TEL AVIV UNIVERSITY  
Department of Computer Science  
0368.4281 – Advanced topics in algorithms  
Spring Semester, 2011/2012  

Homework 4, June 12, 2012  

Due on Wednesday June 27. Please keep a copy of the homework and put a hardcopy in my mailbox if you can!

1. (a) Given a minimum cost circulation in a graph \( G = (V, E) \) show how to compute node potentials \( \pi(v), v \in V \), such that all reduced costs with respect to \( \pi (c^{\pi}(e)) \) of residual edges are non-negative. What is the running time of your algorithm? 
(b) Let \( \pi(v), v \in V \), be potentials such that there exists a feasible circulation \( f \) whose residual edges have non-negative reduced costs with respect to \( \pi \). Given \( \pi \) show how to find a minimum cost circulation whose residual edges have non-negative reduced cost with respect to \( \pi \). What is the running time of your algorithm? 

Try to find efficient and simple algorithms.

2. Give a graph \( G = (V, E) \), with non-negative costs and capacities, a source \( s \), and a sink \( t \), consider the following algorithm for finding a maximum flow from \( s \) to \( t \) of minimum cost. We start with the 0 flow, find an augmenting path of minimum cost in the residual network, augment as much flow along the path as possible, update the residual network, and repeat (that is find a minimum cost augmenting path in the new residual network, augment, and so on).

a) Prove that if all costs and capacities are integers then this algorithm indeed finds a max flow from \( s \) to \( t \) of minimum cost.

b) Assume the maximum cost of an edge is \( C \) and the maximum capacity of an edge is \( U \). Suggest a concrete implementation of this algorithm and analyse its running time.

3. (a) Let \( G = (V_1, V_2, E) \) be a complete bipartite graph where \( |V_1| = |V_2| = n \), and \( E = \{(x, y) \mid x \in V_1, y \in V_2\} \). Each edge \( e \in E \) has a non-negative cost \( c(e) \). Show how to find a minimum cost perfect matching in \( G \) using the minimum cost flow algorithm described in class.

(b) A bus which can carry at most 50 people travels along a path through locations 1, 2, \ldots, \( n \) in this order. For every \( 1 \leq i \leq n \), and \( 1 \leq j \leq n \), such that \( i < j \), you are given \( q_{ij} \), which is the maximum number of people that want to travel from point \( i \) to point \( j \); and the price \( f_{ij} \) for a single ticket that takes you from point \( i \) to point \( j \). Show how to use the minimum cost flow algorithm to find a “plan” for the bus driver to make maximum profit. A plan means how many people to load/drop at each location such that it never carries more than 50 people.

4. Consider the dynamic connectivity algorithm presented in class. Assume that you get as an input a graph \( G = (V, E) \) and a spanning forest \( F \), together with a level \( \ell(e) \) for each edge \( e \in E \). Suggest an algorithm that decides if \( F \) and the level function \( \ell \) satisfy the invariants of the dynamic connectivity algorithm. Analyze the running time of your algorithm.
5. Let \( S \) be a set of horizontal segments on the line. The set \( S \) satisfies the following “nesting” property: Every two segments \( s_1, s_2 \in S \) are either disjoint or one contains the other. In addition each segment \( s \) has a cost \( c(s) \). Assume that the costs of different segments are different. Note that such a family of segments defines a natural forest where \( s_1 \) is the parent of \( s_2 \) if \( s_1 \) contains \( s_2 \) and there is no segment \( s \) that is contained in \( s_1 \) and contains \( s_2 \).

A query is a point \( q \) on the line and the answer should be the largest cost of a segment containing \( q \), or an indication that there is no segment containing \( q \).

(a) Show how to preprocess \( S \) into a simple data structure that can answer such queries in \( O(\log n) \) time per query. What is the size and the construction time of your data structure?

(b) We want to support also an insert and a delete operations. When we insert a segment we are guaranteed that the nesting property is preserved. Describe a data structure in which each operation (insert/delete/and query) takes \( O(\log n) \) time.

(Hint, use dynamic trees and handle high degree vertices in a way similar to the algorithm for maintaining a spanning tree in a dynamic planar graph.)