Exam Algorithms and Networks 2012/2013

This is the exam for part I of Algorithms and Networks.

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.

Each of the five tasks: 1, 2, 3, 4, 5 counts for 2 points.

Some parts are harder than others: use your time well, and make sure you first finish the easier parts!

Good luck!

1. 2-OPT and flow (0.7 + 0.3 + 1.0 points)

(a) Explain in at most five lines the basic ideas of the 2-OPT algorithm for TSP.

(b) Is the 2-OPT algorithm

1. a construction heuristic or
2. an exact algorithm or
3. an improvement heuristic?

Write down the number of the correct answer. No further explanation of your answer for this part is needed.

(c) Suppose we have a flow network with costs and with lower bounds, with the following vertices, arcs, costs, lower bounds, and capacities.

- We have three vertices s, a and t.
- We have three arcs:(s,a), (a,s), and (a,t).
- The capacities of arcs are as follows: $c(s,a) = 5$, $c(a,s) = 3$, $c(a,t) = 4$.
- The lower bounds for arcs are as follows: $\ell(s,a) = 2$, $\ell(a,s) = 0$, $\ell(a,t) = 1$.
- The cost of arcs are as follows: $\text{cost}(s,a) = 10$, $\text{cost}(a,s) = -5$; $\text{cost}(a,t) = -2$.

Now, we send four units of flow from s to t through this network, i.e., we take a flow f with $f(s,a) = 4$ and $f(a,t) = 4$.

Draw the residual network G_f. Give for each arc in the residual network capacity, cost, and lower bound.

1

2. Rounding matrices and sums (1.5 + 0.5 points)

In this exercise, we extend an example of an application of maximum flow.

We have an n by n matrix of real numbers: $A[1 \ldots n, 1 \ldots n]$. For each row i, $1 \leq i \leq n$, we have the *row-sum* $R_i = \sum_{j=1}^{n} a_{ij}$. For each column j, $1 \leq j \leq n$, we have the *column-sum* $C_j = \sum_{i=1}^{n} a_{ij}$.

We also have (and this is new for this exercise) the total sum $S = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij}$.

We want to round each number a_{ij} up or down, such that

- for each row i, the sum of the rounded numbers on row i is either $\lfloor R_i \rfloor$ or $\lceil R_i \rceil$.
- for each column j, the sum of the rounded numbers on row j is either $\lfloor C_j \rfloor$ or $\lceil C_j \rceil$.
- the total sum of all entries in the matrix is either $\lfloor S \rfloor$ or $\lceil S \rceil$.

1. **Explain how we can find such a 'consistent rounding' with help of flow techniques in polynomial time.**
2. Does such a consistent rounding always exist? Why or why not?

(Remark: the only difference between the example discussed in the course and this exercise is the consideration of the total sum and the third condition.)

3. Algorithm design: Matching in almost bipartite graphs (1 + 0.5 + 0.5 points)

In this exercise, we consider the *weighted matching* problem: given is a graph $G = (V, E)$ with n vertices and m edges. Each edge has a *weight*, which is a positive integer, i.e., we have a function $w : E \to \mathbf{Z}^+$. We want to find a matching of maximum total weight.

We consider this problem on a special type of graphs, which we call 7-almost bipartite graphs. A graph $G = (V, E)$ is 7-almost bipartite, if there is a set $F \subseteq E$ of at most seven edges, such that $G' = (V, E - F)$ is bipartite, i.e., we can make G bipartite by removing at most seven edges.

Suppose we have an algorithm A that finds a maximum weighted matching in a bipartite graph with n vertices and m edges in $O(n \log n + nm)$ time.

Now, consider the following problem.

> **Given**: an undirected graph $G = (V, E)$, a weight function $w : E \subseteq \mathbf{Z}^+$, a set of edges $F \subseteq E$ such that $|F| \leq 7$ and $(V, E - F)$ is bipartite.
> **Question:** Find a matching of maximum total weight in G.

2

5. Fairy tale stable marriage (0.5 + 1.5 points)

King Neptune has a problem. His daughter Ariel is deeply in love with the handsome prince Eric. However, King Neptune is strongly against a marriage of Ariel and Eric (as Ariel is a mermaid and Eric is a human.)

He calls his royal advisor Sebastian. Now, according to the laws of the undersea kingdom, marriages are determined at a royal ball. The king must invite Ariel and Eric at the ball, but he may also invite other mermen and mermaids, but always the same number of mermen as of mermaids. The Gale-Shapley algorithm is carried out, and a stable marriage is thus formed, matching each of the mermen and Eric with a mermaid.

The king and Sebastian can bribe the other participants of the ball in making their preference list in any order the king and Sebastian would want. And, they can choose in which order the Gale-Shapley algorithm is carried out.

Actually, they can modify everything they want, with the following exceptions:

- Ariel has Eric at the top of her preference list.
- Eric has Ariel at the top of his preference list.
- The Gale-Shapley algorithm is used.

1. **Is it possible for the king and Sebastian to arrange things such that Eric and Ariel are not matched, or is a marriage between Ariel and Eric unavoidable?** (YES or NO.)
2. **Prove your answer.**

1. **Give an algorithm** that solves this problem efficiently. (You may use algorithm A as a subroutine.)

2. **Explain why your algorithm is correct.**

3. **How much time does your algorithm use? Explain this.** (You may use the stated time for algorithm A as a fact, without proving it.)

4. Investor in vegetables (1+1 points)

An investor has a large store-room, in which he can store vegetables. At any point in time, he can store at most D units. At certain days, a farmer offers some vegetables for sale; and at certain days in time, a food factory wants to buy some vegetables.

The investor can buy all or some of these vegetables from the farmer, then keep these in his store room (but he should at no moment have more than D units in his store room), and then sell these to the factory.

There are n types of vegetables $v_1, \ldots, v_n$.

We assume that each type of vegetable is offered at exactly one day, which we call d_i. (E.g., if at some day, brussels sprouts are offered, then this is the only day that brussels sprouts are offered.) The number of units offered from vegetable type v_i at day d_i is exactly n_i; the investor can buy any amount between 0 and n_i units. The farmer asks for this vegetable e_i euros per unit.

We also assume that there is exactly one day on which the food factory wants to buy vegetable type v_i; this day is called t_i. The food factory wants to buy m_i units at this day, and will pay p_i euros per unit. The investor thus can sell between 0 and m_i units of vegetable v_i at day t_i.

1. Assume that storing food is free. Consider the problem to determine the best plan for buying and selling food, such that the profit of the investor is as large as possible, and the investor never buys or sells more than possible, and never has more that D units in his store room. **Show how this problem can be modeled as a minimum cost flow problem.**

2. Now suppose that storing the food is not free. Actually, the investor has two store rooms, one with capacity D_1 and one with capacity D_2; $D_1 + D_2 = D$. Storing a unit of food in store room one costs K_1 euros per day, and storing a unit of food in store room two costs K_2 euros per day. Again, **show how to model the problem to maximize profit can be modeled as a minimum cost flow problem.** (It is sufficient if you explain how this differs from part 1.)

If you want, you can explain the construction with help of a small but clear example.