HW-4 CSL2020 (Jan 2021)

Assigned: 9 Jan 2021 Submission deadline: 16 Jan 2021 (Saturday) 5 pm Marks: 6+1Bonus

---

Submit two files with name: hw4_q1_roll_number.c and hw4_q2_roll_number.c

1. [HEAP] Write a C program which works as follows: (You may create two global arrays of max_size=100 for the purpose of this problem. However, it is not mandatory.) [3 Marks]

We will test your code for both min- and max- heaps. We will always start with input 1 below. You need to create 2 arrays: one for min-heap and the other for max-heap. All inputs in the question will be positive integers. You should print the current array after finishing each operation so that we can test your code. If the current operation is of type X then print only the max-heap, and if the operation is of type N then print only min-heap (not both together).

(i) On input 1 X n m1 m2 m3 …..

Creates a maX-heap with n positive integers m1, m2, m3, …. You should first fill the array with the integers m1, m2, m3 …. And then convert this array into a max-heap.

(Note that X is used for max heap)

(ii) On input 1 N n m1 m2 m3 …..

Creates a miN-heap with n positive integers m1, m2, m3, …. Similar to **(i)** above.

(Note that N is used for min heap)

(iii and iv) On input 2 X m and on input 2 N m

(iii) Insert integer m into the already existing max heap. **(iv)** same for min heap.

(v and vi) On input 3 X m and on input 3 N m

(v) Delete element m from the existing max heap. **(vi)** same for min heap.

(The way this operation should be implemented is to replace the deleted integer with the last element of the array and then moving elements so that the heap property is satisfied. Note that you may need to move the newly moved element upwards or downwards, depending on the current elements in the heap. If m is not present in the heap, then do nothing.).

(vii and viii) On input 4 X and on input 4 N

(vii) Delete max element from the existing max heap. (viii) Delete min element for min heap

(**ix and x**) **On input 5 X and on input 5 N**

(ix) Sort the array containing max heap. **(x)** Same for min heap.

2. [AVL Tree] Write a C program which works as follows: (You may assume that the tree will not contain more than 100 elements). [3 Marks]

For the purposes of this exercise, you can take a global variable pointing to the root of the AVL tree. You should write separate functions for left-rotate and right-rotate. Call them appropriately to balance the tree when needed.

(i) On input 1 n m1 m2 m3 …..

Create an AVL tree with n positive integers m1, m2, m3, …. You should construct the AVL tree with insertion of elements in the given order.

(ii) On input 2 m

Insert the element m in the existing AVL tree.

(iii) On input 3 m

Delete the element m from the existing AVL tree. Do nothing if m is not present in the tree.

(iv) On input 4

Print the in-order traversal of the tree. For each node, print the contents as (data, height, balance factor). For example, if you have an AVL tree with 2 nodes, with the root and its left child containing data 5 and 3 respectively, then you should print (3,1,0), (5,0,-1).

Questions for extra marks (not mandatory): [1 Marks]

These are not specific to AVL trees. I am putting them here since you have already created an AVL tree. Try not to use any specific property of AVL trees while implementing these.

(v) **On input 5**

Print the width, and the perimeter of the tree.

Width: The width (or diameter) of a tree is the number of nodes on the longest path between two leaf nodes. (It may or may not pass through the root).

Perimeter: Boundary nodes of the tree (sometimes also called boundary traversal – Search it).

(vi) **On input 6 (a,b)**

Print the lowest common ancestor of node containing ‘a’ and ‘b’

(vii) On input 7 (a,b)

Print the route (i.e. values of intermediate nodes) when you move from an ancestor node containing value ‘a’ to a descendant node containing value ‘b’ in the tree. Write -1 if no such path exists.