Sample Exam

Instructions:

Please write clearly and succinctly; be sure to give an overview before plunging into details. Note that if an algorithm is requested, the most efficient one is desired. Open book and notes. No electronic devices (laptops, PDA's, calculators)

1 Dynamic Programming [35 points]

We consider a variation of the 0/1 knapsack problem (so each item can be used at most once). As before there are n items, each with a integer weight w_i and an integer value v_i, and a weight limit W. In addition, each item has a **size** s_i and we have a size limit S (imagine there is both a weight limit on what you can carry and a size limit of what you can fit in the knapsack). We want the subset of items of maximum value such that their weight is at most W and their size is at most S.

We can solve this problem by computing the values of a 3D table T where $T[i, j, k]$ is the best value solution with weight exactly j and size exactly k using a subset of items $1, 2, \ldots, i$ (value - infinity if no solution has this weight and size). Below you are asked to fill in some details.

Part A. Describe how to fill in the initial table values for $i = 1$.

Part B. Give a formula for computing $T[i, j, k]$ assuming you have already computed the $T[i-1, j, k]$ values. Briefly justify your answer.

Part C. Give the running time to fill in the T matrix to within Θ accuracy.

Part D. Assuming we have filled in the entire table T (and saved it), how do we then find the actual optimal set of items to use?

2 Network Flow [10 points]

Suppose you are given a maximum flow f in a network G=(V,E) with integer capacities.

a) Suppose that we increase the capacity of a single edge by 1. Describe how to find a new maximum flow in O($|E| + |V|$) time.

b) Suppose we no longer had integer capacities. Would your solution to a) still find a max-flow in O($|E|+|V|$)

time? Justify your answer.

3 Short Answer [30 points]

For each problem below justify your answer.

A) Prove that the problem of deciding whether a number is NOT prime is in NP.

C) Must all problems which are in the class P have a fast solution algorithm?

E) Suppose that we want to find the shortest simple path from s to a vertex u in a directed, weighted graph with negative cycles. Is it i) impossible to find such a path, ii) possible but always hard, iii) possible and often hard, iv) always easy.

4 Shortest Paths [16 points]

For a weighted, directed graph G, let D be a matrix, such that $D[i,j]$ is the shortest distance between i and j in G (this is already computed).

Suppose we change one arc (p,q) by increasing its length by 2.

A) Describe a graph G (a picture is OK) where this single change will change $\Theta(n^2)$ entries in D.

b) Describe a fast way to compute the new values in D after this change to the arc (p,q). Give the run time of your algorithm.

5 Line Testing [30 points]

Consider a computer network with k processors $P_1, P_2, .., P_k$, and r communication lines $C_1, C_2, ..., C_r$. A processor P_i has the ability to do t_i tests per day, and there is a list L_i which contains the communication lines that processor P_i is able to test. Subject to these constraints we would like to be able to test all the communication lines every day. A testing schedule determines for each processor the lines it should test.

For the two settings below, i) describe a fast way to find a testing schedule or determine that no schedule can test all lines in a single day (a picture may be helpful).
ii) give the running time of your solution in terms of k and r and the sizes of the lists L_i (hint: a careful analysis should yield a better bound).

A) We require that each line be tested twice per day (so it could be tested twice by a single processor, or once each by two different processors).

B) We require that each line be tested by three **different** processors each day (once by each of the three).

6 Approximation for Vertex Cover [14 points]

Consider the *weighted* Vertex cover problem where each vertex has a weight. For an undirected graph G with vertex weights, we now want to find a vertex cover (a set of vertices such that every edge touches at least one vertex in the cover) whose total weight is minimum.

A) Suppose we use the algorithm discussed in class (and in the book), which repeatedly adds the two endpoints of an edge, then deletes all edges which touch the endpoints. Give an example graph where the total weight of the solution found is at least 100 times greater than the optimal one.

B) Now suppose that we use the same algorithm, but all vertices in G have weight one or two. If C^* is the weight of an optimal cover and $\hat{C}$ the weight of a cover found by our algorithm, we want to show that $\hat{C}/C^* \leq r$ for the smallest possible value of r.

Prove a **tight** bound on r (for the correct answer, you can show that $\hat{C}/C^* \leq r$ for all graphs, and give an example graph where $\hat{C}/C^* = r$).