CS 202 Fundamental Structures of Computer Science II

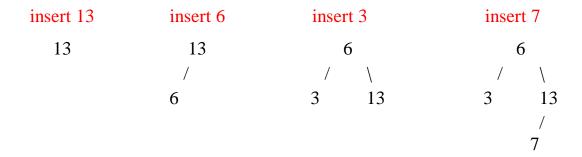
Assignment 3 – Heaps and AVL Trees

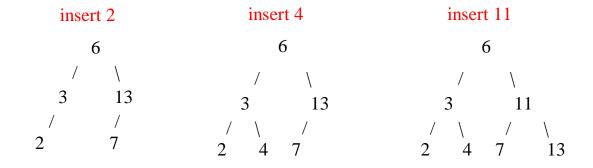
Assigned on: 28 July 2020 (Tuesday)

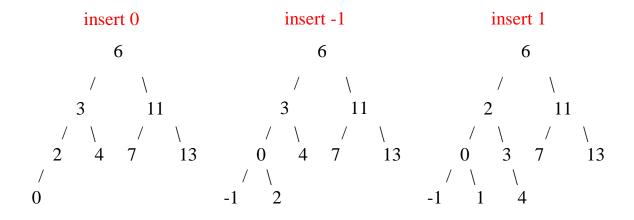
Due Date: 09 August 2020 (Sunday) by 23:55

1) Question Part (50 points)

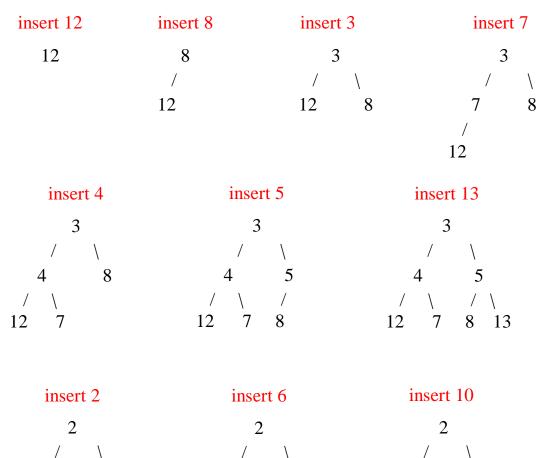
a)

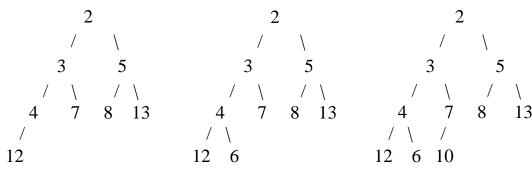


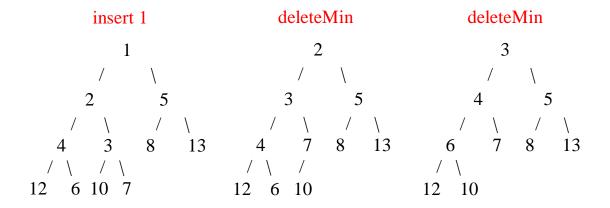




b)

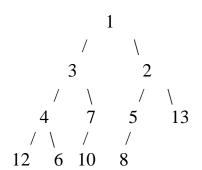






c) Let's consider this min heap. If we print the keys as encountered in preorder traversal the result will be the following:

It is not sorted because there is no order on the children of a



parent. Sometimes the left child is greater than the right one, sometimes the right child is bigger than the left child. It depends on the insertion order. Hence, the sorted order is not obtained by any traversal method.

Postorder traversal: 12, 6, 4, 10, 7, 3, 8, 5, 13, 2, 1 – Not Sorted

d)
$$minN(0) = 0$$

$$minN(1) = 1$$

$$minN(2) = 2$$

$$minN(3) = 4$$

minN(h) = minN(h-1) + minN(h-2) + 1

$$minN(4) = minN(3) + minN(2) + 1 = 7$$

$$minN(5) = minN(4) + minN(3) + 1 = 12$$

$$minN(6) = minN(5) + minN(4) + 1 = 20$$

$$minN(7) = minN(6) + minN(5) + 1 = 33$$

$$minN(8) = minN(7) + minN(6) + 1 = 54$$

$$minN(9) = minN(8) + minN(7) + 1 = 88$$

$$minN(10) = minN(9) + minN(8) + 1 = 143$$

```
Kübra Okumuş
   21600980
   minN(11) = minN(10) + minN(9) + 1 = 232
   minN(12) = minN(11) + minN(10) + 1 = 376
   minN(13) = minN(12) + minN(11) + 1 = 609
   minN(14) = minN(13) + minN(12) + 1 = 986
   minN(15) = minN(14) + minN(13) + 1 = 1596
e) bool isMinHeap(TreeNode * root, int index, int size){
   if root is NULL
          return true;
   if index is greater than or equal to the size // checks complete binary tree property
          return false;
   if (the left child is exist AND it is less than the root) OR (the right child is
     exist AND it is less than the root)
          return false;
   return isMinHeap(root->left, index*2+1, size) AND
           isMinHeap(root->right, index*2+2, size);
   }
   bool isMinHeap(BinaryTree bt){
          return isMinHeap(bt.getRoot, 0, bt.getSize);
```

}

Kübra Okumuş 21600980

2) Report of the Programming Part

Heap is a complete binary tree. If it's all roots have data which are greater than or equal to

their children, then it is called maxheap. Moreover, if all roots of a heap are smaller than their

children, is it a minheap. Index of a left child is equal to 2 * (index of parent) + 1 and index of a

right child is equal to 2 * (index of parent) + 2. First element of the heap is the root element.

Heap satisfies the heap properties after insertion or deletion.

Heapsort function takes an array of integers and builds a heap with them. Then it transfers

the first item of the heap, which is the maximum element, to the sorted array. At the same time,

the first position of the heap takes the item in the last position. Then, it rebuilds the heap with the

size-1. It repeats this process until the whole heap is sorted.

Heapsort function does at most log(N) iterations while rebuilding the heap after putting the

last item to the first position. heaprebuild function is called N times, and it makes 2 comparisons

each time(one for comparing two children with each other, other one is for comparing the parent

with the biggest child). Hence, heapsort makes theoretically at most 2Nlog(N) comparisons.

Data1

Number of data points: 1000

Number of comparisons: 12280

Theoretical number of comparisons: $2 * 1000 * \log 1000 = 13815$

Data2

Number of data points: 2000

Number of comparisons: 27665

Theoretical number of comparisons: $2 * 2000 * \log 2000 = 30403$

Kübra Okumuş 21600980

Data3

Number of data points: 3000

Number of comparisons: 43883

Theoretical number of comparisons: $2 * 3000 * \log 3000 = 48038$

Data4

Number of data points: 4000

Number of comparisons: 61327

Theoretical number of comparisons: $2 * 4000 * \log 4000 = 66352$

Data5

Number of data points: 5000

Number of comparisons: 78567

Theoretical number of comparisons: $2 * 5000 * \log 5000 = 85171$