1.Design an efficient algorithm to find the n/2th largest element in a binary heap of n elements. What is the running time? (Let h denote the height of the heap.)

2.Assume a heap is arranged such that the largest element is on the top. Suppose you are given two heaps S and T, each with m and n elements respectively. What is the running time of an efficient algorithm to find the 10th largest element of S U T?

3.What is minimum possible number of nodes in a red-black tree which contains two black nodes from every root-to-leaf path?

4.Which of the choices below best describes the worst-case performance of the Union-Find data structure we discussed (with union by rank and path-compression)? (Pick the strongest bound which is valid in the worst case.)

5.Suppose you are given a balanced binary search tree (such as a red-black tree). Design an algorithm to find the second largest element in the tree. What is the running time of your algorithm? Explain your answer in 1-2 sentences

6.Suppose we define a red-black tree where along each path the number of black nodes is the same, and there cannot be more than three consecutive red nodes (but there may be two consecutive red nodes).

i. What is the ratio of the longest possible path length to the shortest possible path length in this tree?

ii. For a tree that has b black nodes along each path, what ratio of the maximum possible number of nodes to the minimum possible number of nodes in the tree?

7. Given any node u in a binary tree, let the number of nodes of the subtree tree rooted at u be nodes(u), the number of nodes of the left subtree at u be leftnodes(u), and the number of nodes of the right subtree be righnodes(u). Clearly we have nodes(u) = leftnodes(u) + rightnodes(u) + 1.

We now introduce a new definition of approximately balanced trees. A tree is approximately balanced if leftnodes(u) <= 2*rightnodes(u), for all nodes u in the tree.

Given a tree with n nodes, determine an upper bound for height of the tree, i.e. determine if in the worst case the height = O(n) or O(sqrt(n)), or O(logn), etc.

Hint: use recurrence relations to express the height.

8.Assume that a binary search tree (not a red-black tree) tree was created by inserting the seven numbers in the sequence 1, 2, …, 7

i.Draw the resultant tree

ii. Suppose you are now allowed to do rotation operations anywhere in the tree. Draw a sequence of rotation operations that will eventually complete balance the tree. Clearly mention which edges are being rotated.

9. Assume you are given a red-black tree with seven nodes containing the numbers 1, 2…7. Assume that all the nodes are black.

i. Draw the tree

ii. Insert the number 4.5 into the tree and show the sequence of steps required to rebalance the tree.

10. Suppose we are working with a Union-Find data structure as described in class (assume no path compression is applied). Suppose we consider a set of 10 items that are eventually joined into a single set via 9 union operations

i. Describe the sequence of union operations that will result in the root having the largest number of children. What is the final degree of the root in this case?

ii. Can you have a sequence of union operations that will result in the root having three children? Explain why or why not.

11.Suppose we are working with a Union-Find data structure as described in class (this time path compression is allowed for Find operations). Node z is at a distance of 4 from the root r of its tree (i.e. 4 edges away). We now perform a Find on z.

i. By how much will the degree of r change? Explain your answer.

ii. By how much will the degree of z change? Explain your answer.

12.Recall the definition of the "iterated logarithm function" log*(n).

i. Similarly, try and define the "iterated square root function" sqrt*(n).

ii. What is the value of sqrt*(2^32)?

13. Consider the two expressions A(k) = (2^(2^(2^(….2))))))) and B(k) = (((((((2^2)^2)^2)^2)..^2) where in each function the number of 2's is equal to k

i. (a) Plot A(k) and B(k) as a function of k (just draw a rough plot that shows the shapes of the two functions and shows which function is larger)

ii. How do these functions compare against the Ackermann' s function Ak(1)?

14.Suppose you implement a heap (with the largest element on top) in an array. Consider the different arrays below, determine the one that cannot possibly be a heap:

1. 7 6 5 4 3 2 1
2. 7 3 6 2 1 4 5
3. 7 6 4 3 5 2 1
4. 7 3 6 4 2 5 1
5. 7 4 6 3 2 1 5

15.Suppose you are given a completely balanced binary search tree (a completely balanced tree means that each path from root to leaf is exactly the same). Finding the median of the elements of such a tree takes time

1. O(1)
2. O(log*n)
3. O(logn)
4. O(n)
5. O(nlogn