B1 Algorithms (25 points)

Answer all three parts.

a. [6 points] Describe each of the algorithms below. Provide input requirements, output, and a pseudocode sketch of their implementation. Analyze them in terms of time and space complexity.

   i. Quicksort.

   ii. Radix sort.

   iii. Binary search.

b. **Finding percentiles**. Suppose you are given an unordered set of N distinct real numbers and you are asked to find the value of the p^{th} percentile (the p^{th} percentile, $0 \leq p \leq 100$, of a list is a number q in the list such that ceiling(N*p/100) of the numbers are less than or equal to q; the 50^{th} percentile is called the median).

   i. [4 points] Describe the "median of medians" approach to finding the median of N numbers. Analyze the time complexity of the algorithm using the recurrence relation.

   ii. [4 points] Generalize part i to an algorithm for finding the p^{th} percentile.

   iii. [5 points] Suppose you need to find k different percentiles. Each can be found by applying the recursive algorithm from part ii. An alternative is to simply sort the list, then use constant-time random access to pick out the k values from the sorted list. For what values of k is it asymptotically more efficient to use the "sort first" method than to use the recursive algorithm? (Your answer should provide $f(N)$ in $k = \Omega(f(N))$.)

c. [6 points] Show that any comparison sort algorithm requires at least $\log_2(N!)$ comparisons in the worst case.

B2 Algorithms (25 points)

Answer all six parts.

a. [6 points] Select any 2 of the following 3 items (do only 2 -- if you do all 3, only the first 2 will be graded). For each tree that you select you should describe: the structure of the tree; the order associated with the items stored in the tree; and how an Insert operation is performed on that tree (if the Insert involves a rotation, you need only say that there is a rotation AND what the rotation is designed to accomplish. You do not have to give the details of the rotation). Do not describe how to perform any operation other than Insert.

i. AVL tree

ii. 2-3 tree

iii. Red-black tree

b. [3 points] For binomial heaps, describe: the structure; the order associated with the items stored there; how an Insert operation is performed. Do not describe how to perform any operation other than Insert.

c. [5 points] Give a proof by induction that the trees that exist in a Fibonacci Heap are binomial trees if the only operations executed in the Fibonacci Heap are Insert, Min and Delete_min.

d. [4 points] In a Fibonacci Heap, state the worst case and amortized running times for these operations: Insert, Min, Delete_min, Union, Decrease_Key.

e. [3 points] In a Fibonacci Heap, in the implementation of Decrease_Key there is a cascading cut. Explain why cascading cuts are performed.

f. [4 points] Explain the differences between an average running time and an amortized running time.

B3 Algorithms (25 points)

Select any 4 of the following 5 items (do only 4 -- if you do all 5, only the first 4 will be graded).

1. Carefully explain the key ideas in the Boyer-Moore pattern matching algorithm. Be sure to state the worst case running time.

2. Give an $O(n^3)$ algorithm for determining the optimal order in which to multiple a sequence of matrices.

3. Carefully explain how to multiply two polynomials of degree n in time O(n log n).

4. Carefully explain Dijkstra's shortest path algorithm and the implementation needed for it to run in time O(e + n log n).

5. Carefully explain Kruskal's minimum spanning tree algorithm and the implementation needed for it to run in time O(e log n).

B4 Algorithms (25 points)

Answer all four parts.

a. [5 points] Define NP-complete along with any technical terms that you use in that definition.

b. [5 points] Name and briefly describe 3 distinct NP-complete problems other than those mentioned elsewhere in this problem.

c. [3 points] Is the following problem NP-complete? Explain your answer.

INSTANCE: An undirected graph G.

QUESTION: Does there exist a vertex cover of size 5 for G ?

d. [12 points] You know that given a weighted graph there are several algorithms for finding a minimum weight spanning tree in polynomial time. For some applications, it is not enough to have any spanning tree, but there is an additional requirement that nodes in the spanning tree not have too many neighbors. In this case, the problem becomes NP-complete even if we do not care about the weight. Formally we state the problem as:

DCST (Degree Constrained Spanning Tree)

INSTANCE: An undirected graph G and a positive integer k.

QUESTION: Does there exist a spanning tree for G where each node has degree no more than k?

Give a complete proof that DCST is NP-complete. You may assume that 3SAT, CLIQUE, PARTITION and HAMILTONIAN PATH are known NP-complete problems.

Hint: Think about a specific small value for k that makes the problem hard.