Algorithms and Data Structures
K. Mehlhorn and R. Seidel
Exercises 31 to 34

Summer 2008
Wednesday, Sept. 24th
Thursday, Sept. 25th

Motivation

We study variants of the minimum cost flow problem and applications of it. The book by Ahuja, Magnanti, and Orlin is a rich source for material about flows.
Let $G = (U \cup V, E)$ be a bipartite graph, i.e., every edge has one endpoint in U and one endpoint in V; $w : E \to \mathbb{R}$ is a weight function on the edges. A matching M is a set of edges such that no two edges in M share an endpoint. The weight of a matching is the sum of the weights of its edges, i.e.,

$$w(M) = \sum_{e \in M} w(e) \ .$$

A matching is called perfect if $|M| = \min(|U|, |V|)$.

1. Consider the following network. We have vertices 1 to 4 and edges
   - 1 to 2 with cost 2 and capacity 4,
   - 2 to 4 with cost 3 and capacity 3,
   - 1 to 3 with cost 2 and capacity 2,
   - 2 to 3 with cost 1 and capacity 2,
   - 3 to 4 with cost 1 and capacity 5.

   We want to send 4 units from node 1 to node 4.
   - Execute the negative cycle cancelling algorithm.
   - Execute the mincost augmenting path algorithm. Do two versions, first without node potentials, and then with node potentials. The later version was discussed in class.
2. Ask questions about the material presented in class.
3. Design an algorithm for computing minimum weight matchings in bipartite graphs. Hint: Reduce to mincost flows.
4. Design an algorithm for computing maximum weight matchings in bipartite graphs.
5. Design an algorithm for computing maximum weight perfect matchings in bipartite graphs.

6. Consider a checker board from which some of the squares are missing. We want to know whether the remaining squares can be covered by dominos (a domino covers two adjacent squares, one black and one white).

7. A truck is on a trip through cities 1, 2, to n. There is a request for transporting r_{ij} units from city i to city j. For each unit transported, the truck driver will receive a fare of f_{ij} Euros. The truck has a capacity of L and therefore the driver will not be able to satisfy all requests. Design an algorithm that computes an optimal schedule, i.e., a schedule maximizing revenues.

8. (rank-maximal matchings) Our edge set E is partitioned into disjoints set E_1 to E_r. We call the edges in E_i edges of rank i. For a matching M, its signature is the tuple

$$(|M \cap E_r|, |M \cap E_{r-1}|, \ldots, |M \cap E_1|) \ .$$

   We order signatures lexicographically. For example, $(4,3,2,4)$ is better than $(4,3,1,5)$. Show how to compute a matching with maximum signature. Show how to compute a perfect matching of maximum signature.

   Hint: Reduce to weighted matchings. You must give the edges in E_i much higher weight than the edges in E_{i-1} for all i.

9. (stable matchings) We have a set of n men and n women. Each person has a linear ordering of the persons of the other sex. We call this ordering her or his *preference list*. For example, if there are 3 man and 3 woman, women 1 might rank the men in the order 1, 3, 2. Let M be a perfect matching. A *blocking pair* for M is a pair $(m,w) \notin M$ such that m prefers w over his mate in M and w prefers m over her mate in M. A matching is *stable* if there is no blocking pair.

   (a) Give an instance with $n = 3$, a matching and a blocking pair. Also give a stable matching.

   (b) Consider the following algorithm (please read to the end of the exercise before you accuse the instructor of being a male chauvinist.

   We start with an empty matching M. As the algorithm proceeds, women will delete man from their preference list. The algorithm stops when M is perfect.

   As long as M is not perfect, let w be a woman that is not matched under M. Let m be the topmost man of w's list. We say that w *proposes* to m.

   If m is unmatched, we add (m,w) to M.

   If m is matched, say to w', and m prefers w over w', we replace (m,w') by (m,w) in M. We say that m *rejects* w' for w.

   If m is matched, say to w', and m prefers w' over w, w deletes m from her preference list. We say, that m *rejects* w for w'.

   The algorithm stops with failure if some woman runs out of candidates, i.e., her preference list become empty.

(c) Execute the algorithm on your example of item (a).

(d) Prove:

- if a man is ever matched in the course of the algorithm, it stays matched.
- Moreover, it will be matched to better and better partners according to his preference list.
- The algorithm cannot end with failure, i.e., no woman ever exhausts her preference list.
- The algorithm constructs a stable matching, call it M_0.
- Let w be any woman. Then she was rejected by all man that she prefers over her partner in M_0.
- Show that for no woman w there is a stable matching M such that w prefers her mate in M over her mate in M_0.
  Hint: Consider the *first time* in the execution of the algorithm that a man m rejects a woman w and yet there is a stable matching, say M, with $(m,w) \in M$.
- Argue that the preceding name justifies to call M_0 *woman-optimal*.

Have fun with the solutions.