CS3340 Assignment 2 Solutions
Bradley Assaly-Nesrallah (250779140)

1. 8.2-1

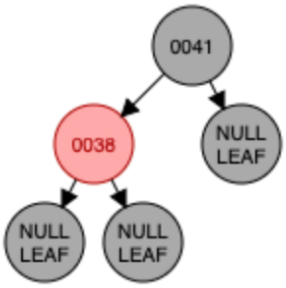
We have an array A = [6,0,2,0,1,3,4,6,1,3,2], and perform the operation counting-sort upon A. We have that the auxiliary array C = [2,4,6,8,9,9,11] after line 8. Then, the loop on lines 10-12 generates the array B = [, , , , ,2, , , ,], then B = [, , , , , 2 , , 3, , ,], then B = [, , , 1 , , 2 , , 3, , ,], and continues until we have the final array B = [0,0,1,1,2,2,3,3,4,6,6]. Which is the final output of counting sort upon the input array so we are done.

2. 13.3-2

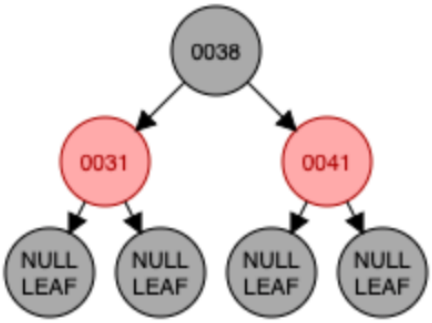
We generate a red black tree by successively inserting the following keys into an initially empty red black tree, 41, 38, 31, 12, 19, 8, here is the resulting output:

Insert 41:

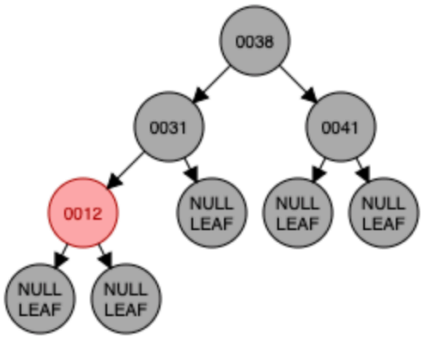
Insert 38:



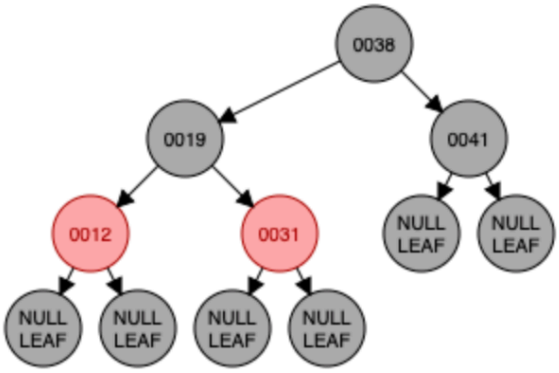
Insert 31:



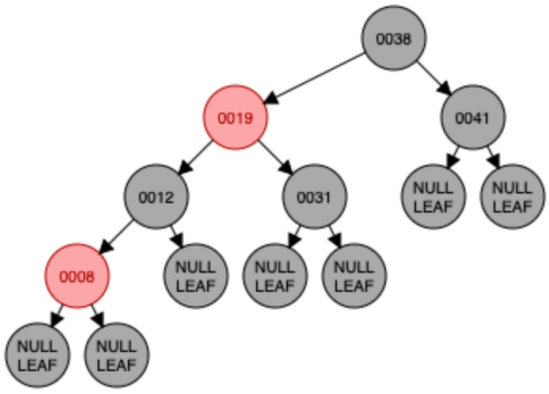
Insert 12:



Insert 19:

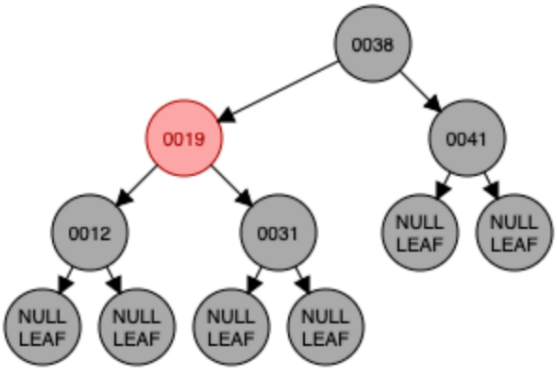


Insert 8:

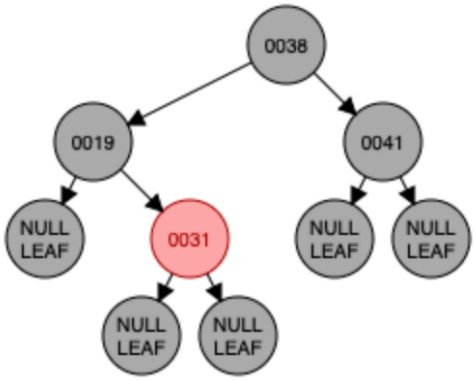


3. 13.4-3
We use the red black tree obtained at the end of question 2 and show the trees that are obtained from successive deletion of the keys in the order 8, 12, 19, 31, 38, 41:

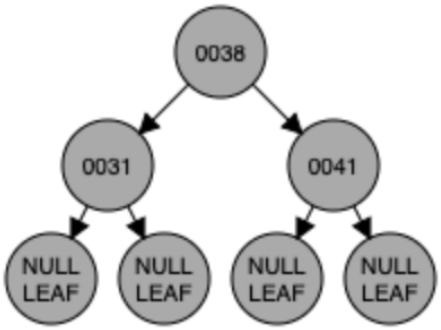
Delete 8:



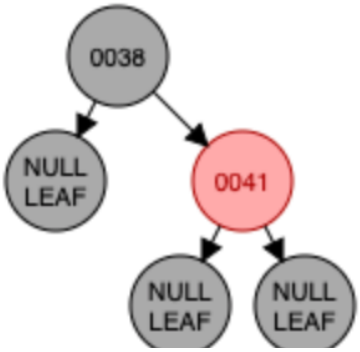
Delete 12:



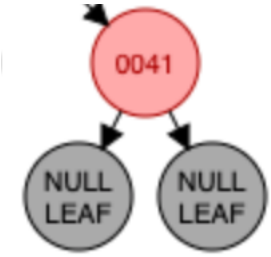
Delete 19:



Delete 31:



Delete 38:



Delete 41:

4. We have k sorted sequences which contain n elements and must design an efficient algorithm to sort all kn elements into one sorted sequence. Firstly, we shall describe the algorithm in english, secondly, show that the algorithm is correct, and thirdly analyze the time complexity of the algorithm. The algorithm is to first merge the k arrays of size n into groups of 2, then we have k/2 arrays left where each new array is sorted. Then continue to merge these groups until there is only one array left, which is the final sorted sequence.

The algorithm is correct since we know that for k sorted arrays of size n. Each step we merge the following array is sorted and size 2n, since we are merging two arrays of size n. There are k/2 total merging, hence after log k iterations we will be done. Since each iteration of the algorithm all of the new merged arrays are sorted, the final solution must be sorted with all kn elements in one sorted sequence. So the algorithm must be correct will have the correct output, a sorted array with all nk elements in all cases, so we are done.

Now we analyze the time complexity of the algorithm. We have k sorted arrays of size n. For the merging, since each iteration would take 2n time, for merging two arrays of size n. There are a total of k/2 merging, so total time in the first iteration would be O(nk). Since there are

O(logk) iterations since we have k arrays and each time we are halting the total number of arrays. Hence the total time complexity of the algorithm is O(nkLogk) so we are done.

5. 16.3-6

We have an optimal prefix code on a set C = {0,1,…,n-1} of characters and wish to transmit this code using as few bits as possible, we will show how to represent any optimal prefix structure using only 2n-1 +n[lgn] bits:

We will be using a binary tree data structure, note that any full binary tree has exactly 2n-1 nodes. We may encode the binary tree using a preorder traversal of a tree T. For mach node we record in the traversal, write 0 if it is an internal node and 1 if it is a leaf node. Since the tree is full, the structure will be unique. Next, we encode any character of C in $\lceil lgn \rceil$ bits. Since we have n characters we will encode them in order of appearance in our preorder traversal using n$\lceil lgn \rceil$] bits. So we are done.v=

6. 21.4-2

We must prove that every node has rank at most $\lfloor lgn \rfloor$. We will proceed by strong induction upon the number of nodes. For the base case if n = 1, then that node has rank equal to 0 = $\lfloor lg1 \rfloor$, and we suppose that the claim holds for n = 1, 2, …., n nodes. For n + 1 nodes, suppose we perform the union operation on two disjoint sets with I and j nodes respectively, where I,j <= n. Thus the root of the first set has at most $\lfloor lga \rfloor$ and the root of the second set has rank at most $\lfloor lgb \rfloor$.

Now consider when the ranks are not equal, then the union operation preserves rank and so we are done, so suppose that the ranks are the same. Then the rank of the union increases by one, so the resulting set has rank $\lfloor lga \rfloor$+1 <= $\lfloor lg(n+1)/2 \rfloor + 1 = \lfloor lg(n+1) \rfloor$. Therefore by induction every node has rank at most $\lfloor lgn \rfloor$, so we are done.

7. 21.4-3

Using the result of exercise 6 we must determine how many bits are necessary to store x.rank for each node x. We know that their value is at most $\lfloor lgn \rfloor$, so we may represent them using $\Theta(lg(lg(n)))$ bits, and will need to use that many bits to represent a number that can take that many values. So we are done.

8. A union find algorithm has been implemented in java