**M1522.000900 Data Structure**

**Spring 2025, Kang**

**Homework 3 Answer Sheet [1/3]**

2023-12753 EunSu Yeo

* Write your answers on the “your answer” columns.
* Do **NOT** write anything on “score” columns.
* Before submission, delete all the blue-colored texts, and convert the file into **PDF-format**.
* Write your proof on the solution sheet and leave the “your answer” column blank.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Question | | Your Answer | Points | Score |
| 1 | (1) | node.val | 2 |  |
| dfs(child, result); | 2 |  |
| “)” | 2 |  |
| (2) | A B ) C ) D E ) ) ) | 4 |  |
| 2 | (1) |  | 7 |  |
| (2) |  | 8 |  |

**M1522.000900 Data Structure**

**Spring 2025, Kang**

**Homework 3 Answer Sheet [2/3]**

2023-12753 EunSu Yeo

|  |  |  |  |
| --- | --- | --- | --- |
| Question | Your Answer | Points | Score |
| 3 | |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | | Node | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | | Parent | 12 | 0 | 4 | 2 | / | 4 | 4 | 8 |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | | Node | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | | Parent | 4 | 8 | 4 | 10 | / | 12 | 12 | 12 | | 15 |  |
| 4 | 1. List of children: 2. Left-child/Right-sibling: | 10 |  |
| 5 | |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 41 | 25 | 11 | 3a | 75 | 12 | 3b | 10 | 49 | 53 | | 3a | 41 | 25 | 11 | 3b | 75 | 12 | 10 | 49 | 53 | | 3a | 3b | 41 | 25 | 11 | 10 | 75 | 12 | 49 | 53 | | 3a | 3b | 10 | 41 | 25 | 11 | 12 | 75 | 49 | 53 | | 10 |  |
| 6 |  | 10 |  |

**M1522.000900 Data Structure**

**Spring 2025, Kang**

**Homework 3 Answer Sheet [3/3]**

2023-12753 EunSu Yeo

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Question | | Your Answer | Points | Score |
| 7 | (1) |  | 5 |  |
| (2) |  | 5 |  |
| (3) |  | 5 |  |
| (4) |  | 5 |  |
| 8 | | 42, 59, 91 | 10 |  |

**Homework 3  
Solution Sheet**

2023-12753

EunSu Yeo

1. **Question 1 [10 points]**
   1. **Q1(1) [3 points]**

**Solution.**

Write here

* 1. **Q1(2) [3 points]**

**Solution.**

Write here

* 1. **Q1(3) [3 points]**

**Solution.**

Write here

* 1. **Q1(4) [3 points]**

**Solution.**

Write here

1. **Question 2 [15 points]**
   1. **[7 points]**

**Solution.**

Write here

* 1. **[8 points]**

**Solution.**

Write here

1. **Question 3 [15 points]**

**Solution.**

Write here

(0,1)  
Both sets have equal size (1,1)  
Root 1 > 0 → attach 1 under 0  
parent[1] = 0  
size[0] = 2

(2,3)  
Equal size (1,1)  
Root 3 > 2 → attach 3 under 2  
parent[3] = 2  
size[2] = 2

(4,5)  
Equal size (1,1)  
5 > 4 → attach 5 under 4  
parent[5] = 4  
size[4] = 2

(6,5)  
Find roots: 6 is its own root, 5’s root is 4  
Sizes: 6 has 1, 4 has 2  
Smaller set 6 attaches under 4  
parent[6] = 4  
size[4] = 3

(3,5)  
Root of 3 is 2 (parent[3] = 2)  
Root of 5 is 4  
Sizes: 2 has 2, 4 has 3  
Smaller root 2 attaches under 4  
parent[2] = 4  
size[4] = 5 (2+3)

(8,9)  
Equal size (1,1)  
9 > 8 → attach 9 under 8  
parent[9] = 8  
size[8] = 2

(7,8)  
Roots: 7 is root, 8 is root  
Sizes: 7 has 1, 8 has 2  
Smaller root 7 attaches under 8  
parent[7] = 8  
size[8] = 3

(2,7)  
Find roots: parent[2] = 4 → root 4  
parent[7] = 8 → root 8  
Sizes: 4 has 5, 8 has 3  
Smaller root 8 attaches under 4  
parent[8] = 4  
size[4] = 8

(10,11)  
Equal size (1,1)  
11 > 10 → attach 11 under 10  
parent[11] = 10  
size[10] = 2

(12,15)  
Equal size (1,1)  
15 > 12 → attach 15 under 12  
parent[15] = 12  
size[12] = 2

(13,15)  
Roots: 13 is root, 15’s root is 12  
Sizes: 13 has 1, 12 has 2  
Smaller root 13 attaches under 12  
parent[13] = 12  
size[12] = 3

(14,13)  
Roots: 14 is root, 13’s root is 12  
Sizes: 14 has 1, 12 has 3  
Smaller root 14 attaches under 12  
parent[14] = 12  
size[12] = 4

(6,11)  
Roots: 6’s root is 4  
11’s root is 10  
Sizes: 4 has 8, 10 has 2  
Smaller root 10 attaches under 4  
parent[10] = 4  
size[4] = 10

(0,14)  
size[0] = 2  
size[12] = 4  
Weighted union rule: attach the smaller set under the larger set  
Since 2 < 4, the root of 0’s set attaches under the root of 12’s set  
Therefore, parent[0] = 12

1. **Question 4 [10 points]**

**Solution.**

Write here

In the List of Children representation, each row in a table stores the index, value, and a pointer to the parent node  
The child nodes that have this node as their parent are connected via a linked list  
Each linked list node consists of a pointer to the index and a pointer to the next node in the list

Let n be the total number of nodes

Data space: nD

Index space: nI (index of the table)

Pointer space:

nP (pointer to parent)

(2n - 1)P (pointers to children)

(n - 1)P (index pointers stored in the linked list)

Total space: n(I + D + 4P) - 2P

Overhead (non-data space): n(I + 4P) - 2P

Space overhead =

In the Leftmost Child/Right Sibling representation, each node has pointers to the leftmost child and the right sibling

Let n be the total number of nodes

Data space: nD

Index space: nI

Pointer space:

nP (pointer to parent)

n × 2P (pointers to leftmost child and right sibling)

Total space: n(I + D + 3P)

Overhead (non-data space): n(I + 3P)

Space overhead =

1. **Question 5 [10 points]**

Write here

By using bubble sort in the ppt 16, at each iteration ,“Bubble up” the smallest element from the input to the output.

1. **Question 6 [10 points]**

Write here

The mergeSort function is called times.  
For subarrays of size less than or equal to the threshold, insertionSort is called instead.

In the recursion tree, the number of calls to insertion sort corresponds to the number of leaf nodes.

We can think the Leaf nodes as the points where mergeSort stops and calls insertionSort.

Therefore, the number of insertion sort calls equals the number of leaf nodes, which is for a complete binary tree.

Hence, the number of insertion sort calls is

1. **Question 7 [20 points]**
   1. **[5 points]**

**Solution.**

Write here

Merge Sort always divides and merges the array regardless of input order, so the best-case time complexity remains O(nlogn).

* 1. **[5 points]**

**Solution.**

Write here

Choosing the minimum value as the pivot results in the worst possible partition each time (highly unbalanced splits), leading to a time complexity of O(n^2).

* 1. **[5 points]**

**Solution.**

Write here

Since the middle index pivot divides the array more evenly even when the input is sorted, the time complexity is O(nlogn)

* 1. **[5 points]**

**Solution.**

Write here

Since sorting all n elements requires at least O(n log n) comparisons, sorting k elements in O(n) limits k.

Finding the **maximum element** in an array takes O(n) time.

Finding the **second largest element** can be done in O(logn) time using efficient methods such as a tournament tree.

Similarly, finding the third, fourth, ..., k-th largest element can be done in roughly O(logn) time each.

Therefore, selecting the top k elements can be done in about O(n+klogn) time.

Hence, the total time complexity is approximately O(n+klogn), and to keep it O(n), we need klogn=O(n).

may be the answer.

1. **Question 8 [10 points]**

**Solution.**

Write here

As the problem gives us that given array is partitioned by the partition function shown in quicksort, we can think that l and r is fixed to l=start index of this array and r=end index of this array. So, we need to find the possible pivots.

In partition function, we know that given array is divided into 2 parts which is lower than the pivot and which is greater than the pivot value.

So, we need to divide the given array into 2 parts using assumed pivot.

Due to this step, we can find 42 and 59 can be pivot.

The elements before the 42 is smaller than 42 and the elements after the elements is bigger than the 42 in the given array. So, 42 can be the pivot.

As the same reason 59 and 91 can be the pivot.