University of Houston

COSC 3320: Algorithms and Data Structures
Spring 2016

Solutions for Homework 5

1. (a) Construct a heap containing the following values, inserted one after the other:

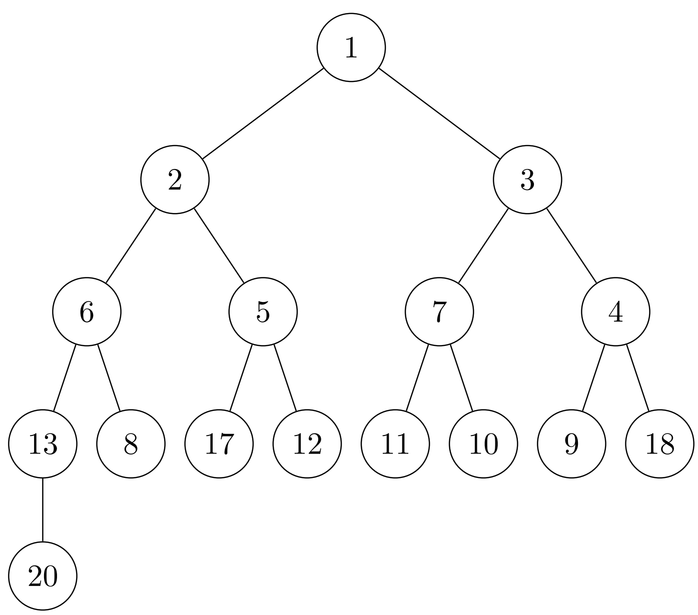
$$9, 20, 8, 17, 4, 11, 2, 1, 6, 12, 5, 10, 7, 3, 18, 13.$$

   You are to draw the final heap, both as a binary tree and as in its standard array implementation.

   (b) Give a (small) example of two distinct permutations of the same set of values such that the two heaps constructed by inserting one value after the other are different.

   Solution:

   (a) The final heap is as follows.



   With the standard array implementation: $1, 2, 3, 6, 5, 7, 4, 13, 8, 17, 12, 11, 10, 9, 18, 20$.

   (b) For example, $1, 2, 3$ and $1, 3, 2$.

2. Given a heap H and a value k, we wish to return all the values in H which are at most k. Let n be the size of H, and m, with $0 \leq m \leq n$, be the number of values to be returned. (Notice that m is unknown at the beginning of the algorithm.)

(a) Design a simple algorithm of complexity $O(1 + m \log n)$.

(b) Design an improved algorithm with complexity $O(1 + m)$. (Hint: you should not modify the heap. Rather, you should work directly on the array implementation of H.)

Solution:

(a)
```
if (H.min() > k) then return
else
  while (H.min() <= k) do
    print(H.removeMin())
```

If the root of H has value bigger than k, then there is nothing to return, and the algorithm has complexity $O(1)$. Otherwise the algorithm is removing m values, and each of such operations has cost $O(\log n)$.

(b)
```
if (H[1] > k) then return
else Find(H,1,k)
```

where `Find(H,1,k)` is the following recursive algorithm

```
Find(H,i,k)
input: Heap H, index 1 <= i <= n, value k
output: All the values which are at most k in the subtree rooted at T[i]
print T[i]
if ((2i <= n) AND (T[2i] <= k)) then Find(T,2i,k)
if ((2i+1 <= n) AND (T[2i+1] <= k)) then Find(T,2i+1,k)
```

If the root of H has value bigger than k, then there is nothing to return, and the algorithm has complexity $O(1)$. Otherwise the complexity of the algorithm is that of a preorder visit of H', the subtree of H whose nodes have values at most k. The size of H' is m, and hence in this case the complexity of the algorithm is $O(m)$.

3. (a) Insert the following keys into an initially empty hash table of 11 slots, numbered 0 through 10, using the hash function $h(k) = (3k+5) \bmod 11$ and assuming collisions are handled by linear probing:

$$13, 8, 33, 2, 20, 7, 15, 42, 9.$$

You are to draw the final hash table.

(b) Same as before, but assuming collisions are handled by quadratic probing.

(c) Same as before, but assuming collisions are handled by double hashing using the secondary hash function $h'(k) = 7 - (k \bmod 7)$.

Solution:

Starting from slot 0 to slot 10; a '$-$' means that the slot is empty.

(a) $13, 2, 42, 9, 7, 33, 15, 8, -, -, 20$

(b) $13, 2, -, 42, 7, 33, 15, 8, 9, -, 20$

(c) $13, 20, 42, -, 7, 33, 15, 8, -, 9, 2$