Purdue University S. Hambrusch
CS 381 Fall 2009

**Sample Final Exam Questions**

1.) (i) Let $\Pi_{dec}$ be an NP-complete decision problem and let $\Pi_{opt}$ be its corresponding optimization problem. Assume $\Pi_{opt}$ can be solved in polynomial time. What does this imply for $\Pi_{dec}$? What does this imply for the class of NP-complete problems and the class NP, respectively?
(ii) Does there exist a polynomial time algorithm to determine whether an undirected graph contains a clique of size 3? Explain.
(iii) If an NP-complete problem can be solved deterministically in $O(n^3)$ time, can every problem in class NP be solved in $O(n^3)$ time?

2.) For each of the problems listed below state the asymptotic running time of the best algorithm you know. If you think the problem is NP-complete, state so (you do not need to give a running time).

1. Sorting $n$ integers $a_1, a_2, \ldots, a_n$ with $0 \leq a_i \leq n^2 \log^2 n$.
2. Determining the $\frac{n}{5}$-th largest element in an unsorted set of size $n$.
3. In a directed, weighted graph $G = (V, E)$ with positive weights and $|V| = n$ and $|E| = m$, determine the shortest path between a given pair of vertices.
4. In an $n$-node rooted tree $T$, determine the number of leaves whose parent has more than one child.
5. Given a boolean formula in conjunctive normal form (i.e., $C_1 \wedge C_2 \wedge \ldots \wedge C_k$, where every $C_i$ contains an arbitrary number of literals $\vee$-ed together), determine whether there exists a truth assignment to the variables satisfying the formula.
6. Given a boolean formula in disjunctive normal form (i.e., $C_1 \vee C_2 \vee \ldots \vee C_k$, where every $C_i$ contains an arbitrary number of literals $\wedge$-ed together), determine whether there exists a truth assignment to the variables satisfying the formula.

3.) Assume $G = (V, E)$ is an undirected, connected, weighted graph, $|V| = n$, $|E| = m$, represented by adjacency lists. Weights can be positive as well as negative. For each problem listed below, give the asymptotic time bound of the best algorithm you know. Give a brief explanation of your solution.
(i) Determine whether $G$ contains at least 10 edges of cost $\geq 100$.
(ii) Determine whether $G$ is a tree.
(iii) Find a spanning tree of $G$ having minimum cost.
(iv) Given two vertices $u$ and $v$, does there exist a path from $u$ to $v$?

(v) Given two vertices $u$ and $v$, determine the length of the longest path from $u$ to $v$.
(vi) Given two vertices $u$ and $v$, determine the length of the shortest path from $u$ to $v$.
(vii) Find a shortest path tree rooted at a given vertex $u$.

4.) Let $G$ be a connected, undirected, and weighted $n$-vertex, $m$-edge graph.
(i) Assume each one of the $m$ edges has weight 2. Describe and analyze an algorithm that finds a minimum cost spanning tree of $G$. Your algorithm should be faster than the algorithms seen in class.
(ii) Assume now that the edges have either weight 1 or weight 2. Describe and analyze an algorithm that finds a minimum spanning tree of $G$. Your algorithm should be faster than the algorithms seen in class.

5.) Describe a data structure supporting each of the following queries in $O(\log n)$ time, where $n$ is the number of entries in the structure.

- insert(*weight*): insert one record of value *weight*
- delete(*weight*): delete one record of value *weight*
- min_weight: returns the smallest weight value and the number of records with minimum weight
- report(*qweight*): returns the number of records with a weight value equal to *qweight*
- report_larger(*qweight*): returns the number of records with a weight value greater than *qweight*

First describe the data structure used and then describe how each query is implemented on it. Make sure to explain how duplicate entries are handled.

6.) Let $G = (V, E)$ be a weighted directed graph (weights may be positive or negative.) Assume that the graph $G$ is input as a weighted adjacency matrix $W$; i.e., $W_{i,j} = w_{ij}$ if there is an edge between $i$ and $j$ and $w_{i,j}$ is the weight of this edge; $W_{i,j} = \infty$ if there is no edge between $i$ and $j$; and $W_{i,i} = 0$ for all $i \in V$.

Given a path between two vertices in $G$, the cost of the path is the *maximum* of the weights of the edges on the path. A *shortest* path between two vertices is a path with minimum cost (among all possible paths between the two vertices). The goal of this problem is to find the *value of the shortest path* for *all pairs* of vertices.
(a) Consider the following dynamic programming approach. Let $V = \{1, ..., n\}$. For a path $P = v_1, v_2, ...., v_{k-1}, v_k$, the vertices $v_2, ...., v_{k-1}$ are called the intermediate vertices. Let $B_{ij}^{(k)}$ be the cost of shortest path from vertex $i$ to vertex $j$ using only intermediate vertices from the set $\{1, ..., k\}$.

Complete the recursive formulation for $B_{i,j}^{(k)}$ (fill in the blanks):

$$B_{i,j}^{(k)} = \begin{cases} \text{\_\_\_\_\_} & \text{if } k = 0 \\ \text{\_\_\_\_\_} & \text{if } k \geq 1 \end{cases}$$

(b) State the running time of the algorithm associated with the recursive formulation. You do not need to give pseudo-code, only a brief explanation on how the time bound is achieved.

7.) The partition problem is defined a follows: Given a set $A$, determine whether the elements of $A$ can be partitioned into two sets so that the sum of the elements in one set is equal to that of the elements in the other set. This problem is known to be NP-complete.

Consider the following 1-processor scheduling problem: Given are $n$ jobs and job $i$ has length $l_i$, penalty $p_i$, and deadline $d_i$ associated with it. The $n$ jobs are to be scheduled on the processor. If job $i$ is not completed by time $d_i$, a penalty of $p_i$ occurs.
In the **Min_Penalty** problem we are given $l_i$, $p_i$, $d_i$, $1 \leq i \leq n$, and a quantity $P$ and are to determine whether there exists a schedule such that the sum of the arising penalties is at most $P$. Show that the Min_Penalty problem is NP-complete.