Algorithmics II (H)

Tutorial Exercises on Graph and Matching Algorithms

1. Let $G=(V, E)$ be a bipartite graph, where $n=|V|$ and $m=|E|$. Assume that $V=U\cup W$, where $\{u, w\}\in E$ implies that $u\in U$ and $w\in W$. Let $R\subseteq U$ be a set of vertices on the left-hand side of G that are coloured red. A *maximum red-matching* M is a maximum (cardinality) matching in G that matches as many red vertices as possible. That is, M is a maximum matching in G such that, over all maximum matchings in G, M matches the largest number of red vertices. Show how to find a maximum red-matching in $O(n(n+m))$ time.

2. Let $S=\{s_1, s_2,\ldots, s_p\}$ be a set of school-leavers and let $U=\{u_1, u_2,\ldots, u_q\}$ be a set of universities. Assume that each school-leaver $s_i\in S$ finds acceptable a subset of the universities, denoted by A_i. Assume further that each university $u_j\in U$ has c_j places. Let $L=|A_1|+|A_2|+\ldots+|A_p|$ and let $C=\max\{c_j : u_j\in U\}$. A *feasible matching* is an assignment of school-leavers to universities such that:

   (i) each school-leaver is assigned to at most one university;
   (ii) a school-leaver is never assigned to an unacceptable university;
   (iii) each university $u_j\in U$ is assigned at most c_j school-leavers.

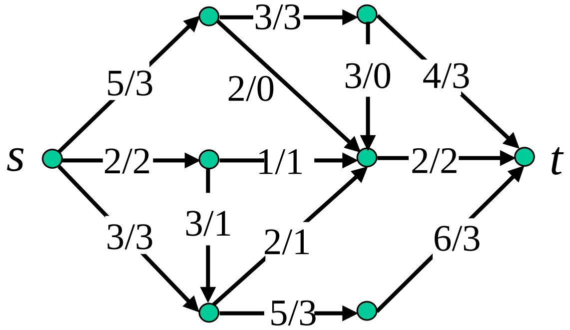
   Show how to construct a maximum feasible matching of school-leavers to universities in $O(p(p+CL))$ time.

3. Let $S=\{s_1, s_2,\ldots, s_p\}$ be a set of students and let $P=\{p_1, p_2,\ldots, p_q\}$ be a set of projects. Assume that each student $s_i\in S$ finds acceptable a subset of the projects, denoted by A_i. Now suppose that $s < t \leq q$ and that the projects $P'=\{p_1, p_2,\ldots, p_t\}$ are *restricted*, i.e. at most s projects from P' can run. Let $n=p+q$ and $m=|A_1|+|A_2|+\ldots+|A_p|$. A *feasible matching* is an assignment of students to projects such that:

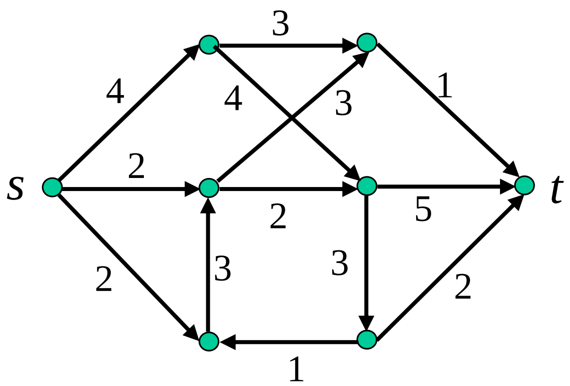
   (i) each student is assigned to at most one project;
   (ii) a student is never assigned to an unacceptable project;
   (iii) each project is assigned at most one student;
   (iv) at most s projects from P' are assigned a student.

   Show how to construct a maximum feasible matching of students to projects in $O(p(n+m+t(t-s)))$ time.

4. Prove that the flow in the following network G is maximum, or else find a larger flow. (For each pair of integers a/b adjacent to an edge e in G, a denotes the capacity of e and b denotes the flow along e. Also, s is the source of G and t is the sink of G.)



5. Find a maximum flow in the following network G. (Each integer adjacent to an edge in G denotes the capacity of that edge. Also, s is the source of G and t is the sink of G.) Prove that the flow is maximum by identifying an appropriate cut.



6. Recall from the lectures that the maximum matching problem in bipartite graphs can be reduced to the network flow problem, as follows. Let $G=(V,E)$ be a bipartite graph, and suppose that G has bipartition $V=U\cup W$. Form a network $G'=(V',E')$ as follows: let $V'=V\cup\{s,t\}$ for two new vertices s, t, and let

$$\begin{aligned} E' = {} & \{(s,u) : u\in U\}\cup \\ & \{(u,w) : u\in U \text{ and } w\in W \text{ and } \{u,w\}\in E\}\cup \\ & \{(w,t) : w\in W\}. \end{aligned}$$

Let s, t be the source and sink of G' respectively. All edges in G' have capacity 1. Show that a maximum flow in G' corresponds to a maximum matching in G.

7. Let $G'=(V',E')$ be the network constructed in Question 6, given a bipartite graph $G=(V,E)$ with bipartition $V=U\cup W$. Show that $2 \min(|U|,|W|)+1$ is an upper bound on the length (i.e. number of edges) of any augmenting path found during an execution of the Ford-Fulkerson algorithm as applied to G'.

8. Recall from the lectures that in an instance of the network flow problem, we assumed that every vertex lies on a directed path from the source s to the sink t in the network G. Show that we lose no generality by making this assumption. In other words, give a reduction from the case that G contains vertices not on a directed path from s to t, to the case where such vertices do not occur.

9. Find the student-optimal and lecturer-optimal stable matchings in the following instance of the stable marriage problem:

s_1:	l_4	l_5	l_3	l_2	l_6	l_1	l_1:	s_1	s_5	s_4	s_2	s_6	s_3
s_2:	l_5	l_6	l_3	l_2	l_4	l_1	l_2:	s_2	s_6	s_1	s_3	s_5	s_4
s_3:	l_6	l_3	l_2	l_1	l_4	l_5	l_3:	s_6	s_2	s_4	s_3	s_1	s_5
s_4:	l_2	l_3	l_5	l_6	l_4	l_1	l_4:	s_4	s_2	s_3	s_5	s_6	s_1
s_5:	l_3	l_2	l_5	l_4	l_6	l_1	l_5:	s_3	s_6	s_4	s_5	s_1	s_2
s_6:	l_6	l_4	l_1	l_2	l_5	l_3	l_6:	s_4	s_1	s_5	s_2	s_3	s_6

Students' preferences | Lecturers' preferences

10. Prove the following facts concerning an execution of the student-oriented Gale/Shapley algorithm when applied to a stable marriage instance with n students and n lecturers:

    (a) If a student s applies to a lecturer l, then there is no stable matching in which s has a better partner than l.
    (b) If a lecturer l receives an application from a student s, then there is no stable matching in which l has a worse partner than s.
    (c) The algorithm terminates when the last lecturer to receive an application his/her first application.
    (d) The total number of applications (apply operations) during this execution is at most $n^2 - n + 1$.

11. The Stable Marriage problem with Ties (SMT) is defined as follows. We are given a set of n students and a set of n lecturers. Each person ranks all members of the opposite "side" in order of preference, but may include ties in his/her preference list. A matching M is *stable* if there is no student s and lecturer l, each of whom strictly prefers the other to his/her partner in M. (That is, l should not belong to the same tie as s's partner, if such a tie exists, and vice versa.) Show that every instance I of SMT admits a stable matching, and moreover give an $O(n^2)$ algorithm for finding a stable matching in I.

12. Give an example instance I of SMT satisfying the property that there is no student-optimal stable matching in I.

13. Let $G=(V,E)$ be a weighted directed graph, where $\{v_1,v_2,\ldots,v_n\}$ and the weight of the edge (v_i,v_j) is an integer denoted by $wt(v_i,v_j)$. Prove that the assignment of values to $D_k(i,j)$ throughout the course of the execution of the Floyd-Warshall algorithm maintains the following invariant: $D_k(i,j)$ is the length of the shortest path from v_i to v_j whose intermediate vertices belong to $\{v_1,\ldots,v_k\}$, for any i, j and k where $v_i \in V$, $v_j \in V$ and $0 \leq k \leq n$.

14. Let G be the weighted directed graph shown below, where the weight of each edge is as shown. Using the Floyd-Warshall algorithm, compute the length of a shortest path, and a representation of an actual shortest path, between each pair of vertices in G. Show your working at each step.