CS 161: Practice Midterm (Fall 2016)

Instructions: Please answer the following questions to the best of your ability. There are five question worth a total of 75 points. You may cite any claim from class or CLRS without proof. Good luck!

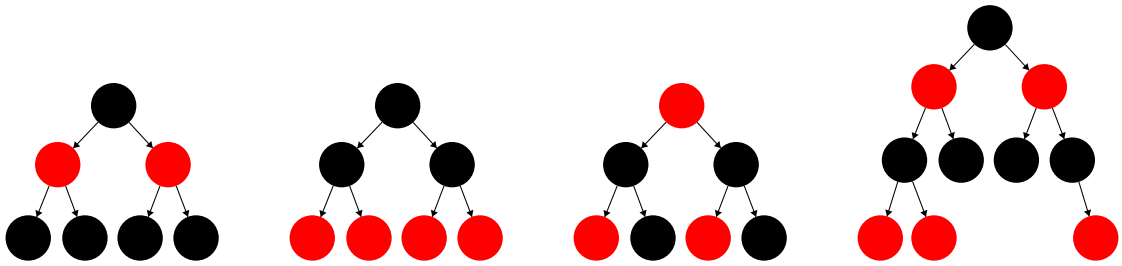
Question 1: True or False (10 points)

For each statements, indicate whether it is true or false. No justification is needed.

1. For all positive functions $f(n)$, we have $f(n) + O(f(n)) = \Theta(f(n))$.
2. $n^{\log n} = O(2^n)$.
3. Depth-first search can be used to detect cycles in both directed and undirected graphs.
4. A list of n English letters can be sorted in $O(n)$ time.
5. Given a sorted list of n distinct numbers, we can construct a binary search tree on the numbers in $O(n)$ time.
6. Suppose a hash table draws its hash function from a universal hash family. Then there will be no collisions if the number of buckets (size of the underlying array) exceeds the number of items in the hash table.
7. Suppose a hash table has more buckets than the size of the universe of keys. Then there exists a hash function such that there will be no collisions.
8. A binary search tree with n nodes has depth $O(\log n)$.
9. $4^n = \Theta\left(2^n\right)$.
10. The expected running time of quicksort is $O(n \log n)$ even if the inputs are not random, but picked by an adversary.

Question 2 (8 points)

You are given the following four red-black trees. Circle the ones that are valid; no proof necessary. Don't circle any subtrees.



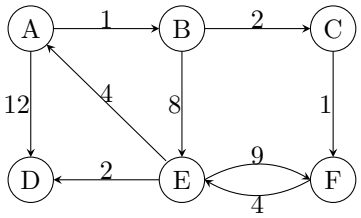
Question 3 (20 points)

A *reversal* in an array L is a pair of indices (i, j) such that $i < j$ and $L[i] > L[j]$. In this problem, you will design an efficient algorithm CountReversals for counting the number of reversals in an array of size n and analyze its running time. In particular, we will design CountReversals by modifying MergeSort. Let $L[a : b]$ denote the subarray of L starting at position a and ending at position b (including a and b).

1. Suppose that we have counted the number of reversals in the subarrays $L[1 : n/2]$ and $L[n/2 + 1 : n]$. Describe the set of reversals we need to count in order to obtain the number of reversals in $L[1 : n]$.

2. Design CountReversals by modifying MergeSort. In particular, describe and analyze the modified merge step.

Question 4 (15 points)



1. Give a possible BFS ordering of the graph originating from node A. Assume that the adjacency list of a node is ordered alphabetically.

2. Give a possible DFS ordering of the graph originating from node A. Assume that the adjacency list of a node is ordered alphabetically.

3. Simulate running Dijkstra's algorithm from node A: List the nodes in the order in which the algorithm computes their correct distance from A. For each node, specify the distance from the source and the path produced by Dijkstra's.

Question 5: Short Questions (22 points)

In this section, you are asked to write short answers (2-3 sentences) to each question. When you design an algorithm, describe how it works, but do not prove its correctness or explain the analysis of the running time.

Merging Binary Search Trees (6 points)

Assume T_1 and T_2 are binary search trees, not necessarily balanced, with m and n nodes, respectively. Design an algorithm that merges T_1 and T_2 into a balanced binary search tree in $O(m+n)$ time.

2-SUM (6 points)

You are given two unsorted arrays A, B of n integers, and an integer z. Find if there are two elements $A[i]$ and $B[j]$ such that $A[i] + B[j] = z$ in expected running time $O(n)$.

Recurrences (10 points)

Solve the recurrences below giving tight upper bounds of the form $T(n) = \Theta(f(n))$. You can use any method from class. You do not have to show your work. If you wish, you may assume that n initially has the form $n = a^i$, for an appropriate constant a.

1. $T(n) = 4T(n/2) + n^2\sqrt{n}$
2. $T(n) = 3T(n/2) + n \log n$
3. $T(n) = T(n/2) + T(n/4) + T(n/8) + n$
4. $T(n) = T(n-1) + 1/n$
5. $T(n) = T(\sqrt{n}) + 1$