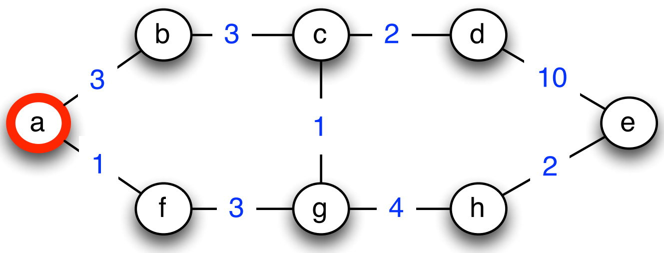
Master 2 Graph Algorithms and Combinatorial Optimization
Final Exam, November 2019
3 hours
No documents are allowed. No computers, cellphones.

Instruction and comments: the points awarded for your answer will be based on the correctness of your answer as well as the clarity of the main steps in your reasoning. All proposed solutions must be proved. **All the exercises are independent.** *The points are indicated so you may adapt your effort (***there are 22 points in total, the mark will be on 20, i.e., you may have 22/20***).* **The total estimated time is 2h55 minutes, meaning that you will have 5 minutes to be sure that you have indicated your name on every sheet of paper!**

Exercise 1 (Dijkstra. (2 points, 15 minutes)) Consider the following graph H.



Give the definition of a shortest path tree rooted in a in the graph H.

Applying the Dijkstra's algorithm on H, compute a shortest path tree rooted in a. The five first steps of the algorithm must be detailed (**at most two lines per steps**).

In particular, indicate the order in which vertices are considered during the execution of the algorithm, and give the final solution.

Exercise 2 (Flow with vertex capacity. (5 points, 45 minutes)) Let $\mathcal{N}$ be the network flow defined by a directed graph $D = (V, A)$, with a source $s \in V$ and a target $t \in V$, and an **integral** capacity function $c : A \to \mathbb{N}$ over the arcs.

1. Give the definition of a flow function from s to t in $\mathcal{N}$.

2. Let $f : A \to \mathbb{R}^+$ be a flow, give the definition of the value $v(f)$ of f.

3. Prove that, because the capacities are integers, the maximum value of a flow between s and t is an integer, and there exists a flow $f : A \to \mathbb{N}$ (i.e., $f(a)$ is an integer for all $a \in A$) with maximum value.

As a concrete example, consider the network flow (with source s and target t) and the initial flow f_0 described in Figure 1.

4. Prove that the initial flow f_0 (in red) is actually a flow and give its value.

5. Applying the Ford-Fulkerson's algorithm, **starting from the initial given flow** f_0, compute a maximum flow from s to t. For each iteration of the algorithm, you must give the **auxiliary digraph, the path on which the flow will be increased and the resulting flow after the iteration.**

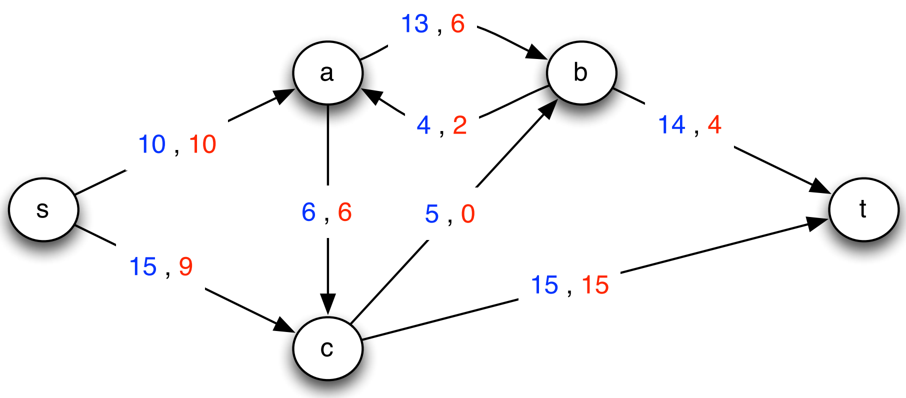


Figure 1: A network flow $\mathcal{N}$ with an initial flow f_0 (in red). The integers in blue denote the edge-capacities. The integers in red denote the value (on each arc) of the initial flow f_0.

6. Give a minimum s-t cut and explain why this provides a certificate proving that the flow that you have computed is maximum.

We now add a new constraint by giving some capacities to the vertices. That is, for every $v \in V$, let $c(v)$ be its capacity. A flow (with vertex-capacity) must moreover satisfies that, for every vertex $v \in V$, the flow leaving v must be at most $c(v)$, i.e., $\sum_{w \in N^+(v)} f(vw) \leq c(v)$ for every $v \in V$. To solve this new problem (to compute a maximum flow with vertex capacities), we will show that it can be modeled as a "normal" flow problem (without vertex capacities).

Let $\mathcal{N} = (D = (V, A), s, t \in V, c : A \cup V \to \mathbb{N})$ be a network flow with extra capacities on the vertices. Let us build the "classical" network flow $\mathcal{N}^* = (D^* = (V^*, A^*), s^-, t^+ \in V^*, c^* : A^* \to \mathbb{N})$ as follows (see Figure 2 for an illustration). For every vertex $v \in V$, create two vertices v^+ and v^- in V^* with an arc $v^-v^+ \in A^*$ with capacity $c^*(v^-v^+) = c(v)$. Moreover, for every $w \in N^-(v)$, add an arc between w^+ and v^- with capacity $c^*(w^+v^-) = c(wv)$, and for every $w \in N^+(v)$, add an arc between v^+ and w^- with capacity $c^*(v^+w^-) = c(vw)$.

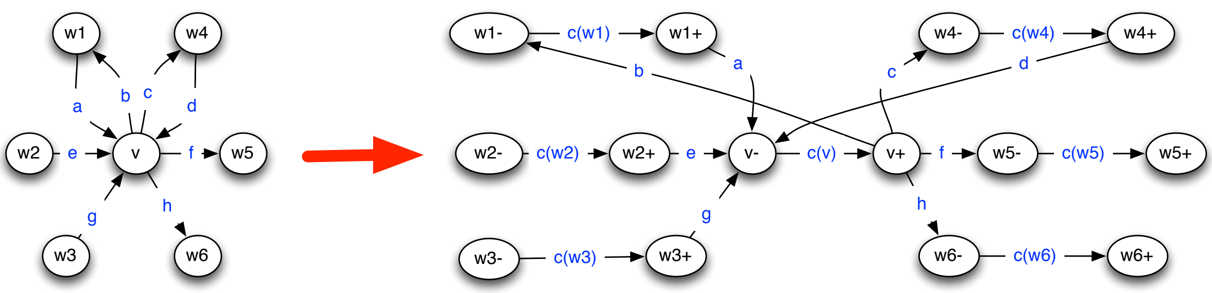


Figure 2: Scheme of the transformation from the network $\mathcal{N}$ (left) to network $\mathcal{N}^*$ (right) around the vertex v.

7. Consider the network flow $\mathcal{N}$ depicted in Figure 1 (omitting the initial flow f_0) with the additional vertex capacities: $c(s) = c(t) = \infty$, $c(a) = 7$, $c(b) = 12$ and $c(c) = 25$. Draw

the network flow $\mathcal{N}^*$ (without vertex capacities) obtained from $\mathcal{N}$ by the transformation described above.

8. Given a general network flow $\mathcal{N}$ (with vertex capacities) and the transformed network $\mathcal{N}^*$ defined above, prove that there is a bijection between the s^--t^+ flows in $\mathcal{N}^*$ and the s-t flows in the network flow $\mathcal{N}$ with vertex capacities.

9. Compute a maximum flow in the network flow $\mathcal{N}$ with vertex capacities defined in question 7.

Exercise 3 (Baseball game. (4 points, 40 minutes)) We are in the middle of the baseball championship. Note that, when two teams meet, there is a unique winner (no ties) and the winning team gains one point (the loosing team gains 0 point).

There are four teams in this championship: Yale (Y), Harvard (H), Cornell (C), and Brown (B). Moreover, the current status of the championship is described by the table below.

Team	*Points*	*To play*	*Yale*	*Harvard*	*Cornell*	*Brown*
Yale	32	8		1	6	1
Harvard	29	4	1		0	3
Cornell	28	7	6	0		1
Brown	27	5	1	3	1	

This table must be read as follows. The row corresponding to Yale says that this team already won 32 points, the Yale team has still 8 remaining games to be played: 1 against the Harvard team, 6 against Cornell, and one against Brown.

The question is to know whether Harvard can win the championship (i.e., be the unique team with maximum number of points at the end of the season).

1. Show that Harvard can win the championship only if Harvard wins all its 4 remaining games, Yale looses all its remaining games, and Cornell wins at most 4 of its remaining games.

To decide whether Harvard can be the unique winner, we first assume that Harvard wins all its remaining games (since this is a necessary condition by previous question). Then, we model the problem as a flow problem in the network flow illustrated in Figure 3.

2. Let $X \in \{Yale\text{-}Cornell, Yale\text{-}Brown, Brown\text{-}Cornell\}$. What does the capacity $c(sX)$ of the arc sX represent in terms of our baseball problem? Let $Y \in \{Yale, Cornell, Brown\}$. What does the capacity $c(Yt)$ of the arc Yt represent?

3. Let us consider one unit of flow following the path (s, X, Y, t) where $X \in \{Yale\text{-}Cornell, Yale\text{-}Brown, Brown\text{-}Cornell\}$ and $Y \in \{Yale, Cornell, Brown\}$. What does it represent in terms of our baseball problem?

4. Can Harvard be the champion at the end of the season? Explain your solution.

The goal of the last question is to generalize the above example.

5. Suppose we are in the middle of a baseball season where each team T_i , $1 \leq i \leq n$ has won $w(i)$ games so far and thus has $w(i)$ points. Let $G_1, G_2, \cdots, G_k$ be the schedule of the remaining games, where each G_i is an unordered pair of teams.

   Given T_i , $w(i)$, $1 \leq i \leq n$, and $G_1, G_2, \cdots, G_k$, give a way to predict that T_1 does or does not have a chance to have the top score at the end of the season? If T_1 has a chance of

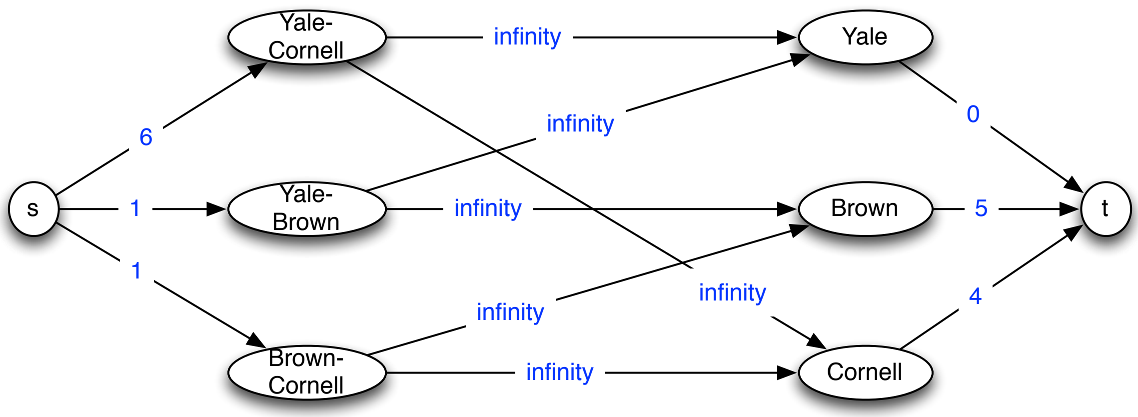


Figure 3: A network flow (arc's capacity in blue) modeling part of the remaining championship.

being champion, how can we find a sequence of outcomes (i.e. results of $G_1, G_2, \cdots, G_k$) such that T_1 reaches the top rank at the end of the season?

Exercise 4 (Dominating set. (2 points, 15 minutes)) A *dominating set* in a graph $G = (V, E)$ is a set of vertices $S \in V$ such that each vertex which is not in S has at least one neighbor in S. A minimum dominating set of G is a dominating set of minimum cardinality.

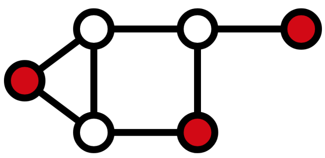


Figure 4: Vertices in red form a dominating set of the graph.

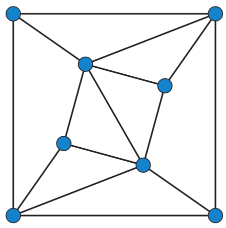


Figure 5: A graph.

1. Give a minimum dominating set of the graph in Figure 5.
2. Model the problem of finding a minimum dominating set as a linear program.

Exercise 5 (Diameter. (3 points, 20 minutes)) We consider a directed graph $G = (V, A)$ and a length function $w : A \to \mathcal{R}$, where $w(a)$, for $a \in A$, is the length of arc a. The *distance* between two vertices in G is the sum of the lengths of all arcs in a shortest path connecting them. The *diameter* of a graph is the greatest distance between any pair of vertices of the graph.

1. What computes the following linear program? Explain clearly the role of the variables, the meaning of the objective function and the constraints, the role of s and t.

   Given a directed graph (V, A) with two distinguished nodes s and t, and a length w_{ij} for each edge $(i, j) \in A$, consider the program with binary variables x_{ij}.

   minimize $\sum_{ij\in A} w_{ij}x_{ij}$ subject to $x \geq 0$ and for all i, $\sum_j x_{ij} - \sum_j x_{ji} = \begin{cases} 1, & \text{if } i = s; \\ -1, & \text{if } i = t; \\ 0, & \text{otherwise.} \end{cases}$.

2. Write a linear programme which computes the diameter of a graph.

Exercise 6 (Feedback vertex set. (6 points, 40 minutes)) A *tournament* is a directed graph $G = (V, E)$, such that, for each pair of vertices, $u, v \in V$, exactly one of (u, v) and (v, u) is in E. See Figure 6 for an example of tournament with 4 vertices.

A *feedback vertex set* for a directed graph G is a subset of the vertices of G whose removal leaves an acyclic graph, i.e., a graph without **directed** cycles (see Figure 7 for an example of graph without directed cycle). An example of feedback vertex set is given in Figure 8. A minimum feedback vertex set is a feedback vertex set of minimum cardinality.

The goal of the exercise is to find a factor 3 approximation algorithm for the problem of finding a minimum feedback vertex set in a tournament.

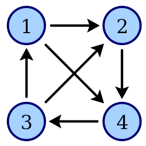


Figure 6: A tournament with 4 vertices.

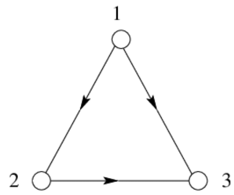


Figure 7: An acyclic tournament with 3 vertices.

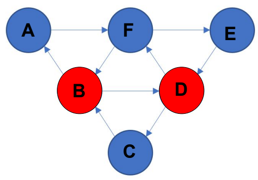


Figure 8: Vertices in red form a Feedback Vertex Set $F = \{B, D\}$ of the graph.

1. Give a tournament with 5 vertices.

2. Provide a minimum feedback vertex set of the tournament in Figure 6.

3. Show that, in a tournament, it is sufficient to "kill" all **directed** cycles of length 3 to obtain a feedback vertex set (in other works, a subset of the vertices of G whose removal makes all directed cycles of length 3 disappear also makes all directed cycles –of any length– disappear.).

4. Propose a simple algorithm with a complexity exponential in $|V|$ to get the minimum feedback vertex set of a tournament. Give its complexity.

We now want to get a polynomial time algorithm providing a good approximation of the minimum feedback vertex set problem. The approximation algorithm for *Set Cover* seen during the course will be the main building block of the algorithm. We remind the definition of the problem and the main seen result.

Reminder of the course. Given a universe U of n elements, a collection of subsets of U, $\mathcal{S} = \{S_1, ..., S_k\}$, and a cost function $c : \mathcal{S} \to Q^+$, the minimum set cover problem consists in finding a minimum cost subcollection of $\mathcal{S}$ that covers all elements of U. The *frequency* of an element is the number of sets it is in. The *frequency of the most frequent element* is denoted by f. The algorithm based on a linear program relaxation seen during the class provides a factor f approximation algorithm of the set cover problem, i.e. it finds in polynomial time a cover whose cost is less or equal to $f \cdot OPT$, where OPT is the cost of a minimum cover.

5. Model the problem of finding the minimum feedback vertex set as a minimum set cover problem (define well what is U, the subsets in $\mathcal{S}$, and their costs).

6. Give a factor 3 algorithm for the problem of finding a minimum feedback vertex set in a tournament.