# EC 504
# Spring, 2021
# Exam 1

## Monday, March 8, 2021

**Instructions**

- Put your name on each page; and both your name and BU ID on the first page.
- This exam is open notes, but no consultation with anyone other than the instructors are permitted during the exam.
- You have one hour and 45 minutes to complete this exam. It is designed to require less than one hour and fifteen minutes of work, so there should not be significant time pressure. Nevertheless, there are many problems, so do not spend too much time on any one question; you can always return to it.
- At the end of the exam, you will have 15 minutes to scan your work and upload it to Gradescope under Exam 1 submission. You don't have to assign pages if you don't want to.

1. (12 pts) Answer True or False to each of the questions below. Each question is worth 2 points. Answer true only if it is true as stated, with no additional assumptions. No explanation is needed, but any explanation is likely to earn partial credit, and no explanation will not earn any credit if the answer is wrong.

   (a) $n^{1.1} \in O((0.1)^n)$

   **Solution:** False. Note that $0.1^n$ goes to 0 as $n$ increases!

   (b) Inserting numbers $1, \dots, n$ into a binary min-heap in that order will take $\Theta(n)$ time.

   **Solution:** True. Note that, since we are inserting them in increasing order, there will be no swaps, as inserting at the end means the parent will have a larger value.

   (c) If $X$ is the inorder successor of $Y$, then $X$ has no right child.

   **Solution:** False. No left child.

   (d) Consider a B+-tree in which the maximum number of keys in a non-leaf node is 5. Then, the minimum number of keys in any non-root, non-leaf node is 3.

   **Solution:** False. Max keys = 5 means 6 children, so minimum number of children is 3, which corresponds to 2 keys.

   (e) In the KMP algorithm for string matching, the prefix function for the string "ABABCAB" is [0,0,1,2,1,1,2].

   **Solution:** False. It is [0,0,1,2,0,1,2].

   (f) A perfect binary tree is a full binary tree where all the leaves are at the same level (the last level). A Red-Black Tree which is also a perfect binary tree can have all black nodes

   **Solution:** True. Every path from root to a leaf has the same height, which will be the black height.

2. (16 pts) Quick questions

   (a) A B+-tree of order 4 is built from scratch by 10 successive insertions. What is the minimum number of node splitting operations that must take place?

   **Solution:** There must be a minimum of 3 splits. A node must split when there are four keys in a node

```
Insertion of 3 keys
    10 20 30

Insertion of 4th key (1st split)
     30
   /   \
10,20   30,40     1 SPLIT

       30
   /       \
9,10,20   30,40,50

       30, 50
   /       \     \
9,10,20   30,40   50,60     1 split

         30, 50
   /       \        \
9,10,20   30,40,45   50,60,70

         30, 50,80
   /       \        \
9,10,20   30,40,45   50,60  70,80   1 split
```

(b) The height of a tree is the length of the longest root-to-leaf path in it, in terms of number of edges in the path. What are the minimum and maximum number of nodes in a binary tree of height 5?

**Solution:** The minimum number of nodes are 6 nodes in a row (1,2,3,4,5,6), e.g. The maximum is a perfect binary tree where the bottom level includes $2^5$ nodes, for a total of $1 + 2 + 4 + 8 + 16 + 32 = 63$.

(c) Consider a complete binary tree with n elements, where the left and the right subtrees of the root are max-heaps. What is the worst case complexity for an efficient algorithm that converts this tree to a max-heap?

**Solution:** This is what you get when you do BuildHeap, for the last step. You just have to downheap the root, which is $O(\log(n))$.

(d) Consider two binomial heaps $H_1$, $H_2$, where $H_1$ has binomial trees of rank 0, 2,3 and 4, and $H_2$ has binomial trees of ranks 0, 3, and 4. What are the ranks of the binomial trees that result when we join (merge, not lazy merge) $H_1$ and $H_2$?

**Solution:** The merged $H_1 + H_2$ will have trees of ranks 1, 2, 4, 5. The two rank 0 trees will merge into a rank 1 tree. The rank 2 tree will stay. The two rank 3 trees will merge into a rank 4 tree, creating 3 rank 4 trees. Two of the rank 4 trees will merge into a rank 5 tree.

3. (6 pts) Given a binary tree (not a binary search tree), post-order traversal of this tree yields the following sequence: 1,6,5,9,10,7. Draw a complete binary tree of height 2 consistent with this traversal.

**Solution:**

```
       7
   /      \
  5        10
 /  \      /
1     6   9
```

4. (6 pts) Consider a hash table of size seven, with starting index zero, and a hash function (3x + 4)mod 7. Assuming the hash table is initially empty, which of the following is the contents of the table when the sequence 1, 3, 8, 10 is inserted into the table using open hashing with linear probing for collision resolution?

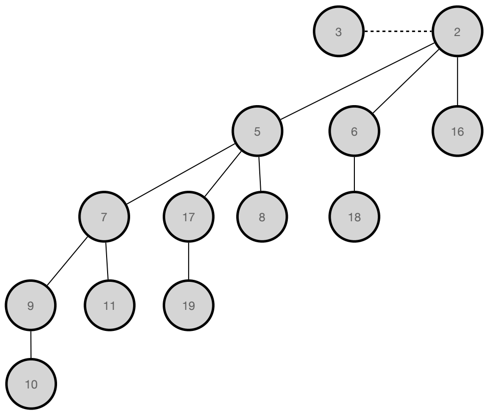
**Solution:** 1 goes to 0; 3 goes to 6; 8 goes to 0, slides to 1, 10 goes to 6, slides to 0, then to 1, then to 2. End result is [1, 8, 10, –, –,–, 3]

5. (6 pts) A binary max-heap implemented using an array is given as $[25, 14, 16, 13, 10, 8, 12]$. What is the content of the array after two delete max operations?

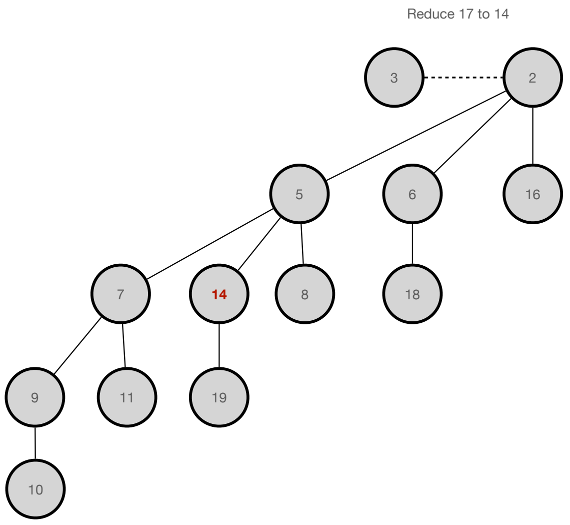
**Solution:** Delete 1: $[25, 14, 16, 13, 10, 8, 12] \to [16, 14, 12, 13, 10, 8]$

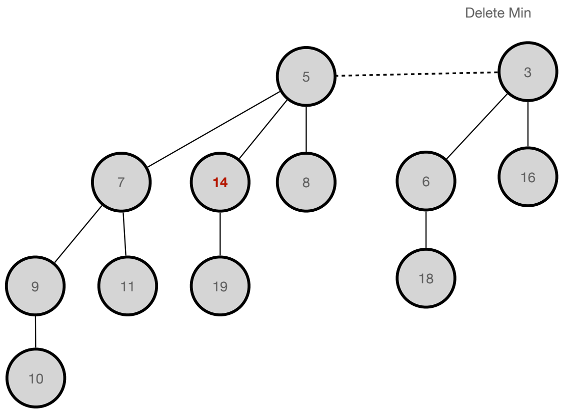
Delete 2: $[16, 14, 12, 13, 10, 8] \to [14, 13, 12, 8, 10]$

6. (6 pts) Consider the Fibonacci min-heap illustrated on the right. Show the Fibonacci heap that results after the following operation: Reduce key 17 to 14. Then, show the Fibonacci heap that results when you delete minimum in the resulting heap from the first part.

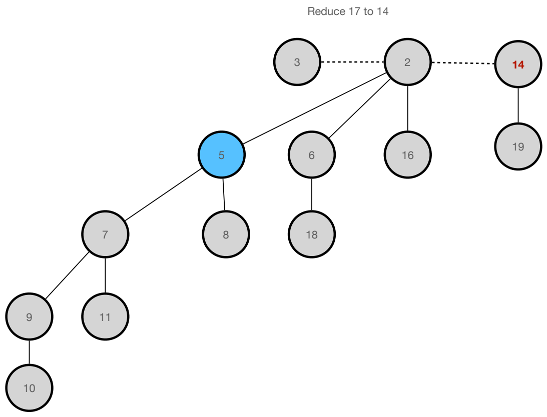


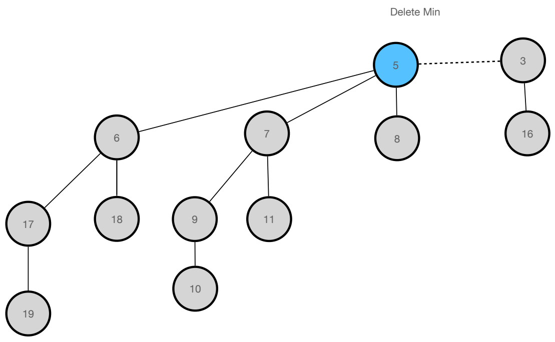
**Solution:** Here is one solution.



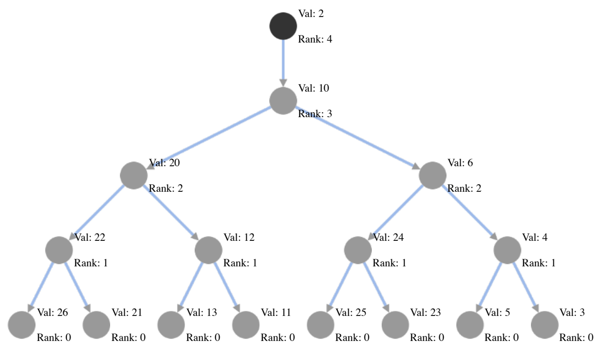


Here is another, depending on interpretation of how to reduce key.

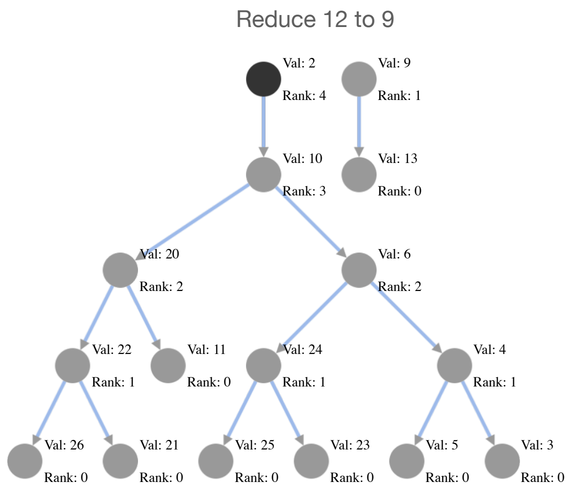


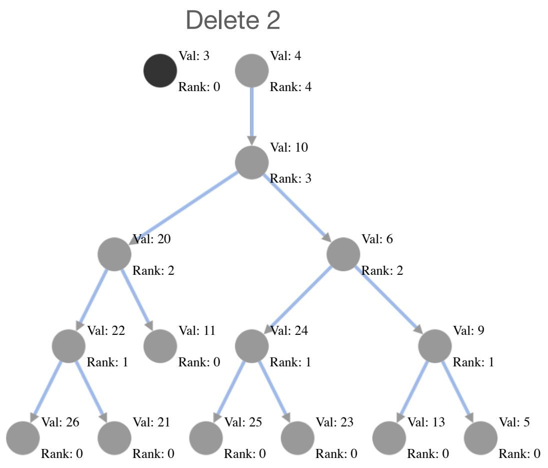


7. (6 pts) Consider the rank-pairing min-heap illustrated on the right. Show the rank-pairing heap, including ranks, that results after the following operation: Reduce key 12 to 9. Then, show the rank-pairing heap that results when you delete minimum in the resulting heap from the first part, using recursive merging, showing the resulting ranks.



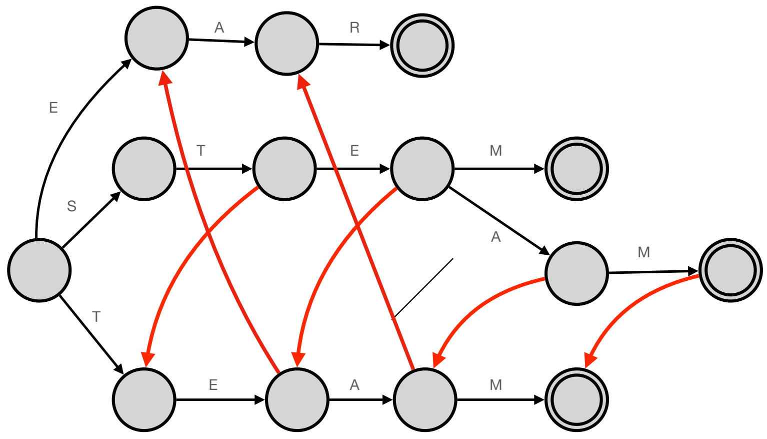
**Solution:**

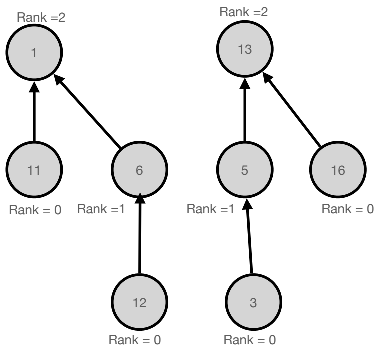




8. (6 pts) Draw the Aho-Corasick trie corresponding to the following search patterns: TEAM, STEAM, STEM, EAR. Show the suffix links, except for the suffix links that revert back to the root. Don't bother showing the output links.

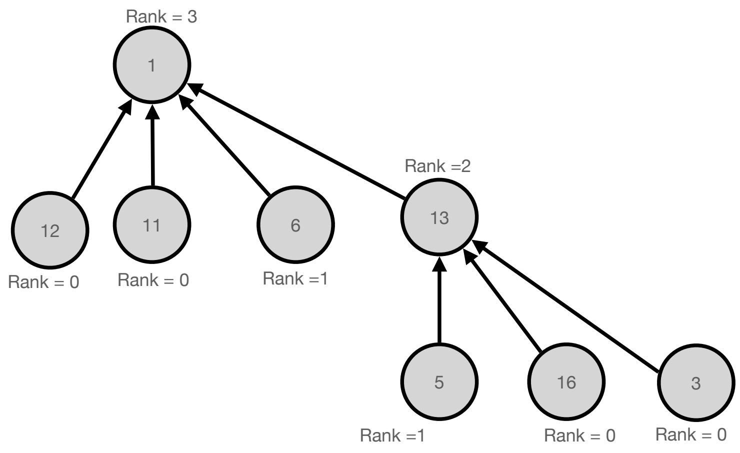
**Solution:**





9. (6 pts) Consider the two trees, shown on the right, that are part of a disjoint set forest. Suppose we find a relation between keys 12 and 3. Show the disjoint set tree that results from the union operation using merge by rank and path compression, where, if two roots have the same rank, merge the larger value root under the smaller value root.

**Solution:**



```
          6
         /
        5
       /
      4
     /
    3
   /
  2
 /
1
```

10. (6 pts) Consider the splay tree shown on the right, arising from inserting the numbers 1,2,3,4,5,6 in sequence.
Show the splay tree that results when you find the key 2 (and splay it to the top.)

**Solution:** First, we do a double rotation from 2.

```
        6              6
       /              /
      5              5
     /              /
    3              2
   / \            / \
  2   4          1   3
 /                    \
1                      4
```

Then we do a second double rotation from 2 to get it to the root.

```
   5                 2
 /  \              /   \
2    6            1     5
```

```
         / \                      / \
        1   3                    3   6
             \                    \
              4                    4
```

11. (6 pts) Consider the B+ tree of order 3, shown on the right. Show the two B+ trees that result when you delete 40 first, then when you delete 45 subsequently.

```
               40
           /       \
         25          45
       /   \       /   \
   15,20 25,30 40   45,50
```

**Solution:**

```
del 40  -->  Must borrow one key from sibling, update navigation.


            40
         /      \
       25         50
     /   \      /   \
 15,20 25,30 40   45,50

            45
         /      \
       25         50
     /   \      /   \
 15,20 25,30 45   50

Del 45 leads to underflow -- must fix.


         25,50
      /    |    \
  15,20 25,30  50
```

12. (6 pts) Consider inserting the keys 2, 6, 5, 3, 8, 9 in this order into the initially empty data structures below. Identify for which of the following structures is the node with the key 6 a child of the node with the key 2.

(a) A binary search tree.

(b) A red-black tree.

(c) A B+ tree of order 3 (maximum of 3 children).

(d) A binomial min-heap.

(e) A splay tree.

(f) A Fibonacci min-heap.

**Solution:** Note a Fibonacci min-heap will have all the keys flat in a linked list, so it won't work. Also note that in a binary search tree, 2 is the root and 6 is the next node inserted, so it will be a child of the root. The other structures require a little analysis.

In a binomial heap, 6 becomes a child of 2 after the 2nd insert, and since there are no deletions, it remains a child of 2 throught the other inserts.

Now, for red-black trees, after inserting 2, 6, 5, the node 5 rotates to the root, and 2 goes left of the root while 6 goes right of the root. It is easy to see that the next 3 inserts now cannot rotate 2 to reconnect as the parent of 6. Similarly, for the B+-tree of order 3, note that 2 can never be in the parent node of the node containing 6, because it is never a "middle" key to be promoted. It is the smallest key.

Finally, for a splay tree, note that 6 becomes the parent of 2 on the 2nd insert. Inserting 5 splits 2 and 6 to the left and right of 5 after splay. Inserting 3 puts 3 between 2 and 5, and no rotation gets 2 as the parent of 6.

So, it is true only for binomial min-heap and binary search tree.

13. (6 pts) For each of these recursions, please give the tightest upper bound for the recursion. You can write your answer as $T(n) \in O(g(n))$ for your best choice of function $g(n)$.

   (a) $T(n) = 5T(n/2) + (n \log n)^2$
   **Solution:** $(n \log n)^2 \in O(n^{log_2(5)-\epsilon})$ for some $\epsilon > 0$, so $T(n) \in \Theta(n^{\log_2(5)})$.

   (b) $T(n+1) = 2n + T(n)$, with $T(1) = 1$.
   **Solution:**
   $T(n) = 1 + 2 + 4 + 6 + \ldots + 2(n-1) = 1 + (n)(n-1) \in \Theta(n^2)$.

   (c) $T(n) = 2T(n^{1/2}) + 1; T(1) = 1$.
   **Solution:** $T(2^k) = 2T(2^{k/2}) + 1$

$$T'(k) = 2T'(k/2) + 1 \rightarrow T'(k) \in O(k) \quad \text{so } T(n) \in \log(n)$$

14. (6 pts) Given two Red-Black search trees, each having $n$, describe and estimate the asymptotic complexity of an algorithm for merging these trees to form another Red-Black binary tree containing $2n$ elements. For full credit, describe an $O(n)$ algorithm. Any slower algorithm will only receive at most half credit.

   **Solution:**

   To do it in $O(n)$, we do the following: Perform inorder on both the trees to obtain two sorted arrays of length $n$. Each of these is $\Theta(n)$. Do a merge operation as in mergeSort, which is $O(n)$, and obtain a merged, sorted list of length 2n. Map this final array to a red-black tree by selecting as the root the middle element (position $n$), the left and right children of the root as the midpoints of the [0..n-1] half and [n+1...2n] half. Recursively adding middle elements as the root and repeating the same process for left and right subarrays. The end result is almost height-balanced, and easily satisfies the red-black properties.