Spring 2002

EXAM 1 Solution

1.
Since we start with the queue Q being empty, delete and multi-delete would only remove those elements already in Q; i.e., those previously inserted. Thus, if we charge an extra cost to Insert, we can use that extra cost to cover the cost for deleting the element in the future.
Let there be i inserts in the sequence of n operations. With the cost of an insert being 2, and those of delete and multi-delete being 1 the total cost C = 2i+(n-i) = n+i. And i can not be greater than n. Thus C = n+i <= 2n = O(n).
Therefore, amortized costs for each operation:

```
insert(x,Q)       : 2
delete(Q)         : 0
multi-delete(k,Q) : 0
```

2.

(a)(1-n) mod (k-1) = -8 mod 3 = 1
Need one dummy run whose length is 0.

First merge 4 runs (0, 10, 20, 30) into one (60 records)

Second merge 4 runs(40, 50, 60, 60) into one(210 records)

And merge the rest and new one (70, 80, 90, 210) into one (450 records)

(b) Need two comparisons to generate one record, except for the first record of each merge step due to loser tree initialization. (3 comparisons to initialize the loser tree)

For the first merge pass, # of comparisons = 2*(60 -1) + 3 = 121

For the second merge pass, # of comparisons = 2*(210-1) + 3 = 421

For the third merge pass, # of comparisons = 2*(450-1) + 3 = 901

So, the total number of comparisons = 1443

(c) Due to the sequential processing,

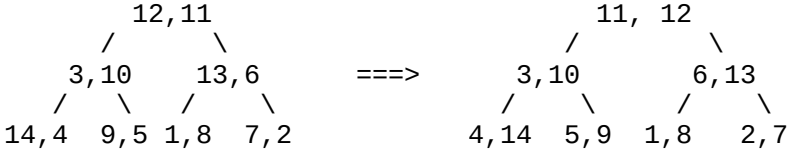
Merge time = Input time + Output time + CPU time = 2 * Input time + CPU time

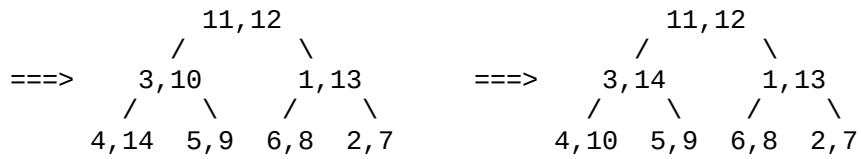
So, [2*(6*10) + 6] + [2*(21*10) + 21] + [2*(45*10) + 45] = 126 + 441 + 945 = 1512 (sec)

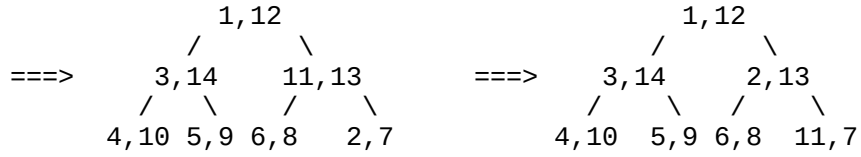
3.

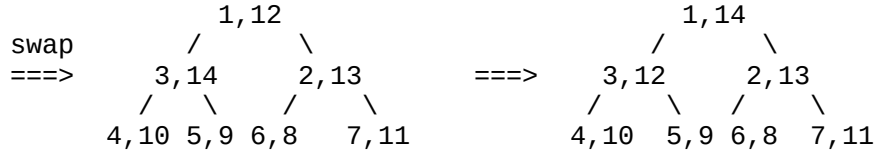
(a) First, arrange keys within each node.

(i) A sample solution (as in samples)

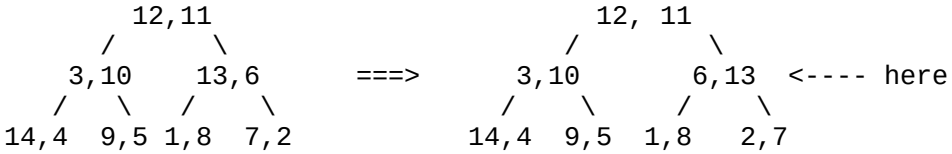


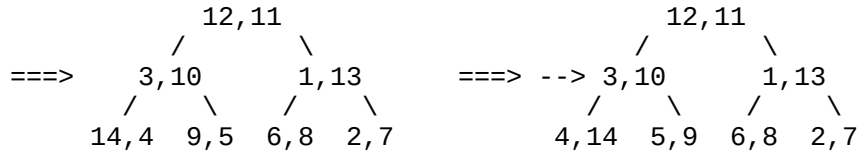


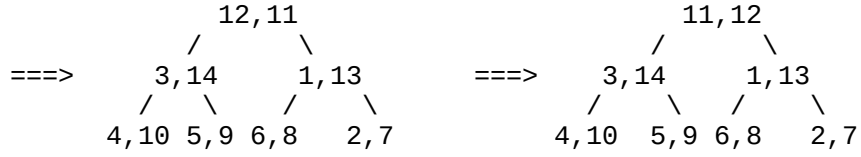


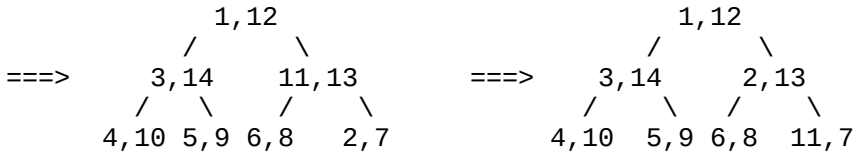


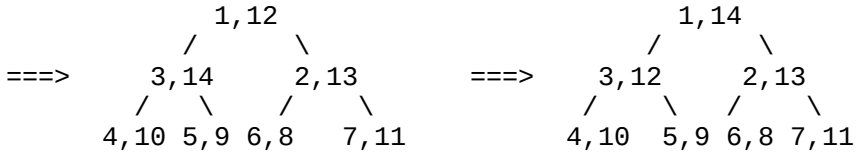
(ii) Another sample solution





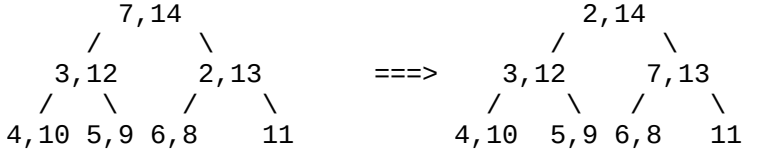


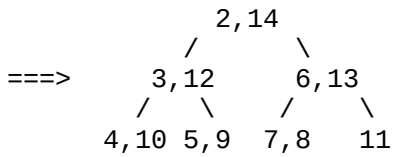




(b)

First, remove the min and insert the minimum key of the last node to the root.



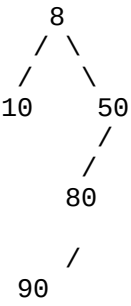


4.

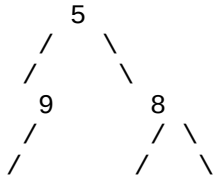
(a) already a min leftist tree

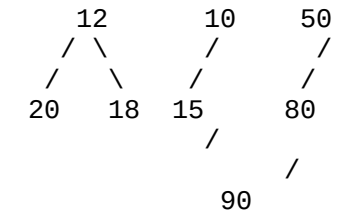
(b)

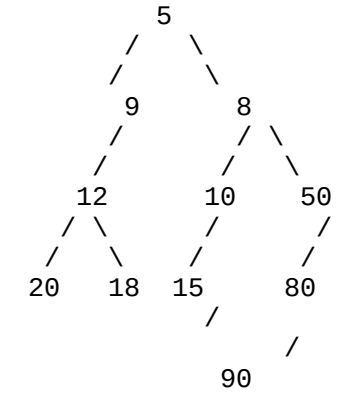
Step 1.



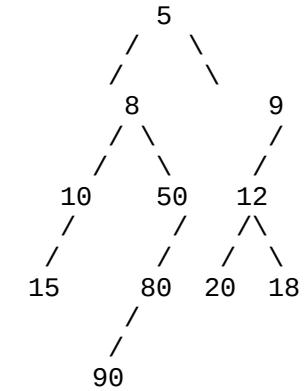
Step2



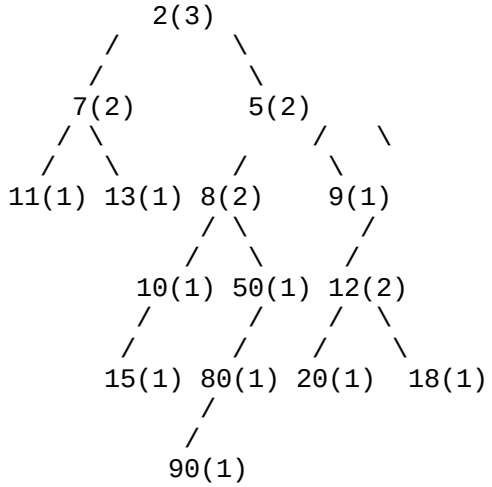




Step3 : swap the left subtree and right subtree

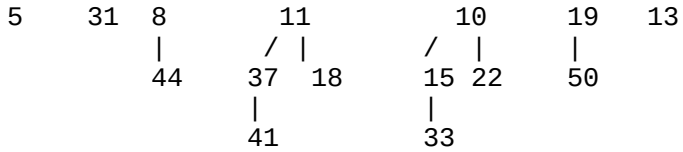


Step4 :

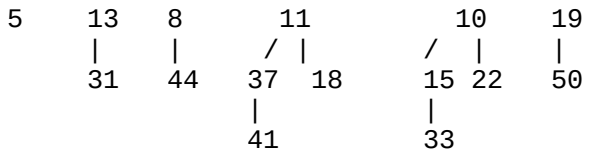


5.

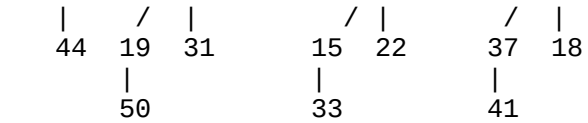
Step1 :
After Delete-Min



Pairwise combine after delete-Min
(Combine two heaps with degree 0)



(Combine two heaps with degree 1)



(combine two heaps with degree 2)