CS 202 Homework 3
Greedy Algorithms, Tuning Algorithms with Data Structures, Flows, Reductions
Due June 3

General directions:

This homework has no implementation problem.

Algorithms may be described at "high level" without actual code. You may use any lower bound, algorithm or data structure from the text or in class, and their correctness and analysis, but be careful. For example, if you construct a graph with n^2 nodes and n^3 edges and then run Dijkstra's algorithm on the resulting graph, the total run-time is $O(n^3 logn)$,because Dijkstra's algorithm is $O(ElogV)$. If you want to use the correctness of Dijkstra's algorithm on the graph, you must check that edge weights are positive, since otherwise the proof that Dijkstra's algorithm gives shortest paths does not go through. If a problem has multiple size parameters, you should express the run-time as a function of all relevant parameters; e.g., saying maximum bi-partite matching takes time $O(|V||E|)$ is more accurate than to say it takes time $O(V^3)$, although both are correct.

For the first four problems, you must prove correctness and give a time analysis. For some problems, correctness may be trivial; for others the analysis may be trivial. If it is trivial, you do not need to go into detail, but you should at least give a one or two line explanation. (Conversely, if you only give a one or two line explanation, I will view this as implicitly claiming that it is trivial.)

Each problem is worth 20 points. Grading may be based on any of the following that are relevant for the problem: the efficiency of your algorithm; the correctness and proof thereof; and the time analysis. The number of points depending on each part is given after the problem, as well as a ballpark estimate of the time analysis for my solutions of the problem.

Greedy algorithms You are the evil personel director of a large factory. You wish all the employees to waste 1 hour today attending a lecture on the Official Policies for Feeding Cats During Employment Hours, but you wish to give this lecture as few times as possible. (You have several possible lecturers available, so lectures can be overlapping.) The employees work schedules for the day are intermeshed, with employee i starting at time s_i and ending at time $f_i > s_i + 1$ hour. There is no midnight shift; no one starts before 2 AM or finishes after 10 PM. Do not assume f_i, s_i are integers.

Here are two strategies for greedy algorithms for this problem. One is correct, the other incorrect. Give a counter-example for the incorrect strategy (4pts), and a proof of correctness for the correct strategy (8 pts). Then give an efficient implementation and time analysis for the correct strategy (8 points).

Strategy A Find the hour when the most workers are present. Schedule a lecture for this hour. Delete these workers and repeat, until all workers are lectured.

Strategy B Look at the first shift to end. Schedule a lecture for the last hour they work. Remove the workers that can attend this lecture and repeat.

Data structures for all pairs shortest paths You need to solve all pairs shortest path repeatedly on the same underlying graph G, but with different edge weights. Describe what algorithms and data structures you would use for optimal efficiency if:

a. **2d grid** G is the $n^{1/2} \times n^{1/2}$ grid, whose nodes are pairs (i, j) with $1 \leq i, j \leq n^{1/2}$ and where (i, j) is connected to $(i+1, j), (i-1, j), (i, j+1)$ and $(i, j-1)$ for $1 < i, j < n^{1/2}$.

b. **hypercube** G is the $d = \log n$ dimensional hypercube, where nodes are $S \subseteq \{1, ..d\}$, and two nodes S, T are connected if and only if $S \subseteq T$ and $|S| = |T| - 1$ or vice versa.

c. **complete bipartite graph** G is the complete bipartite graph with $n/2$ nodes on each side. (All points for efficiency, 4,3,3 pts per problem.)

Sub-trees A similar problem to the tree isomorphism problem from the first homework is to decide whether one tree is a subtree of another. Let T_1 and T_2 be rooted trees and let r_1 and r_2 be their roots. Let $p_1(x), p_2(y)$ be the parent of nodes $x \in T_1, y \in T_2$ in the relevant tree. We say that T_1 is a *sub-tree* of T_2 if and only if we can find a 1-1 map m (not necessarily onto; this is why it is different from isomorphism) from T_1 into T_2 so that $m(r_1) = r_2$ and $m(p_1(x)) = p_2(m(x))$ for every node $x \in T_1$. Give a poly-time algorithm to determine whether T_1 is a sub-tree of T_2 if T_1 and T_2 are rooted trees. Don't assume they are binary trees, or have small degrees. (7 pts. correct poly-time, 3 pts. efficiency. I think all algorithms will be at least cubic time, but maybe you'll suprise me.)

Flight scheduling You are devising a flight scheduler for a travel agency. The scheduler will get a list of available flights, and the customer's origin and destination. For each flight, it is given the cities and times of departure and arrival. The scheduler should output a list of flights that will take the customer from her origin to her destination that arrives as early as possible, subject to giving her at least 15 minutes for each connection. a. Give a formal specification for this problem (Instance, Solution Space,

Constraints, Objective). (4 pts) b. Give an efficient algorithm to solve the problem. (6 pts).

Number puzzle You are trying to solve the following puzzle. You are given the sums for each row and column of an $n \times n$ matrix of integers in the range $1...M$, and wish to reconstruct a matrix that is consistent. In other words, your input is M, $r_1, ...r_n$, $c_1, ...c_n$. Your output should be a matrix $a_{i,j}$ of integers between 1 and M so that $\forall j \sum_i a_{i,j} = r_j$ and $\forall i \sum_j a_{i,j} = c_i$; if no such matrix exists, you should output, "Impossible". Give an efficient algorithm for this problem. (8 points correct, poly-time algorithm; 2 points efficiency. My best algorithm takes $O(n^3)$).