Department of Information and Computing Sciences, Faculty of Science, UU.
Made available in electronic form by the TBC of A–Eskwadraat
In 2010-2011, the course INFOAN was given by Dr. H.L. Bodlaender.

Algorithms and Networks (INFOAN)
07-03-2011

You have two hours for the exam. You may give your answers in Dutch or in English. Write clearly.
You may consult four sides of A4 with notes.
Results used in the course or exercises may be used without further proof, unless explicitly asked.
Some parts are harder than others: use your time well, and make sure you first finish the easier parts!

Question 1. 2-opt (1+1 point)

a) Explain in your own words the 2-opt heuristic for the travelling salesman problem.

b) Suppose our cities are points in the plane, with distances between these cities/points the (usual) Eucledian distances. Suppose we start with a random tour, and then improve it with 2-opt as long as possible. Is it possible that the resulting tour crosses itself? Explain!

Question 2. Weighted cost matching model (2 points)

For the chess club *The Woodpushers*, each week it should be decided which matches are played. At eight o'clock, the arbiter sees which of the members are present. The arbiter mustmake a pairing of the members, i.e., for each member he must decide to which other member a chess-game must be played.
If the number of present members is even, each member plays one game against another member. If the number of present members is odd, there is one member that does not play a game.
Some pairs of members already have played a game earlier in the year against each others, other pairs not. All members have a rating: an integer in the range of l to 2700.
Over all possible such pairings, we want to find a pairing such that

- If possible, nobody should play against someone he/she already has played against. If this is not possible, a pairing must be found such that the number of players that plays against a player they have already played is minimized.
- Over all pairings that fulfill the previous condition, we must take a pairing such that the sum of the rating differences of the matches is as small as possible.

For instance, if players A, B, C, and D have ratings 1000, 1100, 1200 and 1400, player A has played against B, C, and D, and player C has played against D, then a solution is "A-C" and "B-D"; "A-B" and "C-D" is not allowed as this has four instead of two players that play against someone they already played against. **Show that this problem can be modeled as a *minimum cost matching* problem.** In particular, explain what vertices and edges represent in the graph model, and what cost function is used for the edges.

Question 3. Minimum cost flow: model and multiple sources and sinks (2+2 points)

In this exercise, we look at the application of minimum cost flow, as discussed in the course. We first repeat some information given in the course. (Variable names can be different in the description.)
We have an airline, that makes a multistop flight: it starts at location l_1, then flies to location l_2, then to l_3, and so on, until it reaches l_n. The airline never can carry more than B passengers. For each i, j, $1 \leq i < j \leq n$, there are n_{ij} passengers that want to fly from i to j. We can decide if we accept these passengers; a passenger from i to j gives p_{ij} profit if we accept him/ her.

We can model this as a *minimum cost flow with multiple sources and multiple sinks*. Each source s_i has a *supply* $v(s_i)$; each sink t_j has a demand $d(t_j)$. In this variant of the minimum cost flow problem, we look to a flow with minimum cost, such that each source s_i sends out $v(s_i)$ more flow than it receives; each sink t_j received $d(t_j)$ more flow than it receives. As usual, nodes that are no source or no sink send out as much flow as they receive.
To model the problem to decide which passengers to accept in order to maximize profits, we use the following model. We have n vertices representing the locations $l_1, l_2, \ldots, l_n$ with arcs (l_i, l_{i+1}) for $1 \leq i < n$, each with cost 0, and capacity B. For each i, j, $1 \leq i < j \leq n$, we have a vertex a_{ij} which is a source with supply n_{ij}. There are arcs (a_{ij}, l_i) with cost $-p_{ij}$ and capacity n_{ij}, and (a_{ij}, l_j) with cost 0 and capacity n_{ij}. l_i is not a source or a sink. Each l_i, $i > 1$ is a sink with demand $\sum_{j=1}^{i-1} n_{ij}$.

a) Due to revenue management, at most tracks there are passengers that have to pay different prices for the same chair and same track. Suppose that for the track from l_2 to l_5 there are two price categories, and there are $n_{25,1}$ passengers that give a profit of $p_{25,1}$ if we accept them, and $n_{25,2}$ passengers that give a profit of $p_{25,2}$ if we accept them.
Explain how we can augment the model for the situation with multiple price categories, i.e., how we can augment the model such that it handles cases as explained in the previous paragraph. The resulting model must still be an instance of minimum cost flow with multiple sources and sinks.

b) Suppose we have an algorithm A that solves the minimum cost flow problem with one source and one sink. **Explain how we can use A to solve a minimum cost flow problem with multiple sources and multiple sinks.**

Question 4. Nice tours (2 points; you can gain part of the points for a polynomial time algorithm)

For the *World United Tourist Offices*, a program must be made that finds "nice tours". We have an undirected graph $G = (V, E)$, and each edge $e \in E$ has associated to it a *beauty* $b(e) \in N$.
The beauty of a cycle in G is the *minimum* over all edges in the cycle of the beauty of the edge.
We want to determine what the maximum beauty of a cycle in G. I.e., what is the largest K , such that G has a cycle c with all edges on c having $b(e) \geq K$? **Give an algorithm for this problem that uses $O(|V| + |E|)$ time.**
If you do not succeed in finding an algorithm that uses $O(|V| + |E|)$ time: for a part of the score for this part, give an algorithm that uses polynomial time for the problem.
Explain the correctness and running time of your algorithm.