EC 504 – Fall 2020 – Homework 4

Due Thursday, Oct 22, 2020, submitted in the directory `/projectnb/alg504/username/HW4` **on your SCC account by 11:59PM.**

Reading Assignment on GitHub:
Sorting_Data_Structure (Chapters 6, 7) and
Trees.pdf (12, 13) and TreeMath_B.pdf (Appendix B .5)

1. (20 pts) Determine whether the following statements are true or false, and explain briefly why.

   (a) If doubling the size ($N \to 2N$) causes the execute time $T(N)$ of an algorithm to increase by a factor of 4, then $T(N) \in O(4N)$.
   Solution: False: If $T(N) \sim \text{const}N$ then when $N \to 2N$ then $TN) \to T(2N) \sim$)const$2N$ – a factor of 2. Need to squared it $T(N) \in O(N^2($ does it. This is grows larger so in general $T(N) \in O(4N) = O(N)$ would be false

   (b) The height of a binary tree is the maximum number of edges from the root to any leaf path. The maximum number of nodes in a binary tree of height h is $2^{h+1} - 1$.
   Solution: True. See that it works for a complete binary tree at all levels. One level is h = 0, so 1 node. 2 levels is h = 1, so it has 3 nodes. Continue by induction.

   (c) In a binary search tree with no repeated keys, deleting the node with key x, followed by deleting the node with key y, will result in the same search tree as deleting the node with key y, then deleting the node with key x.
   Solution: False: Deletes do not commute. When you delete with one child you replace it directly but when you delete with two children you replace it by the minimun in the right subtree. Here is very simple conter example. One is enough.

```
    3                    6                      6
   / \                  / \                      \
  2   8   delete 3     2   8   delete 2           8
     /
    6


                          3
BUT       delete 2         \     delete 3         8
                            8                    /
                           /                    6
                          6
```

   (d) Inserting numbers 1, . . . , n into a binary min-heap in that order will take $O(n)$ time.
   Solution: True: This the result is already min-heap order just when you load the array.

(e) The second smallest element in a binary min-heap with all elements with distinct values will always be a child of the root.

Solution: True: Suppose it is were false. Then its parent would be larger that violats the min-heap order'

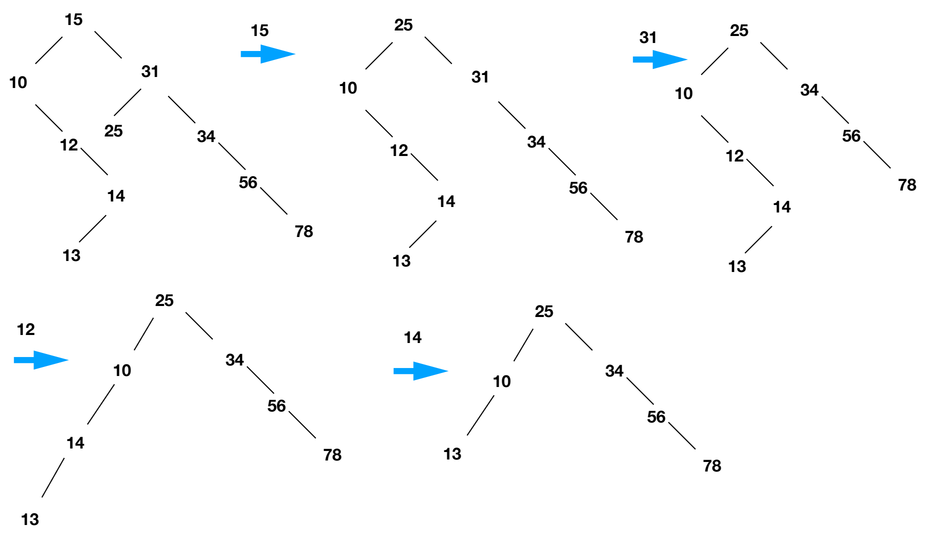
2. (20 pts) This exercise is to learn binary search tree operations

(a) Draw the sequence of binary search trees which results from inserting the following values in left-to-right order, assuming no balancing. 15, 10, 31, 25, 34, 56, 78, 12, 14, 13

Solution: Trial rule. Hard to type. So I don't do every step!

```
15 --> 15 -->  15     --> 15    -->     ...       15
       /      / \        /  \                    /   \
      10    10   31     10  31                  10      31
                            /                    \     / \
                           25                     12  25  34
                                                   \        \
                                                  14         56
                                                  /           \
                                                 13            78
```

(b) Starting from the tree at the end of the previous part, draw the sequence that results from deleting the following nodes in left-to-right order: 15, 31, 12, 14.

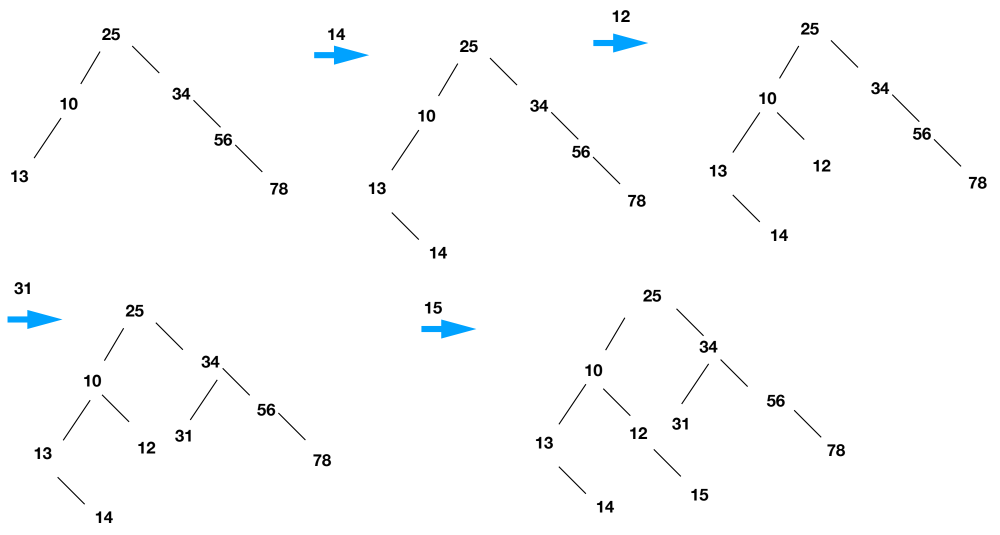


Solution:

(c) After deleting them Draw the sequence of reinserting left-to-right in reverse order: 14, 12, 31, 15. in order into the tree and comment on the result?

Solution:

Comment: Not returned to the same because deletes and inserts are not inverse operations.



3. (20 pts) Reading CRLS Chapter 6 and do the written

   (a) Exercises: 6.1-3, 6.1-4, 6.1-6, and 6.3-3

   (Note chapter 6 give a background to the coding exercise to use an array for a Max Heap. Also there is of course a nice Wikipedia article to look at https://en.wikipedia.org/wiki/Heapsort.)

   Solution:

   Exercise 6.1-3: In a max head the path from the root as you decend to any leaf is a sequence a (sorted) values deceasing. is sorted as you descend. So by the recursive definition of a binary tree each node is itself a max heap of that subtree.

   Exercise 6.1-4: The max heap (with no equal values) has strickly order sequence to smaller values as descend to the leaves (i.e. nodes with no children). Therefore the smallest element must be on leaf at the bottom.

   Exercise 6.1-6: Try it:

```
                   23
              /           \
           17                14
      /         \      /        \
     6           13    10         1
  /    \       /
5        7   12
```

Exercise 6.3-3: Consider with full heap (perfect tree) has rows with 2^d at deapth d. With the last row at deapth d = D there are $n = 1 + 2 + 4 + ... + 2^D = 2^{D+1} - 1$ nodes. So now going back up the number of at height h are

$$2^{D-h} = 2^{D+1}/2^{h+1} = n/2^{h+1} + 1//2^{h+1}$$

So in this case we have cieling of $n/2^{h+1}$. Now is you deplete the lowest row you reduce the height of some elements so this is a maximun.

(b) Exercises: 12.3-2, 12.3-3, 12.3-4, B.5-4

(Note that the degree of in an undirected graph (or tree) is the number links incident on the node. A leaf is a node with degree 1.)

Solution:

Exercises 12.3-2: When you insert an element it follows a path by comparing nodes as you descend utill it is inserted. If there are n comparison it has gone to depth n. The search follows the same path but needs to compare the final value so this is $n + 1$.

Exercises 12.3-3: The inorder walk is walk around the outside of the tree. It has to visit every link twice. To the complexity is the number of links. Since every node has at most 3 links shared. There are at most $O(N)$ links. The mininum number of links is N so this this worst case is also $O(N)$ (Obvious you can't do better than visiting every node!) If you also assume the building the tree as part of the algorithm then the bests is $O(NlogN)$ to insert into complete tree and $O(N^2)$ totally in one line.

Exercises 12.3-4: No it is not communative. Here is an example:; build in 5 2 1 4 3 order. Frist 1 then 2 (left only for 1 and left only for 2)

```
              5                    5                    5
            /                    /                    /
          2           ==>      2          ==>      4
        /   \                  \                 /
      1      4                  4              3
            /                  /
           3                  3
```

consider. Frist 2 then 1 (Right Left for 2 and left only for 1

```
             5                    5                    5
           /                  /                       /
         2           ==>     3           ==>         3
```

```
 /   \                /  \                    \
1      4             1    4                    4
      /
     3
```

Exercises B.5-4: Consider the minunum height H(N) for a tree with N nodes. A binary tree wih N nodes is a recursive structure with a left sub-tree, T_L and a right sub-tree T_R. Let $H(N)$ be the minimun heigth of a tree with N nodes. Base case $N = 1$ we check that $H(1) = \lfloor lg(0) \rfloor = 0$. We kwon the height of T is the 1 plus the max of the hieght of T_L and T_R we have

$$H(N) = 1 + Max[H(k)] \text{for} \quad k = 0, ..., N/2 - 1$$

but now we know $H(k) = \quad \lfloor lg(k) \rfloor$ so

$$H(N) = 1 + \quad \lfloor lg(N/2) \rfloor = 1 - 1 + \quad \lfloor lg(N) \rfloor$$

Coding Exercise

Solution:

```
There is template program on the GitHub but you all the basic
framework can be adapted from the coding exercise in HW1.  One goal of
these coding exercises it to have  a set of C tools and Gnuplot tools
to reuse.
```

4. (40pts) Implement a Max Heap for n elements as and array `int HeapArray[n+1];` of $n+1$ setting elements by placing the integers setting `HeapArray[0] = n` and copying the elements putting the elements in sequence into `HeapArray[i],   for i = 1,2,..., n` (May choose to have an longer array with extra space and a save a value `heapSize` to tell how many are in the heap. This is a useful index in any case!

   (a) Provide the in `HW4_codes/heap.cpp` to enable:

   ```
   (1) Insert random sequence to Heap array
   (2) Bottom up Heapify for Max Heap
   (3) Delete any key and restore Max Heap
   (4) Insert new key and restore Max Heap
   (4) Sort in place and print out array
   ```

   Put final code with `Makefile` in `/projectnb/alg504/username/HW4`

   (b) Analize the behavior of this code with the following plots:

```
(1) Plot timing for range of sise n = 8, 16, 32,....2^20
(2)  Histogram the performance over random permutation of UnsortedList100.txt
(3) Combine these two part to define the average for
n = 8, 16, 32,....2^20 and fit T(n) = a + b n Log[n] + c n*n
```

Place these figures in `/projectnb/alg504/username/HW4` as well.