Problem Set 2

Name: Neeraj Pandey

Collaborators: Tanuj Sood

Problem 2-1.

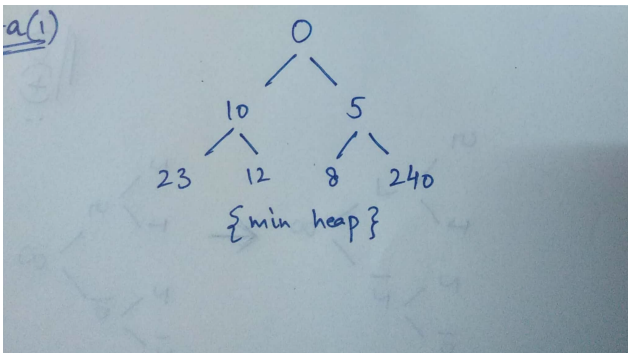
(a) Here, we assume that for A_0 and A_1 there is one endpoint in the array $e_i = max[A_i]$ and for the smaller endpoint (e_0), we have $e_0 = max(A_i[x_0 : \frac{n}{2} + 1])$. For the large endpoint, (e_1), we have $e_1 = max(A_i[\frac{n}{2} : x_1 + 1])$. Further, if we have an integer value greater than e_0 or e_i in any of the array, that value will be the endpoint. Therefore, we have $e_i = max[A_i]$

(b) Using the divide and conquer algorithm, the first step is to split the array into 2 parts as A_0 and A_1. In the above problem, we had an assumption that each of the endpoints lies in each half. So, split each of the halved repeatedly(recursively) until there is only 1 element left in the split array. So, store the values for the largest value in each half. Further, the endpoints are going to be A_0 and A_1. Next step is to merge the arrays back into the original order, and find the length between the two stored index values which will give the maximum length of the possible zipline.

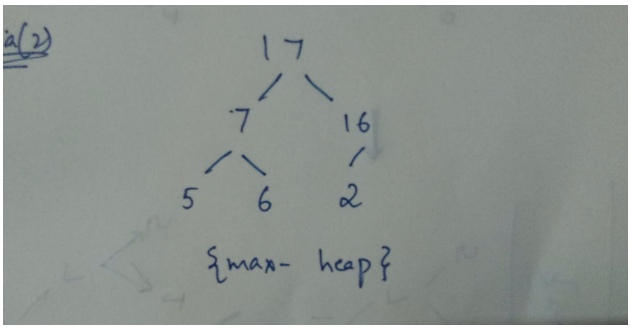
Problem 2-2.

(a)

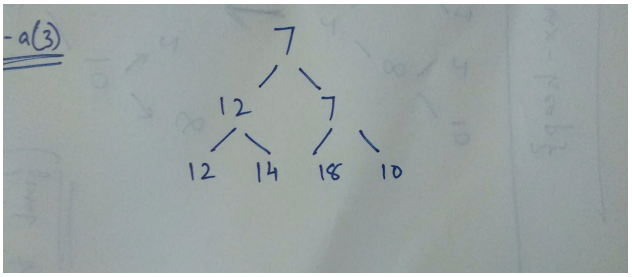
- (1) Min-heap



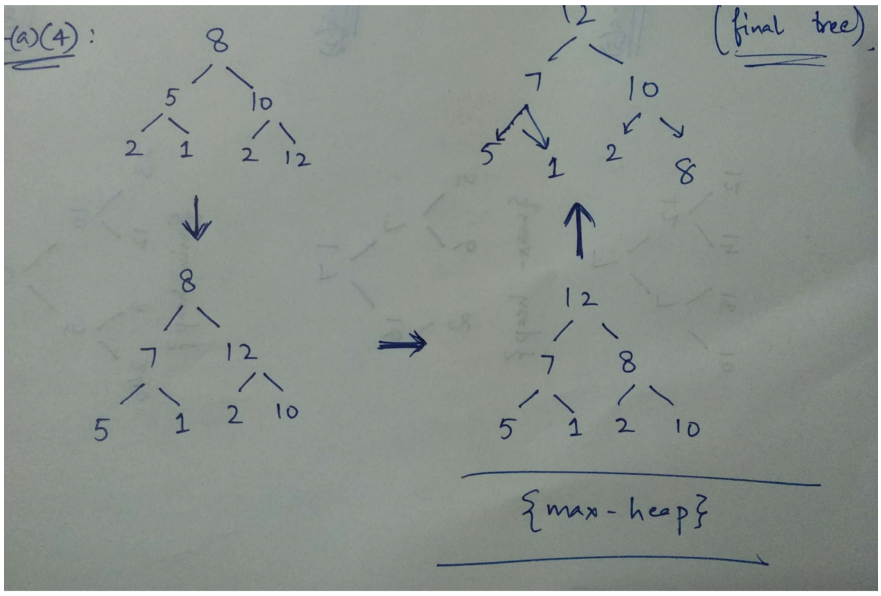
- (2) Max-heap



- (3) Min-Heap



- (4) Not Min-heap, Not Max-heap



(b) Given X = 6, 5, 3, 2, 6, 1, 4
As it is a max-heap, value of A,

$$A = \{6, 6\}$$

As A will be 6, B can take values of either 5 or 6

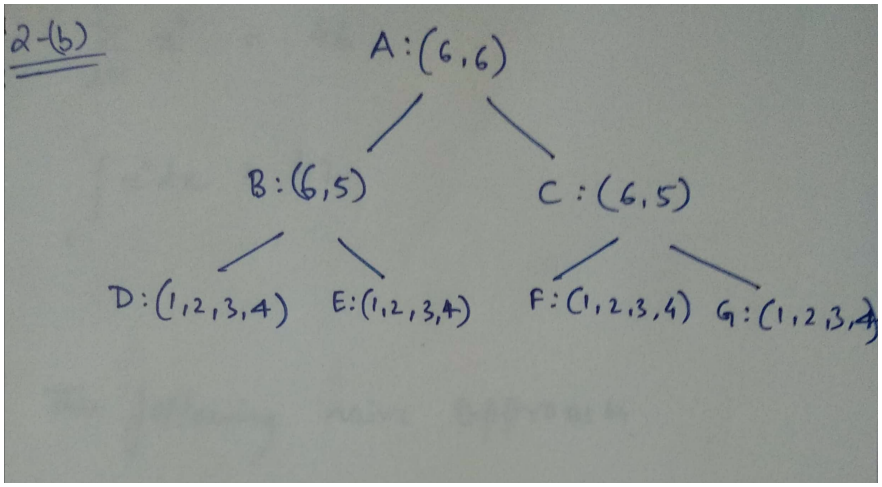
$$B = \{5, 6\}$$

$$D = B - \{1, 2, 3, 4\}$$

$$E = D - \{1, 2, , 3, 4\}$$

$$F = E - D - \{1, 2, 3, 4\}$$

$$G = E - D - F - \{1, 2, 3, 4\}$$



Problem 2-3.

(a) There are r TA's, so we have to merge r sorted lists for the instructor to check. Further, we don't know the number of TA's, if they are two or more, we can sort all the grades using the $k - way$ merge algorithm using the $min - heap$ as the grades are sorted and $r >= 2$. From the k list of students, create a $min - heap$ with a pointer before each element. We will have r elements in the min-heap, pick a single element from the r sorted lists. The root node of the $min - heap$ consists the smallest values, pick the value and add it to the final array. By picking the root element, we increment the pointer in the list and insert in the heap and this will take the time complexity of $O(logr)$. As we have removed an element from the heap and then added an element to it, we repeat this for s times for s students. This entire insertion and deletion of elements will take $O(2slogk)$ (removing the constant), we get $O(slogk)$ using the k-way merge technique and using the $min - heap$.

Problem 2-4.

(a) Priority Queue : Sorted array
Sorted array has $O(n)$ for inserting new element because you have compare n elements to sort in worst case. Sorted array has $O(1)$ to find max and min because it is sorted already and first and last will be min and max.

(b) Using priority queue to use insertion sort as it is already sorted and $record\_bowl$ will take $O(n)$ while $best\_bowl$ will take $O(k)$. We will create a heap and call all the elements of $record\_bowl$. We will heapify after every insertion as the insertion is not expected to be sorted. Now, take out k elements from the heap which will take $O(k)$. Further, on every iteration of $record\_bowl$, it will take $O(k + logn)$ as the time complexity is linear and depends on where it is within the heap. To take k elements out, it will be a linear $O(k)$ for $best\_bowl$.

(c) Submit your implementation online.