B1 Algorithms (25 points)

Select any 5 of the following 6 items (do only 5 -- if you do all 6, only the first 5 will be graded). For each item that you select, give a short explanation of the item (data structure, algorithm, or technique). You should capture the main idea/ideas of the data structure, algorithm, or technique in your explanation, and, for algorithms, state the running time.

1) FFT

2) Dijkstra's algorithm

3) Fibonacci trees

4) The Knuth-Morris-Pratt algorithm

5) Kruskal's algorithm

6) An algorithm to find optimal binary search trees

B2 Algorithms (25 points)

Do all three parts.

a. [10 points] For each of the following NP-complete problems (items i, ii, iii, and iv below), carefully specify the INSTANCE and the QUESTION using the format in this example:

Example: HAMILTONIAN PATH

INSTANCE: Undirected graph G=(V,E).

QUESTION: Does G contain a Hamiltonian Path? That is, does G contain a path that includes each vertex of G exactly once?

i) PARTITION

ii) CLIQUE

iii) 3SAT

iv) Any one additional NP-complete problem of your choosing that is not a variation of or closely related to HAMILTONIAN PATH, PARTITION, CLIQUE, 3SAT or TSP.

b. [12 points] Give a complete proof that the following problem is NP-complete. You may assume that 3SAT, CLIQUE, PARTITION and HAMILTONIAN PATH are known NP-complete problems.

TSP: TRAVELING SALESPERSON PATH

INSTANCE: An integer B, and a complete weighted graph G=(V,E,w) where w is a function mapping each edge in E to a positive integer.

QUESTION: Does there exist a path in G that contains each vertex exactly once, and for which the sum of the weights of the edges in the path does not exceed B?

c. [3 points] Suppose TSP is changed to be 0-1TSP, which is identical to TSP, except that the function w is a mapping of each edge to a weight that is either zero or one. Is the problem still NP-complete? Explain briefly.

B3 Algorithms (25 points)

Do all four parts.

a. [10 points] For each of the following algorithms, describe the main idea/ideas, and state the running time, if it is not given.

i. Heapsort
ii. Selection in worst-case linear time.

b. [6 points] Explain the decision tree model, and then use it to prove that any algorithm to sort n items by comparisons of keys must do at least $\lceil lg(n!) \rceil$ key comparisons in the worst case.

c. [5 points] Describe an algorithm that, given an unordered set S of n integers and another integer x, determines whether there exist two integers in S whose sum is exactly x. The algorithm that you give should have a running time that is as small as possible.

d. [4 points] Suppose it is also known in part c that the n integers and x are in the range of 1 to n^2. Give an O(n) algorithm for solving the problem.

B4 Algorithms (25 points)

Answer all three parts

This problem concerns UNION/FIND implementations. Assume that the sets are implemented as trees, with each item pointing to its parent, using a PARENT array. The roots of the trees point to themselves.

a. [8 points] Prove that the worst case cost of processing a sequence of n UNION/FIND operations is $O(n^2)$ when the operations are implemented as follows:

```
UNION(i,j):  PARENT[i] = PARENT[j]     // assume that i and j are roots of their respective trees

FIND(i): j = i
         while PARENT[j] ≠ j do
              j = PARENT[j]
         return(j)
```

b. [8 points] Prove that the worst case cost of processing a sequence of n UNION/FIND operations is O(n log n) if the UNION is implemented using a ranking rule and the FIND is implemented as in case a.

c. [9 points] Sketch a proof that the worst case cost of processing a sequence of n UNION/FIND operations is O(n log* n) if the UNION is implemented using a ranking rule and the FIND is implemented using a collapsing rule. In your proof sketch you need only explain how to account for the cost of the FIND operations. Be sure to include all major elements of that analysis.