Asymptotics, Recurrence and Basic Algorithms

1. [1 pt] What is the solution to the recurrence **T(n) = 2T(n-1) + 1, T(1) = 1**

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

2. [1 pt] What is the solution to the recurrence **T(n) = T(n/2) + n, T(1) = 1**

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

3. [1 pt] What is the solution to the recurrence **T(n) = 2T(n/2) + 1, T(1) = 1**

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

4. [1 pt] Determine the **correct** statement(s) in the group below
    1. The worst-case performance of quick sort using a random partition element is the same as that of merge sort
    2. The worst-case performance of quick sort using the median as a partition element is different from that of heap sort
    3. **Heap sort has the same worst-case performance as merge sort**
    4. All three algorithms mentioned above are based on the divide and conquer paradigm, i.e., divide the set into two, recursively sort each part, and combine the results.
    5. Merge sort does not require any more extra space than the other two algorithms

5. [1 pt] Determine the **incorrect** statement in the group below

    1. **The worst-case performance of quick sort is more than that of merge sort**
    2. Heap sort has the same worst-case performance as merge sort
    3. The worst-case performance of quick sort can be improved using the median-finding algorithm
    4. **All three algorithms mentioned above are based on the divide and conquer paradigm, i.e., divide the set into two, recursively sort each part, and combine the results.**
    5. Merge sort requires more extra space than the other two algorithms

6. [1 pt] Determine the **correct** statement(s) in the group below

    1. **The worst-case performance of quick sort using a random partition element is the same as that of bubble sort**
    2. **The worst-case performance of quick sort using the median as a partition element is different**

from that of bubble sort

3. **Heap sort has the same worst-case performance as merge sort**
4. Of the four sorting algorithms mentioned above, three algorithms are based on the divide and conquer paradigm, i.e., divide the set into two, recursively sort each part, and combine the results.
5. **Quick sort does not require any more extra space than heap sort**

BST and Balanced Trees:

1. Assume that a binary search tree (not a red-black 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.

2. 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)

3. [1 pt] 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). Determine the correct statement(s) below:

1. Finding the largest element of this tree can be done in O(1) time
2. **Finding the second smallest element requires at least O(logn) time**
3. **Finding the median can be done in O(1) time**
4. Finding the median requires at least O(logn) time
5. None of the above

Heaps:

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 denotes the height of the heap.)

See Quiz 2 Practice Question Solution

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?

See Quiz 2 Practice Question Solution

3. 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

See Quiz 2 Practice Question Solution.

4.[1 pt] Suppose you implement a heap (with the largest element on top) in an array. Consider the different arrays below, determine the one(s) that **cannot possibly be a heap**:

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

See Quiz 2 Practice Question Solution

Red-Black Trees

1. Insert the following numbers into an initially empty red-black tree: 8 2 4 7 5 3 1 6. (To solve this problem correctly, you will need to refer to the text-book for details about red-black trees in addition to the main ideas discussed in class).

2. Design a "worst-case" red-black tree with 10 nodes, i.e., a red-black tree with the longest possible path from the root to a leaf.

See Quiz 2 Practice Question Solution

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

See Quiz 2 Practice Question Solution

4.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.

Solution: Parent of right most child, O(log n)

5. 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 two 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? **3**

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?

$(2^{3b} -1)/(2^{b}-1)$

6. 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.

Union-find

1. Can you use any of the previously studied data structures (e.g. heaps, red-black trees) for the Union-Find problem? Explain your answer.

2. What is the worst-case performance of the Union-Find data structure we discussed (with union by rank and path-compression)?

3. 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.

Solution: i. Always union others with one item. Final degree of root will be 9.

ii. Suppose we are given, 1 to 10.

By union by rank: union (1,2), union(1,3), union(1,4), union(1,5), union(1,6), union (7,8), union (9,10), union (7,9), union(1, 7).

By union by size: Not Possible.

4.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.

5. 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})?

6. [1 pt] Determine the incorrect statement below concerning the Union-Find data structure we discussed in class (with union by rank and path-compression).

1. The union operation sometimes may not increase the height of the resultant tree **T**
2. The union operation can at most increase the height of the resultant tree by one **T**
3. The union operation always increases the height of the resultant tree by one **F**
4. The find operation sometimes may not increase the degree of the root of the resultant tree **T**
5. The find operation requires two traversals from node to root **F**

Graphs

1. [1 pt] For an undirected graph with n vertices and m edges, the degree of a vertex is defined as the number of edges that have the vertex as an endpoint. Suppose the degree of each vertex of the graph <=4. Which of the following statement(s) are correct?

1. **m <= 4n**
2. **m <= 2n**
3. m <= n
4. **m <= n^2**
5. none of the above

2. [1 pt] For an undirected graph with n vertices and m edges, the degree of a vertex is defined as the number of edges that have the vertex as an endpoint. Suppose the degree of each vertex of the graph >=4. Which of the following statement(s) are correct?

1. m >= 4n
2. **m >= 2n**

3. m >= n
4. m >= n^2
5. none of the above

3. [1 pt] You are told that an undirected graph has n vertices and n -1 edges, and is disconnected. Which of the following are correct statement(s)?

1. The graph may not have a cycle
2. The graph definitely has a cycle
3. The graph has exactly two connected components
4. The graph may have more than two connected components
5. None of the above

4. Design an algorithm to check whether an undirected graph G = (V, E) has a Euler cycle (find out what Euler cycle means). What is the complexity of your algorithm?

5. Consider a directed graph G = (V, E) that has no cycles. Design an efficient algorithm that outputs the vertices in a sequence {v_1, v_2....v_n} such that for all i < j, there is no edge from v_j to v_i. (Hint: use depth-first search)

6. Suppose you are given a graph that is “almost disconnected”, i.e., the removal of a single edge will disconnect the graph. Design an algorithm to efficiently find this edge.

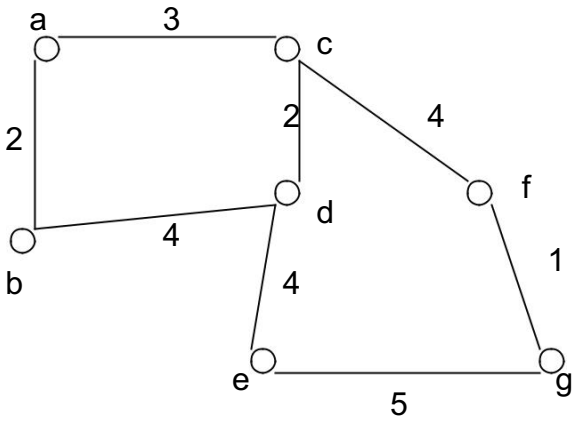
7. What are the kinds of graphs for which depth-first search orderings and breadth-first search orderings are the same?

8. [1 pt] For an undirected graph with n vertices and m edges, the degree of a vertex is defined as the number of edges that have the vertex as an endpoint. Suppose the degree of each vertex of the graph is 2. Which of the following statements is correct?

1. m = 2n
2. m = n
3. m = n/2
4. m = n^2
5. none of the above

Minimum Spanning Trees

1. [2 pt] Run Kruskal's MST algorithm on the following graph



Solution: fg U cd U ab U ac U cf U de

2. [2 pt] Consider an undirected graph with n vertices, and m edges. Assume that the edges are of two types: m_1 red edges and m_2 green edges. Thus $m = m_1 + m_2$. The red edges have weight 1, and the green edges have weight 2. Design and analyze an efficient algorithm to compute the minimum spanning tree of such a graph.

3. Analyze the running time of Kruskal's minimum spanning tree algorithm when the input is an adjacency matrix.

4. [2 pt] A weighted undirected graph with n vertices and m edges is said to satisfy the triangle inequality if for every edge (u, v), the weight of (u, v) is less than or equal to the length of any other alternate path from u to v.

a. Prove that for such a graph, the total weight of all edges is <= (m-n+1)*MST, where MST is the total weight of all edges of the minimum spanning tree.

(Hint: What is the maximum possible weight of an edge of the graph that does not belong to the minimum spanning tree?)

5. A "geometric graph" is a special type of graph where the nodes are points on a 2-dimensional surface and edges are straight lines joining pairs of nodes. Show that the minimum spanning tree of such graphs cannot have edges that cross each other (other than at their endpoints).

6. Suppose you have constructed a minimum spanning tree of a graph (arbitrary graph with arbitrary weights). A new edge is now inserted into the graph. Design an algorithm that will efficiently compute the new MST.

7. [2 pt] Can you simply reverse Kruskal's algorithm to compute the **maximum spanning tree** of a weighted undirected graph? Either give a proof that this works, or show by an example why this will not work.

8. [2 pt] Consider a weighted graph G with n vertices and m edges such that all weights are distinct. Let MST be the minimum spanning tree of this graph.

1. Suppose you insert another edge into this graph, and let MSTInsert be the new minimum spanning tree of this graph. What is the minimum number of common edges between MST and MSTInsert?

2. Given G, MST, and the new edge, design and analyze an efficient algorithm to compute MSTInsert.

Instead of inserting, suppose you delete an edge from this graph, and let MSTDelete be the new minimum spanning tree of this graph. What is the minimum number of common edges between MST and MSTDelete