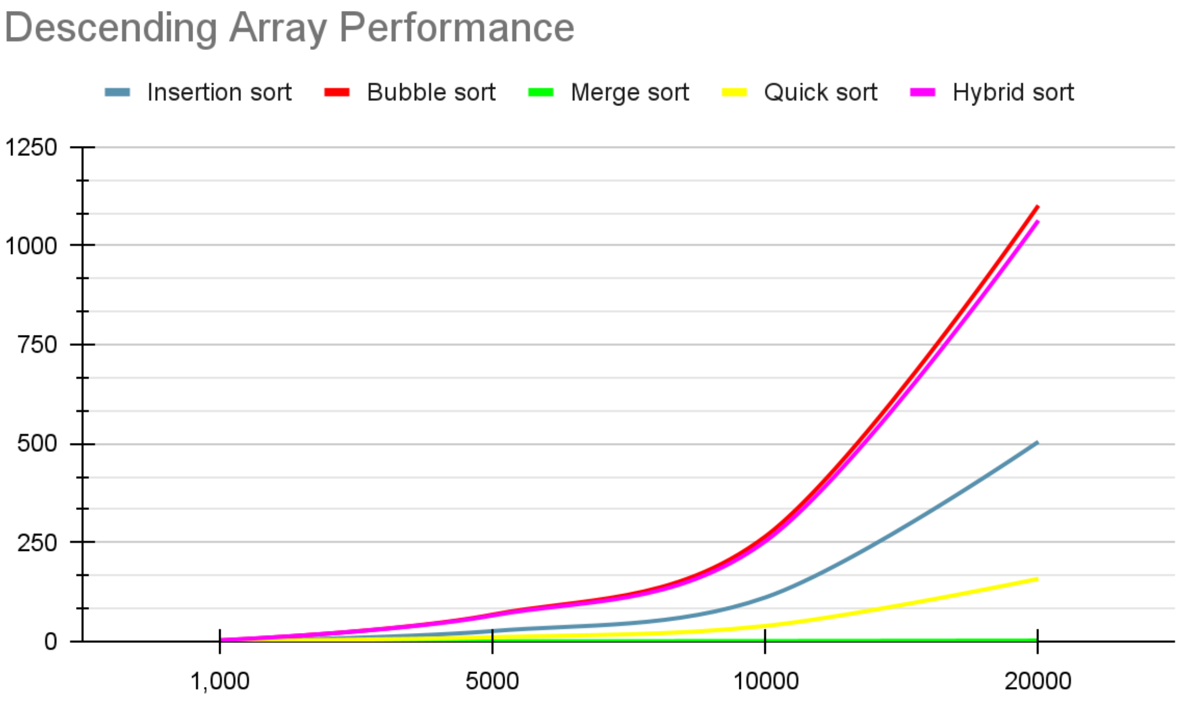
Array	Elapsed Time (ms)					Number of Comparisons					Number of Moves				
	Insertion sort	Bubble sort	Merge sort	Quick sort	Hybrid sort	Insertion sort	Bubble sort	Merge sort	Quick sort	Hybrid sort	Insertion sort	Bubble sort	Merge sort	Quick sort	Hybrid sort
R1K	1.121	3.099	0.107	0.083	2.384	499500	499500	8712	11257	400617	3000	537783	19952	17784	441681
R5K	25.642	69.858	0.724	0.593	30.376	12997000	12997000	63929	78621	5756178	18000	14031942	143568	124311	6262701
R10K	111.902	270.94	1.575	1.097	202.387	62992000	62992000	184463	231595	43311654	48000	68788011	410800	352557	47434005
R20K	361.408	1078.67	3.209	2.389	356.541	262982000	262982000	445467	554210	110006353	108000	289752759	985264	856134	120968010
A1K	1.104	0.903	0.074	0.874	0.825	263481500	263481500	451154	720314	110173595	111000	289758723	1005216	1353093	121467030
A5K	22.606	22.584	0.466	16.381	16.242	275979000	275979000	486104	4057040	113517830	126000	289800024	1128832	11353122	131480325
A10K	91.081	92.187	1.097	61.51	60.126	325974000	325974000	561483	16290696	125768026	156000	289890201	1396064	48031893	168186666
A20K	371.881	362.094	2.261	233.217	229.418	525964000	525964000	723216	62077430	171591006	216000	290086218	1970528	185344302	305558928
D1K	1.274	2.679	0.093	0.835	3.015	526463500	526463500	728966	62214412	172084554	219000	291574803	1990480	185573085	307000221
D5K	26.231	67.657	0.534	10.774	65.518	538961000	538961000	764596	64852329	184542134	234000	328968219	2114096	190255383	344094927
D10K	111.251	265.285	1.101	39.219	253.07	588956000	588956000	841249	74765291	234487169	264000	478593675	2381328	207408516	493301358
D20K	504.631	1101.74	2.248	158.268	1063.93	788946000	788946000	1005163	112341995	434377204	324000	1078200723	2955792	273604776	1092069159

Random Sorted Arrays Performance



Ascending Array Performance





Final Comments:

Looking at the results, we can observe that the sorting algorithms' performance varies significantly depending on the input array type and size. Generally, the Merge sort and Quick sort algorithms perform better than the Insertion sort and Bubble sort algorithms for large input sizes. The Hybrid sort algorithm seems to perform well across all input array types and sizes, which is expected as it is designed to use the best of both Quick sort and Insertion sort algorithms.

For small input sizes, Insertion sort and Bubble sort algorithms perform well on all input array types. However, as the input size increases, the performance of these algorithms deteriorates significantly, especially on partially sorted input arrays.

When it comes to partially sorted input arrays (A and D), Merge sort seems to perform better than Quick sort, especially for larger input sizes. This is because the dividing strategy used in Merge sort works well on partially sorted input arrays.

In conclusion, the choice of sorting algorithm depends on the input array type and size. For small input sizes, Insertion sort and Bubble sort algorithms are suitable. For larger input sizes, Merge sort, Quick sort, or Hybrid sort algorithms can be used depending on the input array type.

Relevant specifications for device used to conduct these tests:

Processor: Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz   2.59 GHz

Installed RAM: 16.0 GB (15.8 GB usable)

Operating System: Windows 11 Pro 21H2 Build 22000.1219

System Type: 64-bit operating system, x64-based processor