ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Midterm Exam, Advanced Algorithms 2018-2019

- You are only allowed to have a handwritten A4 page written on both sides.
- Communication, calculators, cell phones, computers, etc... are not allowed.
- Your explanations should be clear enough and in sufficient detail that a fellow student can understand them. In particular, do not only give pseudo-code without explanations. A good guideline is that a description of an algorithm should be such that a fellow student can easily implement the algorithm following the description.
- **You are allowed to refer to material covered in the lecture notes** including theorems without reproving them. You may also refer to statements given in the exercise sheets and the homework sheet. You are however *not* allowed to refer to material from any specific solution to these exercises.
- **Do not touch until the start of the exam.**

Good luck!

Name: _____ **N° Sciper:** _____

Problem 1 / 20 points	Problem 2 / 14 points	Problem 3 / 22 points	Problem 4 / 22 points	Problem 5 / 22 points

Total / 100

1 *(20 pts)* **Duality of linear programming.** Consider the following linear program:

$$\begin{aligned} \text{Minimize} \quad & 6x_1 + 15x_2 + 8x_3 \\ \text{Subject to} \quad & 2x_1 + 6x_2 + x_3 \geq 3 \\ & x_1 + 2x_2 + 3x_3 \geq 4 \\ & x_1, x_2, x_3 \geq 0 \end{aligned}$$

Write down its dual and the complementarity slackness conditions.

2 *(14 pts)* **Karger's randomized algorithm for min-cut.** In class, we saw Karger's beautiful randomized algorithm for finding a minimum cut in an undirected graph $G = (V, E)$. Recall that his algorithm works by repeatedly contracting a randomly selected edge until the graph only consists of two vertices which define the returned cut. For general graphs, we showed that the returned cut is a minimum cut with probability at least $1/\binom{n}{2}$.

In this problem, we are going to analyze the algorithm in the special case when the input graph is a tree. Specifically, you should show that if the input graph $G = (V, E)$ is a spanning tree, then Karger's algorithm returns a minimum cut with probability 1.

(In this problem you are asked to show that Karger's min-cut algorithm returns a minimum cut with probability 1 *if the input graph is a spanning tree. Recall that you are allowed to refer to material covered in the lecture notes.)*

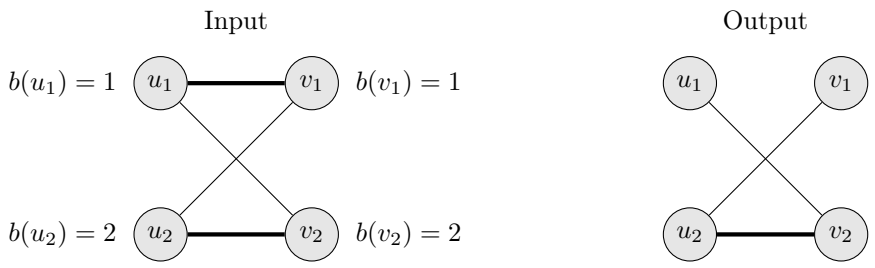
3 *(22 pts)* **Structure of extreme point solutions.** In this problem, we consider a generalization of the min-cost perfect matching problem. The generalization is called the *min-cost perfect b-matching problem* and is defined as follows:

Input: A graph $G = (V, E)$ with edge costs $c : E \to \mathbb{R}$ and degree bounds $b : V \to \{1, 2, \ldots, n\}$.

Output: A subset $F \subseteq E$ of minimum cost $\sum_{e \in F} c(e)$ such that for each vertex $v \in V$:

- The number of edges incident to v in F equals $b(v)$, i.e., $|\{e \in F : v \in e\}| = b(v)$.

Note that min-cost perfect matching problem is the special case when $b(v) = 1$ for all $v \in V$. An example with general b's is as follows:



On the left, we illustrate the input graph with the degree bounds (the b's). Thin and thick edges have cost 1 and 2, respectively. On the right, we illustrate a solution of cost $1 + 1 + 2 = 4$. It is a feasible solution since the degree of each vertex v equals $b(v)$ in the solution.

Your task is to prove the following statement: If the input graph $G = (V, E)$ is bipartite then any extreme point solution to the following linear programming relaxation (that has a variable x_e for every edge $e \in E$) is integral:

$$\begin{aligned} \textbf{Minimize} \quad & \sum_{e \in E} c(e) x_e \\ \textbf{subject to} \quad & \sum_{e \in E : v \in e} x_e = b(v) \qquad \text{for all } v \in V \\ & 0 \leq x_e \leq 1 \qquad \text{for all } e \in E. \end{aligned}$$

(In this problem you are asked to prove that every extreme point solution to the above linear program is integral assuming that the input graph G is bipartite. Recall that you are allowed to refer to material covered in the lecture notes.)

4 *(22 pts)* **Probabilistic analysis.** Consider the following algorithm RANDOM-CHECK that takes as input two subsets $S \subseteq E$ and $T \subseteq E$ of the same ground set E.

RANDOM-CHECK(S, T)

1. For each element $e \in E$, independently of other elements randomly set

$$x_e = \begin{cases} 1 & \text{with probability } 1/3 \\ 0 & \text{with probability } 2/3 \end{cases}$$

2. **if** $\sum_{e \in S} x_e = \sum_{e \in T} x_e$ **then**
3. **return** true
4. **else**
5. **return** false

Note that RANDOM-CHECK(S, T) returns true with probability 1 if $S = T$. Your task is to analyze the probability that the algorithm returns true if $S \neq T$. Specifically prove that RANDOM-CHECK(S, T) returns true with probability at most $2/3$ if $S \neq T$.

(In this problem you are asked to prove that RANDOM-CHECK*(S, T) returns true with probability at most $2/3$ if $S \neq T$. Recall that you are allowed to refer to material covered in the lecture notes.)*

5 *(22 pts)* **Matroids.** Design a polynomial-time algorithm for the matroid matching problem:

Input: A bipartite graph $G = (A \cup B, E)$ and two matroids $\mathcal{M}_A = (A, \mathcal{I}_A)$, $\mathcal{M}_B = (B, \mathcal{I}_B)$.

Output: A matching $M \subseteq E$ of maximum cardinality satisfying:

(i) the vertices $A' = \{a \in A : \text{there is a } b \in B \text{ such that } \{a, b\} \in M\}$ of A that are matched by M form an independent set in $\mathcal{M}_A$, i.e., $A' \in \mathcal{I}_A$; and

(ii) the vertices $B' = \{b \in B : \text{there is an } a \in A \text{ such that } \{a, b\} \in M\}$ of B that are matched by M form an independent set in $\mathcal{M}_B$, i.e., $B' \in \mathcal{I}_B$.

We assume that the independence oracles for both matroids $\mathcal{M}_A$ and $\mathcal{M}_B$ can be implemented in polynomial-time. Also to your help you may use the following fact without proving it.

Fact (obtaining a new matroid by copying elements). Let $\mathcal{M} = (N, \mathcal{I})$ be a matroid where $N = \{e_1, \ldots, e_n\}$ consists of n elements. Now, for each $i = 1, \ldots, n$, make k_i copies of e_i to obtain the new ground set

$$N' = \{e_1^{(1)}, e_1^{(2)}, \ldots, e_1^{(k_1)}, e_2^{(1)}, e_2^{(2)}, \ldots, e_2^{(k_2)}, \ldots, e_n^{(1)}, e_n^{(2)}, \ldots, e_n^{(k_n)}\},$$

where we denote the k_i copies of e_i by $e_i^{(1)}, e_i^{(2)}, \ldots, e_i^{(k_i)}$. Then $(N', \mathcal{I}')$ is a matroid where a subset $I' \subseteq N'$ is independent, i.e., $I' \in \mathcal{I}'$, if and only if the following conditions hold:

(i) I' contains at most one copy of each element, i.e., we have $|I' \cap \{e_i^{(1)}, \ldots, e_i^{(k_i)}\}| \leq 1$ for each $i = 1, \ldots, n$;

(ii) the original elements corresponding to the copies in I' form an independent set in $\mathcal{I}$, i.e., if $I' = \{e_{i_1}^{(j_1)}, e_{i_2}^{(j_2)}, \ldots, e_{i_\ell}^{(j_\ell)}\}$ then $\{e_{i_1}, e_{i_2}, \ldots, e_{i_\ell}\} \in \mathcal{I}$.

Moreover, if the independence oracle of $(N, \mathcal{I})$ can be implemented in polynomial time, then the independence oracle of $(N', \mathcal{I}')$ can be implemented in polynomial time.

(In this problem you are asked to design and analyze a polynomial-time algorithm for the matroid matching problem. You are allowed to use the above fact without any proof and to assume that all independence oracles can be implemented in polynomial time. Recall that you are allowed to refer to material covered in the lecture notes.)