Introduction to Algorithms November 14, 2003
Massachusetts Institute of Technology 6.046J/18.410J
Professors Shafi Goldwasser and Silvio Micali Handout 23

Problem Set 7

This problem set is due **in class** on **Wednesday, November 26.**

Reading: Chapters §25.1-25.2, §26.1-26.3.

There are **five** problems. Each problem is to be done on a separate sheet (or sheets) of three-hole punched paper. Mark the top of each sheet with your name, the course number, the problem number, your recitation section, the date, and the names of any students with whom you collaborated.

Problem 7-1. Shortest Paths

Consider a given directed weighted graph $G = (V, E)$ in which all edge weights are positive. Suppose that you have already computed a distance matrix D, where $D_{i,j} = d(i, j)$ is the length of the shortest path from node i to node j.

(a) Give an $O(1)$ algorithm that, on input s, u, v, t, finds the length of the shortest path from s to t that passes through both u and v.

(b) Give an $O(n)$ algorithm that, on input s and t, outputs a list of all vertices $v \in V$ such that v is on *some* shortest path from s to t.

(c) Give an $O(n \log n)$ algorithm that, on input s and t, outputs a shortest s-t path in G, i.e. your algorithm should output a list of vertices $s = v_0, v_1, \ldots, v_k = t$ such that $\sum_{i=1}^{k} d(v_{i-1}, v_i) = d(s, t)$.)

Problem 7-2. All-Pairs Shortest Paths

Give an implementation of the FLOYD-WARSHALL algorithm that uses $O(n^2)$ space.

Problem 7-3. Transitive Closure

We define a directed graph G^- to be **edge-parsimonious** if among all directed graphs with the same transitive closure as G^-, graph G^- has a minimum number of edges. (See CLR pages 632–633 for a definition of transitive closure.)

Given a directed acyclic graph G, give an $O(VE)$ algorithm that finds an edge-parsimonious graph G^- with the same transitive closure as G.

Problem 7-4. Greedy Maximal Matching.

Give a linear time algorithm that, on input graph $G = (V, E)$, finds a matching with size at least half that of a maximum matching.

Problem 7-5. Flying Friends to San Francisco

You are stuck in San Francisco and you want to fly as many friends as possible from Boston to San Francisco to celebrate your birthday tommorrow. You have a schedule F of n flights available to you today. An entry F_i in this schedule is described by five values $F_i = (s_i, t_i, d_i, a_i, x_i)$:

- s_i = starting city
- t_i = ending city
- d_i = departure time
- a_i = arrival time
- x_i = number seats available

Give an efficient algorithm that determines the maximum number of friends you can fly from Boston to San Francisco in time for your birthday party using the flights from schedule F. You may assume that all flights run precisely on time, and that transfers between flights are instant (i.e., if $a_i \leq d_j$ for flights F_i and F_j and $t_i = s_j$, then it is possible for one student to take flight F_i and then flight F_j). Be sure to argue correctness and analyze the running time of your algorithm.

Hint: You might want to transform the schedule into a max-flow instance and then solve the max-flow instance. You may use any max-flow algorithm as a black box.