EC 504
Spring, 2022
Exam 1

Wednesday, March 16, 2022

Last Name	First Name	Student ID #

Honor Code: This exam represents only my own work. I did not give or receive help.

Signature: ____________________________________

Instructions

- Put your name on each page; and both your name and BU ID on the first page.
- This exam allows you to use 10 double sided pages of notes. No consultation with anyone other than the instructors are permitted during the exam.
- You have one hour and 50 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.

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 rational explanation is likely to earn partial credit, and no explanation will not earn any credit if the answer is wrong.

   (a) $n^{0.5} \in O((0.9)^n)$

   Solution: False. Note that 0.9^n goes to 0 as n increases!

   (b) We have a binary min-heap with n elements and wish to insert an array with n more elements into the heap (not necessarily one at a time) to form a min-heap with $2n$ elements. The total time required for this is $\Theta(n \log(n))$.

   Solution: False. Just put the two arrays back to back, and build the heap. $\Theta(n)$.

   (c) The tree on the right is an example of a tree that can occur in a binomial min-heap:

```
            5
      _____|__
     /     |    \
   10      8     7
  /  \
12    14
/
13
```

   Solution: False. it is not a binomial tree because it has two children of order 0, no child of order 1.

   (d) The tree on the right is an example of a tree that can occur in a Fibonacci min-heap by reducing keys in a binomial tree of order 3.

```
      5
     /  |
   10   6
  /
12
/
13
```

   Solution: True. Start from a binomial tree of order 3 as shown below:

```
                      5
         -------------|-------
        /             |       |
       10             6      23
    /   |             |
  12   14             7
 /
13
```

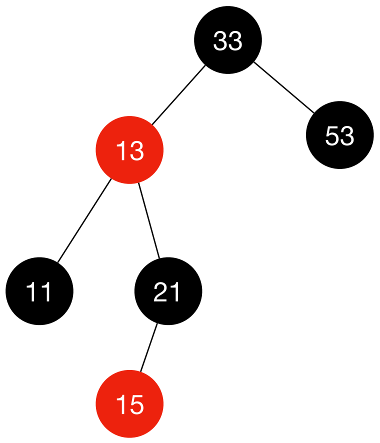
Now, reduce keys of 23 to 0, 7 to 1, 14 to 2.

(e) Consider a B+-tree in which the maximum number of keys in a non-leaf node is 9. Then, the minimum number of keys in any non-root, non-leaf node is 5.
Solution: False. Max keys = 9 means 10 children, so minimum number of children is 5, which corresponds to 4 keys.

(f) In the KMP algorithm for string matching, the prefix function for the string "AAABBBA" is [0,1,2,0,1,2,1].
Solution: False. It is [0,1,2,0,0,0,1].

2. (16 pts) Quick questions

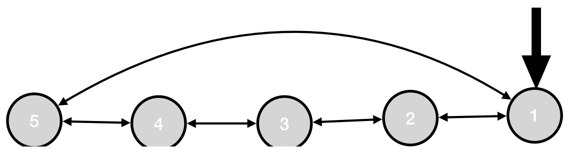
(a) The height of a tree is the length of the longest root-to-leaf path in it, in terms of number of edges in the path. What is the minimum number of nodes in a red-black tree of height 3?
Solution: The minimum number of nodes is 6, illustrated by this tree with black height 2.



(b) Consider a complete binary tree with n elements, stored in an array, where the left and the right subtrees of the root are ordered as max-heaps. Describe an algorithm with worst case complexity $\Theta(n)$ for converting this tree to a **min-heap** ?
Solution: This is what you get when you do build a heap using heapify, which is $\Theta(n)$.

(c) Consider two binomial heaps H_1, H_2, where H_1 has binomial trees of rank 0, 1, 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 2, 4, 5. The two rank 0 trees will merge into a rank 1 tree, which merges with the rank 1 tree to form a rank 2 tree. 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.

(d) Insert the following keys into an empty Fibonacci heap and show the final Fibonacci heap that results: 1,2,3, 4, 5.
Solution: See figure below.

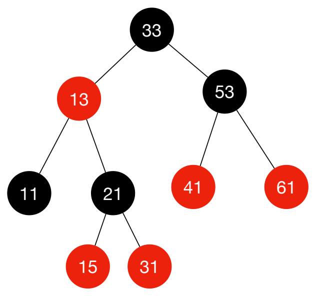


3. (6 pts) Given a binary tree (not a binary search tree), **pre-order** traversal of this tree yields the following sequence: 8, 5, 2, 6, 12, 9. Draw a complete binary tree of height 2 consistent with this traversal.

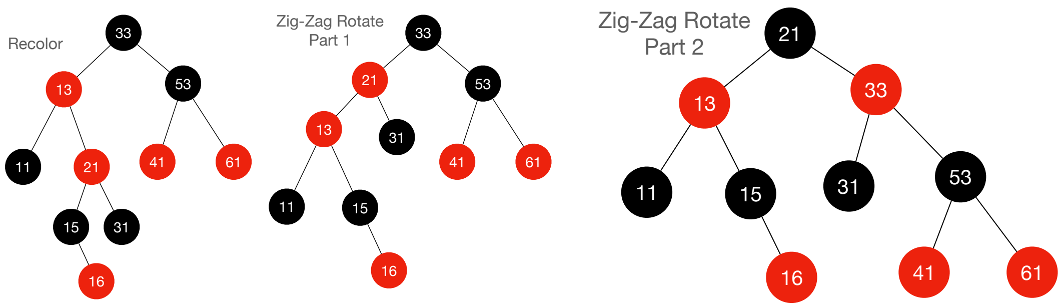
Solution:

```
         8
      /     \
     5       12
   /  \     /
  2    6   9
```

4. (6 pts) Consider the red-black tree shown on the right. Show the red-black tree that results when you insert the key 16 into the tree and restore the red-black property.



Solution: The solution is illustrated below. It involves a recolor, then a zig-zag rotation.



5. (6 pts) A hash table of length 10 uses open addressing with hash function h(k)=k mod 10, and linear probing. Insert the following keys into the hash table using linear probing in this order: 42, 46, 33, 23, 34, 52.

Solution:

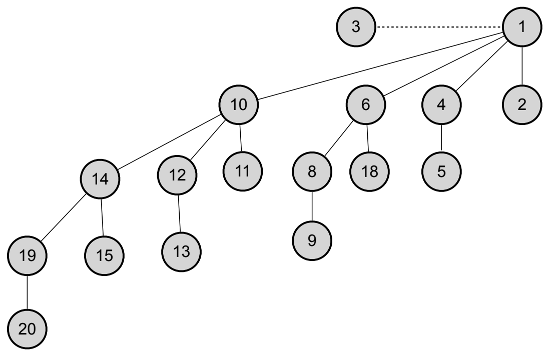
```
Slot position:   0   1   2   3   4   5   6   7   8   9
                        42  33  23  34  46  52
```

6. (6 pts) A binary min-heap implemented using an array is given as $[8, 10, 16, 19, 18, 17, 20]$. What is the content of the array after two delete min operations?

Solution: Delete 1: $[8, 10, 16, 19, 18, 17, 20] \rightarrow [10, 18, 16, 19, 20, 17]$

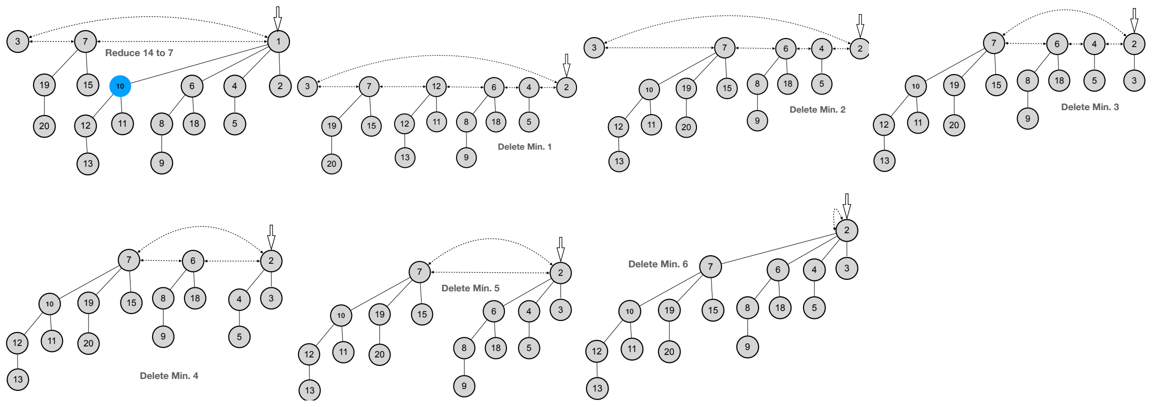
Delete 2: $[10, 18, 16, 19, 20, 17] \rightarrow [16, 18, 17, 19, 20]$

7. (6 pts) Consider the Fibonacci min-heap illustrated on the right. Show the Fibonacci heap that results after the following operation: Reduce key 14 to 7. Then, show the Fibonacci heap that results when you delete minimum in the resulting heap from the first part. Note that you may get different results depending on the merge order you choose. Any reasonable merge order will be fine, so don't worry about it. We don't grade this using a compute program...

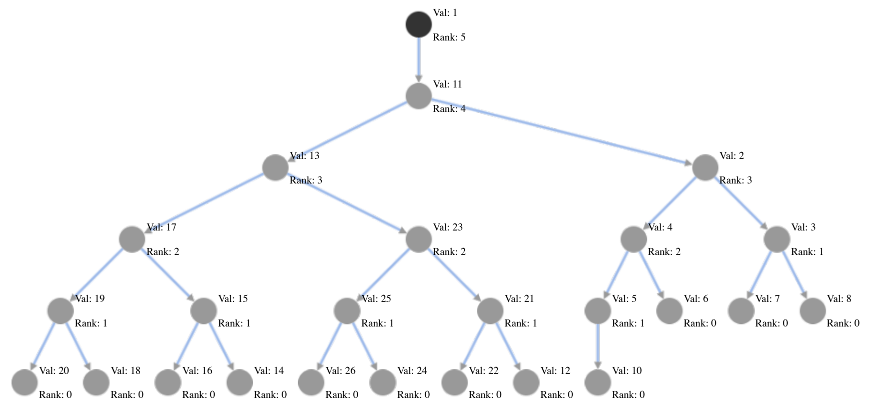


Solution:

The solutions are illustrated below. Decreasing the key from 14 to 7, and then deleting the minimum

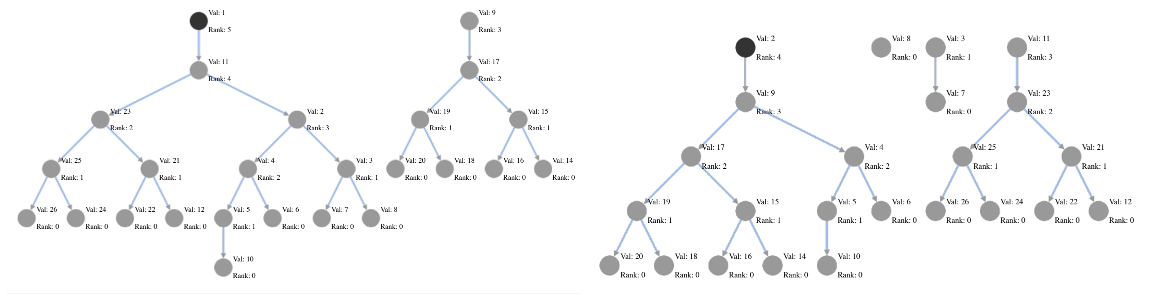


8. (6 pts) Consider the rank-pairing min-heap illustrated on the right. Show the rank-pairing heap, including ranks, that results after the following operation: Reduce key 13 to 9. Then, show the rank-pairing heap that results when you delete minimum in the resulting heap from the first part, using recursive merging, showing the resulting ranks. Note that you may get different results depending on the merge order you choose. Any reasonable merge order will be fine, so don't worry about it.



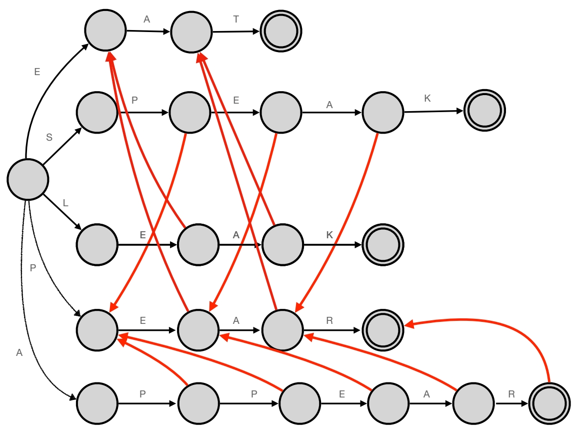
Solution:

The solutions are illustrated below. Decreasing the key from 13 to 9, and then deleting the minimum.



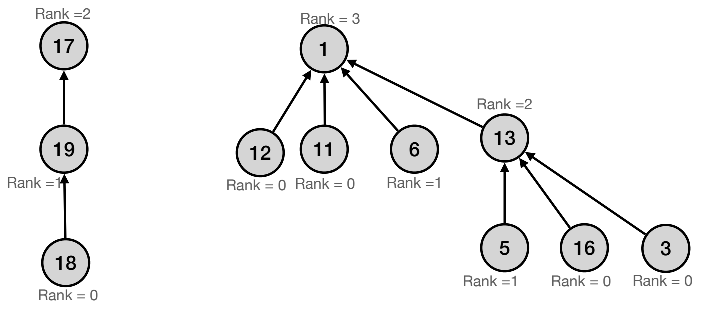
9. Draw the Aho-Corasick trie corresponding to the following search patterns: SPEAK, LEAK, EAT, PEAR, APPEAR. Show the suffix links, except for the suffix links that revert back to the root. Don't bother showing the output links.

Solution:

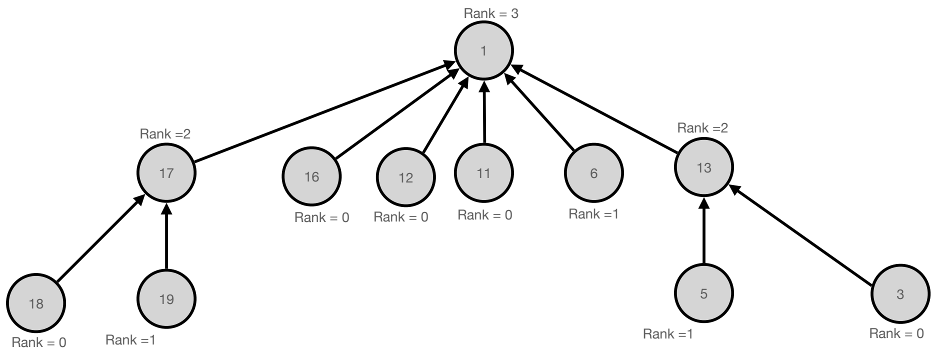


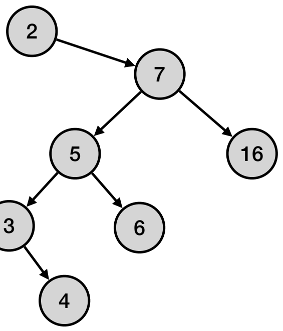
The trie is shown on the right:

10. (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 16 and 18. 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.



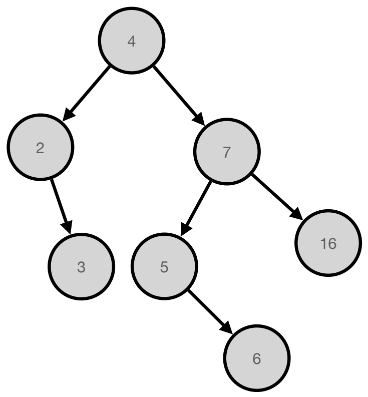
Solution: The merged trees are shown below, using merge-by-rank and path compression.





11. (6 pts) Consider the splay tree shown on the right. Show the splay tree that results when you find the key 4 (and splay it to the top using bottom up splaying.)

Solution:



12. (6 pts) Consider the B+ tree of order 3, shown on the right. Show the two B+ trees that result when you delete 9 first, then when you delete 8 subsequently.

```
               7
          /          \
      3,5              9
    /   |   \         /    \
  1   3    5,6    7,8    9
```

Solution:

```
Delete 9: borrow from sibling

             7
        /       \
      3,5         8
    /  |  \      /  \
   1   3  5,6   7    8

Del 8 leads to underflow -- must fix.
            5
          /    \
         3      7
       / \    /  \
      1   3  5,6  7
```

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) $T(n) = 8T(n/2) + n^3$
   Solution: $\log_2(8) = 3$, so the natural growth is n^3, which is also the order of the forcing function. By the Master theorem, $T(n) \in \Theta(n^3 \log(n) =) \in O(n^3 \log(n))$.

   (b) $T(n) = 3T(n/5) + \log^2(n)$.
   Solution: The natural growth is $n^{\log_5(3)}$ is much larger than the forcing function $\log(n^2)$, so the answer is $T(n) \in O(n^{\log_5(3)})$.

   (c) $T(n) = T(n-1) + 2^n$.
   Solution: Note that $T(n) = \sum_{k=1}^{n} 2^k$. This is like the integral of an exponential, so $T(n) \in O(2^n)$. Note that you can actually sum this, as

$$\sum_{k=1}^{n} 2^k = 2^n(\sum_{k=1}^{n} (\frac{1}{2})^{k-1}) = 2^n \cdot 2 \cdot [1 - (\frac{1}{2})^n] \in \Theta(2^n)$$

14. (6 pts) Consider a data structure using a sequence of arrays $A_0, A_1, A_2, \dots$ where the i-th array has 2^i elements. New elements are inserted first into A_0, and if A_0 is full, its contents are first moved onto A_1 before inserting into A_0. If A_1 is full, the its contents are first moved onto A_2. In general, if A_i is full, and we try to move elements into it, we first move the contents of A_i into A_{i+1} to make room to add elements before we add elements to A_i.

   (a) Assume initially you have an empty data structure. You now insert 128 elements. What is the index of the largest array that will contain elements?

   Solution: The maximum number of elements in arrays 0, 1, 2, 3, 4, 5 and 6 is $1 + 2 + 4 + 8 + 16+32+64 = 127$. Hence, 128 will require elements in A_7, so it will require 8 arrays. The index of the largest array with elelements is 7.

   (b) What is the amortized cost of an individual insertion assuming the data structure starts empty, as the number of insertions n grows? Hint: you can do this by looking at the maximum number of data movements times the number of entries that move that much.

   Solution: For any n, roughly $n/2$ of the elements will be in the array $A_{\log n}$. Each of these elements will have moved $\log n$ times. Hence, n inserts will require at least $n \log n/2$ data movements, so n inserts is in $\Omega(n \log n)$. Since the maximum amount of movement by any element is $\log n$, the index of the largest array with elements, then the maximum amount of work for n elements is $O(n \log n)$. Hence, the amortized cost per insert is $\Theta(\log n)$.

   (c) Suppose each array A_i holds i elements now instead of 2^i. What is the amortized cost of an insertion now, as n grows?

   Solution: Doing a similar computation, for arrays up to k, the maximum number of elements in the arrays is $k(k+1)/2$ (summing the capacities of all the arrays up to index k. Hence, for n elements, the last array with elements has $O(\sqrt{n})$ elements, each of which has moved $\sqrt{n}$ times. Elements in array k will have moved k steps, and array k will have k elements in it.

   The total amount of movement is

$$\sum_{k=1}^{\sqrt{n}} k^2 \in \Theta(n^{3/2})$$

   using an integral approximation. Hence, the amortized cost of n insertions is $\sqrt{n}$.