Bilkent University

Emrehan Hoşver

CS 202

Homework 1

**Question 1a:**

$f(n) = 8n^4+5n^3+7$

$8n^4+5n^3+7 <= cn^5$ then we need make c stand alone thus we divide each side with $n^5$

$8/n+5/n^2+7/n^5 <= c$ then we need to test this equation with $n_0$ value to check if it holds or not

$8/1+5/1^2+7/1^5 <= 20$, so as a result $f(n) <= O(n^5)$ is true for $n >= 1$ and $c = 20$ .

**Question 1b:**

Selection Sort: Find min element and swap it with first element of the subarray. In each turn shrink subarray from its head. Arr[i,n-1] , i starts form 0 and goes to n-1.

Array for Selection Sort is = [22,8,49,25,18,30,20,15,35,27] and size of the array is 10.

1. 8 is min in array[0,n-1] = [8,22,49,25,18,30,20,15,35,27]
2. 15 is min in array[1,n-1] = [8,15,49,25,18,30,20,22,35,27]
3. 18 is min in array[2,n-1] = [8,15,18,25,49,30,20,22,35,27]
4. 20 is min in array[3,n-1] = [8,15,18,20,49,30,25,22,35,27]
5. 22 is min in array[4,n-1] = [8,15,18,20,22,30,25,49,35,27]
6. 25 is min in array[5,n-1] = [8,15,18,20,22,25,30,49,35,27]
7. 27 is min in array[6,n-1] = [8,15,18,20,22,25,27,49,35,30]
8. 30 is min in array[7,n-1] = [8,15,18,20,22,25,27,30,49,35]
9. 35 is min in array[8,n-1] = [8,15,18,20,22,25,27,30,35,49]
10. End of Selection Sort

Bubble Sort: Check each element with it's right neighbout and if it is bigger than it's neighbour swap them, do this operation for array[i] i from 0 to n-2.

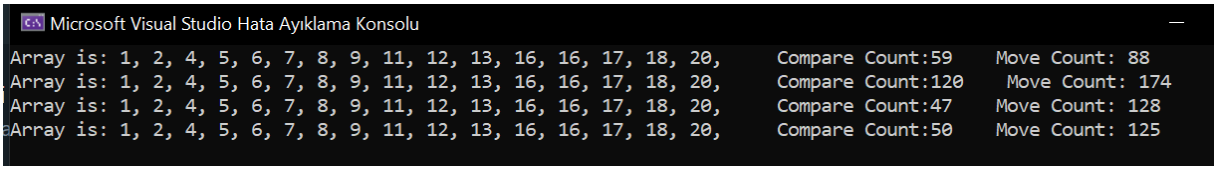
Array for Bubble Sort is = [22,8,49,25,18,36,20,15,35,27] and size of the array is 10.

1. Step 1 array = [8,22,25,18,30,20,15,35,27,49]
2. Step 2 array = [8,22,18,25,20,15,30,27,35,49]
3. Step 3 array = [8,18,22,20,15,25,27,30,35,49]
4. Step 4 array = [8,18,20,15,22,25,27,30,35,49]
5. Step 5 array = [8,18,15,20,22,25,27,30,35,49]
6. Step 6 array = [8,15,18,20,22,25,27,30,35,49]
7. End of Bubble Sort

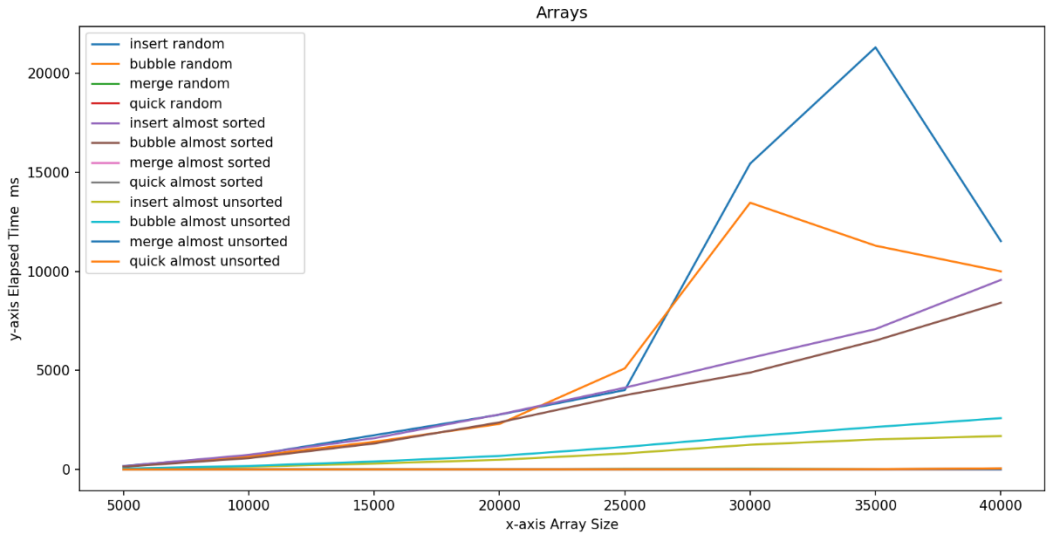
For the same array bubble sort requires least step than insertion sort (6 < 9).

**Question2:**

Among Insertion Sort, Bubble Sort, Merge Sort and Quick Sort theoretically quick sort and merge sort has great speed advantage over insertion sort and bubble sort. During the experiment measured times for algorithms shows that distinction clearly. Also move counts and comparsion counts are relatively changes. Results of sorting given array and calculating move count and comparison count:




As it can be seen, bubble sort is least efficient algorithm for the given array in comparison count and move count. However, comparison count of other three algorithms are similar. Merge Sort and Quick Sort also has similar move counts.




The table above, demonstrates how algorithms' runtimes changes due to orientation of arrays.

We can clearly say that merge sort and quick sort has huge performance superiority to insertion and bubble sort. Insertion sort and bubble sort are easy to observe due to their time complexities. Worst case and best case shows great difference in insertion and bubble sort. However, quick sort and merge sort are hard to observe especially if they are in same graph

with insertion and bubble sort. Because there is a huge difference between their elapse times. Theoretically, insertion and bubble sort hold their time complexities. However, graphs are not smooth as theoretical ones due to orientation of arrays. Arrays in this experiment are conducted with random re arrengenments therefore, there are some values in arrays that we can not control and think about. So thats why there is a litttle difference in graph for insertion and bubble sort. We can not observe quick sort and merge sort in this chart due to being unscalable with insertion and bubble sort for elapsed times.