CPSC 250

William Friend

**Homework #2 –Chapter 11. Sorting Algorithms**

**Due Date: Oct. 5, 2021 (No Late Submissions will be Accepted.)**

Grades depend on neatness and clarity. Write your answers with enough detail about your approach and concepts used, so that the grader will be able to understand it easily.

When you upload your file, please make one file that includes all figures inside. if it is not one file, I will not grade yours.

1. (10 points) Textbook p 327, Question 1 & 2
2. (10 points) Textbook p 330, Question 3 & 4
3. (10 points) Textbook p 332, Question 5 & 6
4. (10 points) Textbook p 337, Question 7
5. (10 points) Textbook p 343, Question 9
6. (10 points) Textbook p 346, Question 10
7. (40 points) Textbook p 350, Exercises 6, 7, 10, & 12

CPSC 250

**Question 1** Trace the selection sort as it sorts the following array into ascending order:

20 80 40 25 60 30’

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 20 | 80 | 40 | 25 | 60 | 30 |
| 20 | 30 | 40 | 25 | 60 | 80 |
| 20 | 30 | 40 | 25 | 60 | 80 |
| 20 | 30 | 25 | 40 | 60 | 80 |
| 20 | 25 | 30 | 40 | 60 | 80 |
| 20 | 25 | 30 | 40 | 60 | 80 |

**Question 2** Repeat the previous question, but instead sort the array into descending order.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 20 | 80 | 40 | 25 | 60 | 30 |
| 30 | 80 | 40 | 25 | 60 | 20 |
| 60 | 80 | 40 | 25 | 30 | 20 |
| 60 | 80 | 40 | 30 | 25 | 20 |
| 60 | 80 | 40 | 30 | 25 | 20 |
| 80 | 60 | 40 | 30 | 25 | 20 |

**Question 3** Trace the bubble sort as it sorts the following array into ascending order:

25 30 20 80 40 60.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 25 | 30 | 20 | 80 | 40 | 60 |
| 25 | 20 | 30 | 80 | 40 | 60 |
| 25 | 20 | 30 | 80 | 40 | 60 |
| 25 | 20 | 30 | 40 | 80 | 60 |
| 25 | 20 | 30 | 40 | 60 | 80 |
| 25 | 20 | 30 | 40 | 60 | 80 |
| 20 | 25 | 30 | 40 | 60 | 80 |
| 20 | 25 | 30 | 40 | 60 | 80 |
| 20 | 25 | 30 | 40 | 60 | 80 |
| 20 | 25 | 30 | 40 | 60 | 80 |
| 20 | 25 | 30 | 40 | 60 | 80 |
| 20 | 25 | 30 | 40 | 60 | 80 |
| 20 | 25 | 30 | 40 | 60 | 80 |
| 20 | 25 | 30 | 40 | 60 | 80 |
| 20 | 25 | 30 | 40 | 60 | 80 |

**Question 4** Repeat the previous question, but instead sort the array into descending order.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 25 | 30 | 20 | 80 | 40 | 60 |
| 30 | 25 | 20 | 80 | 40 | 60 |
| 30 | 25 | 20 | 80 | 40 | 60 |
| 30 | 25 | 80 | 20 | 40 | 60 |
| 30 | 25 | 80 | 40 | 20 | 60 |
| 30 | 25 | 80 | 40 | 60 | 20 |
| 30 | 25 | 80 | 40 | 60 | 20 |
| 30 | 25 | 80 | 40 | 60 | 20 |
| 30 | 80 | 25 | 40 | 60 | 20 |
| 30 | 80 | 40 | 25 | 60 | 20 |
| 30 | 80 | 40 | 60 | 25 | 20 |
| 30 | 80 | 40 | 60 | 25 | 20 |
| 80 | 30 | 40 | 60 | 25 | 20 |
| 80 | 40 | 30 | 60 | 25 | 20 |
| 80 | 40 | 30 | 60 | 25 | 20 |
| 80 | 40 | 60 | 30 | 25 | 20 |
| 80 | 40 | 60 | 30 | 25 | 20 |
| 80 | 60 | 40 | 30 | 25 | 20 |
| 80 | 60 | 40 | 30 | 25 | 20 |
| 80 | 60 | 40 | 30 | 25 | 20 |
| 80 | 60 | 40 | 30 | 25 | 20 |

**Question 5** Trace the insertion sort as it sorts the array in Checkpoint Question 3 into ascending order.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 25 | 30 | 20 | 80 | 40 | 60 |
| 25 | 30 | 20 | 80 | 40 | 60 |
| 25 | 25 | 30 | 80 | 40 | 60 |
| 20 | 25 | 30 | 80 | 40 | 60 |
| 20 | 25 | 30 | 80 | 40 | 60 |
| 20 | 25 | 30 | 80 | 80 | 60 |
| 20 | 25 | 30 | 40 | 80 | 60 |
| 20 | 25 | 30 | 40 | 60 | 80 |

**Question 6** Repeat the previous question, but instead sort the array into descending order.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 25 | 30 | 20 | 80 | 40 | 60 |
| 25 | 25 | 20 | 80 | 40 | 60 |
| 30 | 25 | 20 | 80 | 40 | 60 |
| 30 | 25 | 20 | 80 | 40 | 60 |
| 30 | 30 | 25 | 20 | 40 | 60 |
| 80 | 30 | 25 | 20 | 40 | 60 |
| 80 | 30 | 30 | 25 | 20 | 60 |
| 80 | 40 | 30 | 25 | 20 | 60 |
| 80 | 40 | 40 | 30 | 25 | 20 |
| 80 | 60 | 40 | 30 | 25 | 20 |

**Question 7** By drawing a diagram like the one shown in Figure 11-6, trace the merge sort as it sorts the following array into ascending order: 25 30 20 80 40 60.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | 25 | 30 | 20 | 80 | 40 | 60 |  |  |  |  |  |  |
|  |  |  | 25 | 30 | 20 |  |  |  |  |  |  | 80 | 40 | 60 |  |  |  |
|  |  |  | 25 | 30 | 20 |  |  |  |  |  |  | 80 | 40 | 60 |  |  |  |
|  |  |  | 25 | 30 |  |  |  |  |  |  |  | 80 | 40 |  |  |  |  |
|  |  |  | 25 | 30 |  |  |  |  |  |  |  | 40 | 80 |  |  |  |  |
|  |  |  | 20 | 25 | 30 |  |  |  |  |  |  | 40 | 60 | 80 |  |  |  |
|  |  |  |  |  |  | 20 | 25 | 30 | 40 | 60 | 80 |  |  |  |  |  |  |

**Question 9** Trace the quick sort’s partitioning algorithm as it partitions the following array:

38 16 40 39 12 27

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 38 | 16 | 40 | 39 | 12 | 27 |  |
| mid = first + (last – first) /2  = 0 + (5-0) / 2  =2  sort index’s 0, 2, 5 in acending order | | | | | | |
| 27 | 16 | 38 | 39 | 12 | 40 |  |
| swap index 2 and 4 | | | | | | |
| 27 | 16 | 12 | 39 | 38 | 40 |  |
| pivot = 38 at index 4  indexFromLeft = 1, indexFromright = 3  find first value/index from indexFromLeft greater or equal to pivot=38  39 at index 3  find first value/index from indexFromRight smaller or equal to pivot=38  12 at index 2  indexFromLeft>IndexFromRight so swap indexFromLeft and pivot | | | | | | |
| 27 | 16 | 12 | 38 | 39 | 40 |  |
| pivot = 33  partitioned based on 38  27, 16, 12 < 38 < 39, 40  **partitioned** | | | | | | |

**Question 10** Suppose that you sort a large array of integers by using a merge sort. Next you use a binary search to determine whether a given integer occurs in the array. Finally, you display all of the integers in the sorted array.

**a.** Which algorithm is faster, in general: the merge sort or the binary search? Explain in terms of Big O notation.

Merge sort is O(nlogn)

Binary search is O(logn)

log 2 < 2 log 2

Therefore, binary search is faster than merge sort

**b.** Which algorithm is faster, in general: binary search or displaying the integers? Explain in terms of Big O notation.

binary search is O(logn)

display function is O(n)

log2 < 2

Therefore, binary search is faster than display function.

CPSC 250

1. (40 points) Textbook p 350, Exercises 6, 7, 10, & 12

6.

worst case, reversed

bubble sort = O(n(n+1)/2) - n

n = 25

(n(n+1)/2) - n

(n^2+n-2n)/2

(n^2-n)/2

n(n-1)/2

25\*24/2

=300 comparisons

best case, already sorted’

bubble sort = O(n-1)

=24 comparisons

7.

Worst behavior for bubble sort will always be when the array is in the reverse order it should be in.

ex.

{3, 2, 1} is bad when you want {1, 2, 3}

**10**.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | 20 | 80 | 40 | 25 | 60 | 30 |  |  |  |  |  |  |
|  |  |  | 20 | 80 | 40 |  |  |  |  |  |  | 25 | 60 | 30 |  |  |  |
|  |  | 20 | 80 |  | 40 |  |  |  |  |  | 25 | 60 |  | 30 |  |  |  |
|  |  | 20 | 80 |  |  |  |  |  |  |  | 25 | 60 |  |  |  |  |  |
|  |  | 20 | 80 |  |  |  |  |  |  |  | 25 | 60 |  |  |  |  |  |
|  |  |  | 20 | 40 | 80 |  |  |  |  |  |  |  | 25 | 30 | 60 |  |  |
|  |  |  | 20 |  |  |  |  |  |  |  |  |  | 25 |  |  |  |  |
|  |  |  |  | 40 |  |  |  |  |  |  |  |  |  | 30 |  |  |  |
|  |  |  |  | 30 |  |  |  |  |  |  |  |  |  | 40 |  |  |  |
|  |  |  |  |  | 80 |  |  |  |  |  |  |  |  |  | 60 |  |  |
|  |  |  |  |  | 60 |  |  |  |  |  |  |  |  |  | 80 |  |  |
|  |  |  |  |  |  | 20 | 25 | 30 | 40 | 60 | 80 |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

**12**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **20** | **80** | **40** | **25** | **60** | **10** | **15** |  |  |  |
| **quickSOrt(theArray[],0,6)**  **MIN\_SIZE=3**  last-first+1=6-0+1=7>3, activate call to  partition(theArray[],0,6)  mid = first+(last-first)/2  =0+(0-6)/2  =3 | | | | | | | | | |
| **sort index 0, 3, 6** | | | | | | | | | |
| **15** | **80** | **40** | **20** | **60** | **10** | **25** |  |  |  |
| **swap index 3 and 5** | | | | | | | | | |
| **15** | **80** | **40** | **10** | **60** | **20** | **25** |  |  |  |
| **pivot = 20 at 5**  **indexFromLeft = 2, indexFromRight = 3**  **40 at 2 > pivot**  **10 at 3 < pivot** | | | | | | | | | |
| **15** | **10** | **40** | **80** | **60** | **20** | **25** |  |  |  |
| **pivot = 20 at 5**  **indexFromLeft = 1, indexFromRight = 4**  **40 at 2 > pivot**  **10 at 1 < pivot** | | | | | | | | | |
| **15** | **10** | **20** | **80** | **60** | **40** | **25** |  |  |  |
| **pivot is 2\partitioned based on pivot element 20**  **quickSort(TheArray[],0,1) quickSort(TheArray[],0,1)**  **quickSort(theArray,0,1)** | | | | | | | | | |
| **10** | **15** | **20** | **80** | **60** | **40** | **25** |  |  |  |
| **10** | **15** | **20** | **25** | **60** | **40** | **80** |  |  |  |
| **10** | **15** | **20** | **25** | **40** | **60** | **80** |  |  |  |
| |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | **10** | **15** | **20** | **25** | **40** | **60** | **80** | |  |  |  |  |  |  |  |  |  |
| |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | **10** | **15** | **20** | **25** | **40** | **60** | **80** | |  |  |  |  |  |  |  |  |  |
| |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | **10** | **15** | **20** | **25** | **40** | **60** | **80** | |  |  |  |  |  |  |  |  |  |

