The Solutions of Homework # 4

Problem 1.

(a)

Solution:

W.l.o.g, we assume the height of the tree is at least 1. The node with the minimum is at the root and the node with the maximum is one of the children of the root.

Since every node at an even depth is larger than its grandparent, the root is smaller than the nodes at depth 2, the smallest node at depth 2 is smaller than the nodes at depth 4, and so on. Therefore, the root is smallest node among those of even depth. Also, a node at an odd depth is larger than its parent, it means it is at least larger than a node of even depth. Accordingly, the root is with the minimum priority in the tree. Similarly, the left and right subtree of the root are two max-min heaps and they are with the maximum priorities in the trees.

□

(b)

Solution:

deleteMin: The process can be divided into 9 steps as follows.

Step 1. Remove the root node.

Step 2. Move the last element of last level to root.

Step 3. Compare the value with its grandchildren's.

Step 4. If the value is more than the least value of its grandchildren, then swap them; otherwise goto step 8.

Step 5. Compare the value with its parent's.

Step 6. If the value is larger, then swap them and go back to the node(child).

Step 7. Repeat step 3 and step 6 until min-max heap property holds.

Step 8. Compare the value with its children's.

Step 9. If the value is more than the least value of its children, then swap them.

deleteMax The process can be divided into 9 steps as follows.

Step 1. Remove the largest child of the root.

Step 2. Move the last element of last level to the node just deleted.

Step 3. Compare the value with its grandchildren's.

Step 4. If the value is less than the most value of its grandchildren, then swap them; otherwise goto step 8.

Step 5. Compare the value with its parent's.

Step 6. If the value is smaller, then swap them and go back to the node(child).

Step 7. Repeat step 3 to step 6 until min-max heap property holds.

Step 8. Compare the value with its children's.

Step 9. If the value is less than the most value of its children, then swap them.

□

Problem 2.

Solution:

Refer to Figure 1 to Figure 3.

□

Problem 3.

Solution:

For each positive integer n, we would like to create a Fibonacci heap consisting of just one tree that is a linear chain of n nodes. To be more precise, the values of the nodes are all odd numbers. If $n = 1$, the case is trivial.

Now, let's consider how can we derive case $n = k + 1$ from case $n = k$. We have a Fibonacci heap H consisting of just one tree that is a linear chain of k nodes with values being odd numbers. Let m be the min of H. W.l.o.g, we assume $m > 3$. By doing the following 6 operations consecutively, we derive H' which satisfies the desired property.

Step 1. Insert $m + 1$.

Step 2. Insert $m - 2$.

Step 3. Insert $m - 3$.

Step 4. Delete the minimum.

Step 5. Decrease $m + 1$ to $m - 3$.

Step 6. Delete the minimum.

□

Problem 4.

(a)

Solution:

The resulting leftist heap is a full binary tree.

$k = 1, 2$ are trivial cases. Assume the statement is satisfied when $k = n$. Let H be a leftest heap and full binary tree with node values from 1 to $2^n - 1$. We insert $2^n, \ldots, 2^{n+1} - 1$ into H consecutively. Consider the insertion of 2^n. Since $2^n > i$ for each $i \leq 2^n - 1$, 2^n is to be a child of a node at the bottom level of the right subtree of the root of H. If $n > 2$, the null path lengths of the children of the root haven't changed. In other words, at the root level, there is no need to swap the left and right children. Accordingly, to insert $2^n, 2^n + 1, \ldots, 2^n + 2^{n-1} - 1$ into H is to insert these numbers into the right subtree. Because $k = n$ is true, the right subtree is a full binary tree after these insertions, the null path length of the right child of the root has just increased 1 after the insertion of $2^n + 2^{n-1} - 1$, and we swap the left and right children of the root. Therefore, after the insertions of $2^n, 2^n + 1, \ldots, 2^n + 2^{n-1} - 1$, we have a heap with a left subtree a full binary tree of size $2^n - 1$ and right subtree a full binary tree of size $2^{n-1} - 1$. The insertions of $2^n + 2^{n-1}, \ldots, 2^{n+1} - 1$ are similar.

□

(b)

Solution:

The the resulting leftist heap is a left-skewed tree.

To insert a node X into a leftist heap H is equivalent to merge the leftist heap with the node. X is also a leftist heap. The value of X is less than the value of any node of H and the left child of X is null, so H is the left child of the root X after the union of X and H.

□

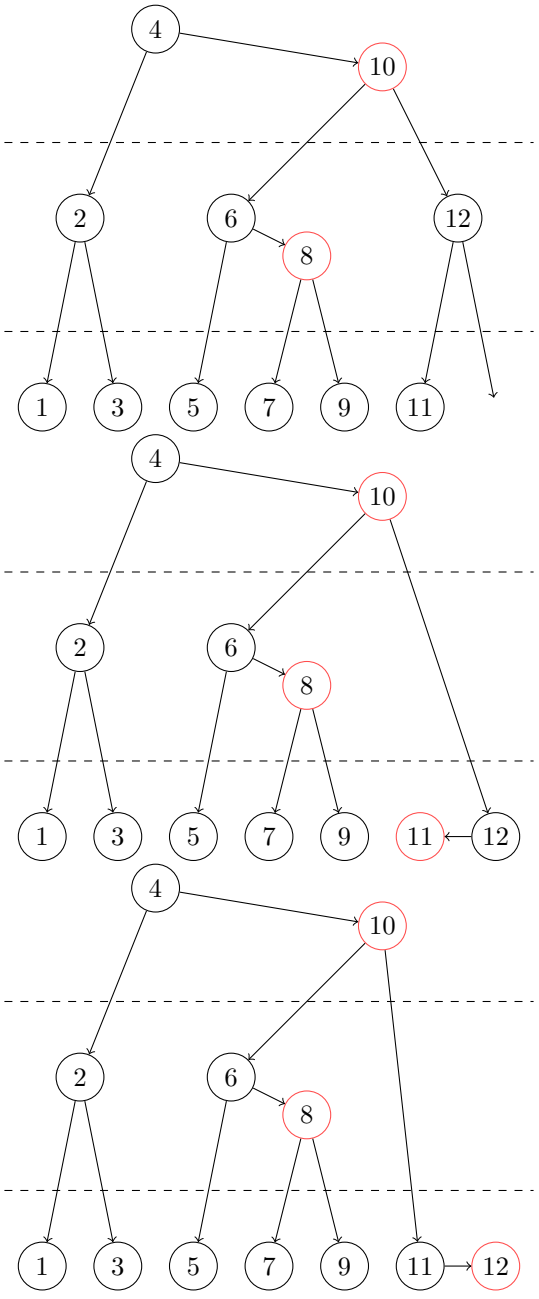

Figure 1:

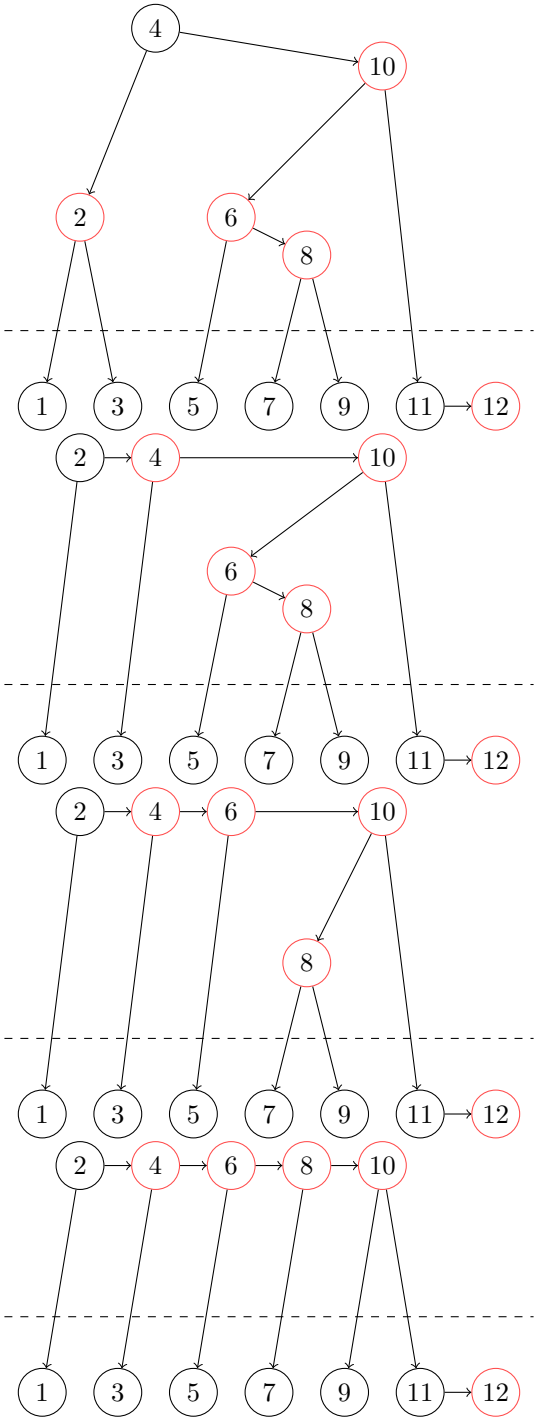
4
10
2
6
8
1
3
5
7
9
11
12
2
4
10
6
8
1
3
5
7
9
11
12
2
4
6
10
8
1
3
5
7
9
11
12
2
4
6
8
10
1
3
5
7
9
11
12

Figure 2:

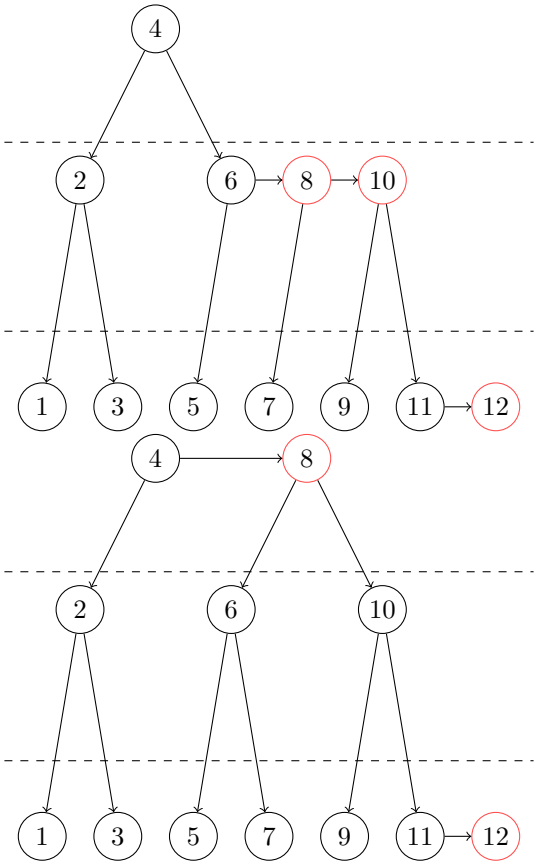

Figure 3: