COP 5536 / AD 711R

Advanced Data Structures

Sample Solution - Exam 1

1.

(1) Aggregate Method

Assume that we performed *n* operations.

Each element which is put into the stack *S* by a *push(x, S)* operation can be removed ***at most once*** by either *pop(S)* or *multi-pop(k, S).*

Since there can be at most *n push(x, S)* operations, there can be at most *n pop(S)* operations (including counting the appropriate number of *pop(S)* operations for each *multi-pop(k, S)*). It means that the total time taken for the entire sequence is at most O(*n*).

Thus each operation takes O(1) time.

(2) Accounting Method

Guess the amortized costs for each operation as following:

Push(x, S) : 2

Pop(S) : 0

Multi-pop(k, S) : 0

For a push(x, S) operation, we charge 2 units of cost. One unit is used to pay for the cost of the push(x, S) operation and the other unit is assigned to the element as a credit.

For a pop(S) operation, we charge nothing. Instead the cost for the pop(S) operation is paid for by using the credit of one unit that was credited with the element.

For a multi-pop(k, S) operation, we charge nothing. The cost of removing each element can be paid for by using the credit stored with each element.

Each operation can be paid for and the total credit stored in the stack is never negative. Since each element has a credit of one unit while it is in the stack S, and there can never be a negative number of elements in S. The total charge for a sequence of *n* operations is an upper bound on the total cost for the sequence which is at most *2*n* (an element in the stack can be removed at most once either a *pop(S)* or *multi-pop(k, S)* operation but not both).

Thus, the total cost is O(n).

Dividing by the number of operations, n, gives an amortized complexity of O(1) for each operation.

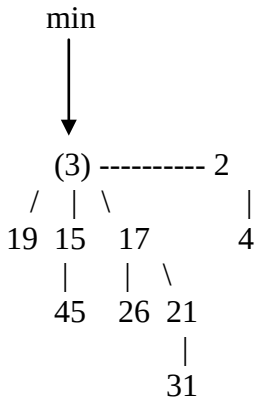
2

a)

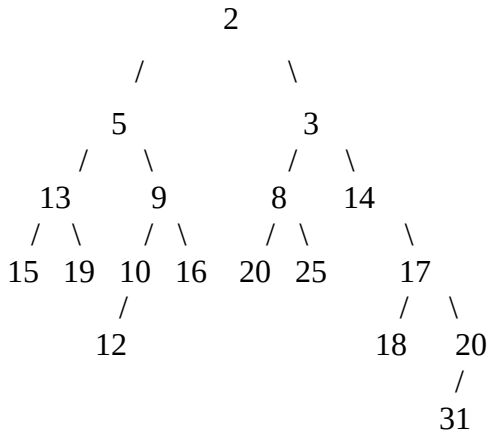
15 - 45 - 3 - 19 - 17 - 26 - 31 - 21 - 1 - 4 - 2

min

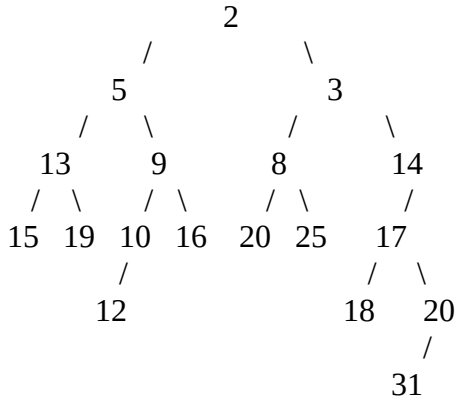
b)



3. After meld 2 and 3 sub-trees.

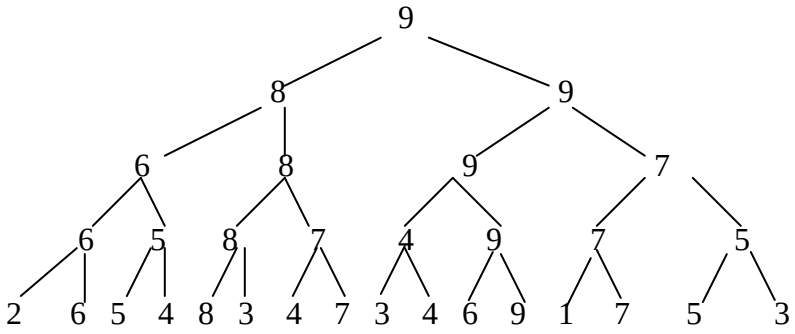


Swap left sub-tree and right sub-tree of 14



4)

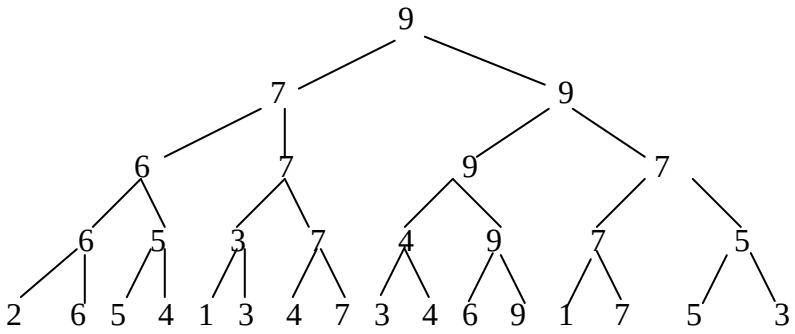
a)



b)

It takes time of the height of the tree to find a bin. Hence, it takes O(logN) time.

After allocating the bin for an item,



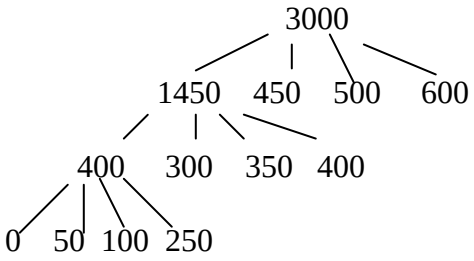
5)

a) q = (1-r) mod (k-1)

= -8 mod 3

= (9-8) mod 3

Hence q =1



b)

level 1 = (400/50*10 + (400/50)*20 + (400/50)*20 = 80 + 160 + 80 =320 sec

level 2 = (1450/50)*10 + (1450/50)*20 + (1450/50)*20 = 290 + 580 + 290 = 1160 sec

level 3 = (3000/50)*10 + (3000/50)*20 + (3000/50)*20 = 600 + 1200 + 600 = 2400 sec

Total time = 320 + 1160 + 2400 = 3880 sec