Final 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.
- **Problems are not necessarily ordered by difficulty.**
- **Do not touch until the start of the exam.**

Good luck!

Name: ____________________ **N° Sciper:** ______________

Problem 1 / 10 points	Problem 2 / 12 points	Problem 3 / 15 points	Problem 4 / 20 points	Problem 5 / 20 points	Problem 6 / 23 points

Total / 100

1 *(10 pts)* **Simplex method.** Suppose we use the Simplex method to solve the following linear program:

$$\begin{aligned} \textbf{maximize} \quad & 4x_1 - 6x_2 + 4x_3 \\ \textbf{subject to} \quad & x_1 - 3x_2 + x_3 + s_1 = 1 \\ & x_1 + s_2 = 8 \\ & 3x_2 + 2x_3 + s_3 = 6 \\ & x_1,\ x_2,\ x_3,\ s_1,\ s_2,\ s_3 \geq 0 \end{aligned}$$

At the current step, we have the following Simplex tableau:

$$\begin{aligned} x_1 &= 1 + 3x_2 - x_3 - s_1 \\ s_2 &= 7 - 3x_2 + x_3 + s_1 \\ s_3 &= 6 - 3x_2 - 2x_3 \\ \hline z &= 4 + 6x_2 - 4s_1 \end{aligned}$$

Write the tableau obtained by executing one iteration (pivot) of the Simplex method starting from the above tableau.

Solution:

At the current step, we have the following Simplex tableau:

$$\begin{aligned} x_1 &= 1 + 3x_2 - x_3 - s_1 && (1) \\ s_2 &= 7 - 3x_2 + x_3 + s_1 && (2) \\ s_3 &= 6 - 3x_2 - 2x_3 && (3) \\ \hline z &= 4 + 6x_2 - 4s_1 \\ x_1 &:= 1\ x_2 := 0\ x_3 := 0\ s_1 := 0\ s_2 := 7\ s_3 := 6 \end{aligned}$$

Only x_2 has a positive coefficient in z, we will pivot x_2. We have
$\nearrow x_2 \longrightarrow\ x_2 \leq\ \infty\ (1),\ x_2 \leq 7/3\ (2),\ x_2 \leq 6/3\ (3) \longrightarrow x_2 := 2,\ s_3 := 0$

$$\begin{aligned} x_1 &= 7 - 3x_3 - s_1 - s_3 \\ s_2 &= 1 + 3x_3 + s_1 + s_3 \\ x_2 &= 2 - 2x_3/3 - s_3/3 \\ \hline z &= 16 - 4x_3 - 2s_3 - 4s_1 \\ x_1 &:= 7\ x_2 := 2\ x_3 := 0\ s_1 := 0\ s_2 := 1\ s_3 := 0 \end{aligned}$$

2 *(12 pts)* **Improving Professor Ueli von Gruyères' estimator.** Last year Professor Ueli von Gruyères worked hard to to obtain an estimator $\mathcal{A}$ to estimate the total cheese consumption of fondue lovers in Switzerland. For a small $\epsilon > 0$, his estimator $\mathcal{A}$ only asks $3/\epsilon^2$ random persons and have the following guarantee: if we let W denote the true answer and let X be the random output of $\mathcal{A}$ then

$$\Pr[|X - W| \geq \epsilon W] \leq 1/3\,.$$

However, Ueli is now stuck because the error probability of 1/3 is too high. We are therefore going to help Ueli by designing a new estimator with a much higher success probability while still only asking relatively few persons.

For a fixed small parameter $\delta > 0$, your task is to design and analyze an estimator that outputs a random value Y with the following guarantee:

$$\Pr[|Y - W| \geq \epsilon W] \leq \delta\,. \tag{1}$$

Your estimator should ask at most $3000 \log(1/\delta)/\epsilon^2$ persons about their preferences.

While you should explain why your estimator works and what tools to use to analyze it, *you do not need to do any detailed calculations.*

(In this problem you are asked to (i) design an estimator that asks at most $3000 \log(1/\delta)/\epsilon^2$ *persons and (ii) explain why it satisfies the guarantee* (1)*. Recall that you are allowed to refer to material covered in the lecture notes.)*

Solution: We define our estimator as follows:

- Let $t = 1000 \log(1/\delta)$.
- Run t independent copies of $\mathcal{A}$ to obtain estimates $X_1, X_2, \ldots, X_t$.
- Output Y to be the *median* of $X_1, \ldots, X_t$.

Let I_i be the indicator random variable that $|X_i - W| \geq \epsilon W$. For us to have $|Y - W| \geq \epsilon W$ it must be that $\sum_{i=1}^t I_i \geq t/2$. However, $\mathbb{E}[\sum_{i=1}^t I_i] \leq t/3$ and it is a sum of *independent* random variables taking values in $\{0, 1\}$. We can thus apply Chernoff bounds to obtain

$$\Pr[|Y - W| \geq \epsilon W] \leq \Pr[\sum_{i=1}^{t} I_i \geq t/2] \leq e^{-t/100} \leq \delta\,,$$

where we used that $t = 1000 \log(1/\delta)$.

3 *(15 pts)* **Spectral graph theory.** Consider a d-regular undirected graph $G = (V, E)$ and let M be its normalized adjacency matrix. As seen in class, M has $n = |V|$ eigenvalues $1 = \lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n \geq -1$ and the corresponding eigenvectors $v_1, v_2, \ldots, v_n \in \mathbb{R}^n$ can be selected to be orthogonal vectors where

$$v_1 = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \text{ is the all one vector.}$$

Assuming that $\lambda_2 = 1$, your task is to design a procedure FindDisconnectedSet(v_2) that takes as input the second eigenvector and outputs a non-empty subset $S \subsetneq V$ of the vertices such that there is no edge crossing the cut defined by S. In other words, the output S must satisfy $S \neq \emptyset, S \neq V$ and any edge $e \in E$ has either both endpoints in S or both endpoints in $V \setminus S$.

We remark that your procedure FindDisconnectedSet does **not** know the edgeset E of the graph. Thus it needs to define the set S only based on the values $v_2(i)$ the second eigenvector assigns to every vertex $i \in V$.

(In this problem you are asked to (i) design the algorithm FindDisconnectedSet *and (ii) argue that it outputs a non-empty $S \subsetneq V$ that cuts 0 edges assuming $\lambda_2 = 1$. Recall that you are allowed to refer to material covered in the lecture notes.)*

Solution:

Let $S = \{i \in V : v_2(i) \leq 0\}$. Note that $S \neq \emptyset$ and $S \neq V$ since $v_2 \perp v_1$ and $v_2 \neq \mathbf{0}$.

In class we saw that if $\lambda_2 = 1$ then all vertices in a connected component must receive the same value by the second eigenvector v_2. In particular adjacent vertices receive the same value. It follows that no vertex i with $v_2(i) \leq 0$ is adjacent to a vertex j with $v_2(j) > 0$ and so no edges crosses the cut defined by S.

For a different proof that doesn't rely on orthogonality note that it is enough to choose S to be the set of all vertices with their v_2 values equal to some value (that occurs in v_2). For example $S = \{i \in V : v_2(i) = v_2(1)\}$ - this is the set of all vertices whose second eigenvalue is equal to the one of vertex 1. Now $S \neq \emptyset$ since the vertex 1 is contained in S. Also $S \neq V$ since that would mean that all vertices have the same eigenvalue, but this cannot happen since $v_2 \neq v_1$ (more precisely v_2 cannot be a multiple of v_1). From what was written above, all vertices $v \notin S$ have eigenvalues different from $v_1(1)$, and as such cannot be connected to vertices in S. This means that S defines a cut, as requested.

Note that the spectral graph partitioning algorithm seen in class uses the edges of the graph so doesn't work directly.

(This page is intentionally left blank.)

4 *(consisting of subproblems **a-b**, 20 pts)* **Online algorithms.** You have 1 Euro and your goal is to exchange it to Swiss francs during the next two consecutive days. The exchange rate is an arbitrary function from days to real numbers from the interval $[1, W^2]$, where $W \geq 1$ is known to the algorithm.

More precisely, at day 1, you learn the exchange rate $x_1 \in [1, W^2]$, where x_1 is the amount of Swiss francs you can buy from 1 Euro. You then need to decide between the following two options:

(i) Trade the whole 1 Euro at day 1 and receive x_1 Swiss francs.

(ii) Wait and trade the whole 1 Euro at day 2 at exchange rate $x_2 \in [1, W^2]$. The exchange rate x_2 is known only at day 2, i.e., after you made your decision at day 1.

In the following two subproblems, we will analyze the competitive ratio of optimal deterministic algorithms. Recall that we say that an online algorithm is c-competitive if, for any $x_1, x_2 \in [1, W^2]$, it exchanges the 1 Euro into at least $c \cdot \max\{x_1, x_2\}$ Swiss francs.

4a *(10 pts)* Give a deterministic algorithm with a competitive ratio of $1/W$.

(In this problem you are asked to (i) design a deterministic online algorithm for the above problem and (ii) to prove that your algorithm is $1/W$-competitive. Recall that you are allowed to refer to material covered in the lecture notes.)

Solution: The algorithm is as follows:

- If $x_1 \geq W$, then do the exchange on day 1 and receive x_1 Swiss francs.
- Otherwise, do the exchange on day 2 and receive x_2 Swiss francs.

We now analyze its competitiveness. If $x_1 \geq W$, then our algorithm gets at least W Swiss francs. Optimum is at most W^2 and so we are $1/W$ competitive. Otherwise if $x_1 < W$ then we get $x_2 \geq 1$ Swiss francs which is $x_2/\max(x_2, x_1) \geq 1/W$ competitive.

4b *(10 pts)* Show that any deterministic algorithm has a competitive ratio of at most $1/W$.

(In this problem you are asked to prove that any deterministic algorithm has a competitive ratio of at most $1/W$ for the above problem. Recall that you are allowed to refer to material covered in the lecture notes.)

Solution: Consider an deterministic online algorithm $\mathcal{A}$ and set $x_1 = W$. There are two cases depending on whether $\mathcal{A}$ trades the 1 Euro the first day or not. Suppose first that $\mathcal{A}$ trades the Euro at day 1. Then we set $x_2 = W^2$ and so the algorithm is only $W/W^2 = 1/W$ competitive. For the other case when $\mathcal{A}$ waits for the second day, we set $x_2 = 1$. Then $\mathcal{A}$ gets 1 Swiss franc whereas optimum would get W and so the algorithm is only $1/W$ competitive again.

5 *(20 pts)* **Streaming algorithms.** In the following problem Alice holds a string $x = \langle x_1, x_2, \ldots, x_n\rangle$ and Bob holds a string $y = \langle y_1, y_2, \ldots, y_n\rangle$. Both strings are of length n and $x_i, y_i \in \{1, 2, \ldots, n\}$ for $i = 1, 2, \ldots, n$. The goal is for Alice and Bob to use little communication to estimate the quantity

$$Q = \sum_{i=1}^{n} (x_i + y_i)^2 \,.$$

A trivial solution is for Alice to transfer all of her string x to Bob who then computes Q exactly. However this requires Alice to send $\Theta(n \log n)$ bits of information to Bob. In the following, we use randomization and approximation to achieve a huge improvement on the number of bits transferred from Alice to Bob. Indeed, for a small parameter $\epsilon > 0$, your task is to devise and analyze a protocol of the following type:

- On input x, Alice uses a randomized algorithm to compute a message m that consists of $O(\log(n)/\epsilon^2)$ bits. She then transmits the message m to Bob.
- Bob then, as a function of y and the message m, computes an estimate Z.

Your protocol should ensure that

$$\Pr[|Z - Q| \geq \epsilon Q] \leq 1/3 \,, \tag{2}$$

where the probability is over the randomness used by Alice.

(In this problem you are asked to (i) explain how Alice computes the message m of $O(\log(n)/\epsilon^2)$ bits (ii) explain how Bob calculates the estimate Z, and (iii) prove that the calculated estimate satisfies (2). Recall that you are allowed to refer to material covered in the lecture notes.)

Solution: We use the idea of the AMS algorithm. We first describe how Alice Alice calculates the message m. Let $\mathcal{A}$ be the following procedure:

- Select a random $h : [n] \to \{\pm 1\}$ 4-wise independent hash function. h takes $O(\log n)$ bits to store.
- Calculate $A = \sum_{i=1}^{n} h(i) x_i$.

Let $t = 6/\epsilon^2$. Alice runs $\mathcal{A}$ t times. Let h_i and A_i be the hash function and the quantity calculated by i:th invokation of $\mathcal{A}$. Then Alice transmits the information $h_1, A_1, h_2, A_2, \ldots, h_t, A_t$ to Bob. Note that each h_i takes $O(\log n)$ bits to store and each A_i is an integer between $-n^2$ and n^2 and so it also takes $O(\log n)$ bits to store. Therefore the message Alice transmits to Bob is $O(\log(n)/\epsilon^2)$ bits.

Now Bob calculates the estimate Z as follows:

- For $\ell = 1, 2, \ldots, t$, let $Z_\ell = A_\ell + \sum_{i=1}^{n} h_\ell(i) y_i$.
- Output $Z = \frac{\sum_{\ell=1}^{t} Z_\ell^2}{t}$.

To prove that Z satisfies (2), we first analyze a single Z_ℓ. First, note that $Z_\ell = A_\ell + \sum_{i=1}^{n} h_\ell(i) y_i = \sum_{i=1}^{n} h_\ell(i)(x_i + y_i) = \sum_{i=1}^{n} h_\ell(i) f_i$, where we let $f_i = x_i + y_i$. And so $Z_\ell = \sum_{i=1}^{n} h_\ell(i) f_i$ where h_ℓ is a random 4-wise independent hash function. This is exactly the setting of the analysis of the AMS streaming algorithm seen in class. And so over the random selection of the hash function, we know that

$$\mathbb{E}[Z_\ell^2] = \sum_{i=1}^{n} f_i^2 = Q$$

and

$$\mathrm{Var}[Z_\ell^2] \leq 2\left(\sum_{i=1}^{n} f_i^2\right)^2 = 2Q^2\,.$$

Therefore, we have that

$$\mathbb{E}[Z] = Q \qquad \text{and} \qquad \mathrm{Var}[Z] \leq \frac{2Q^2}{t}\,.$$

So by Chebychev's inequality

$$\Pr[|Z - Q| \geq \epsilon Q] \leq \frac{2Q^2/t}{\epsilon^2 Q^2} \leq 1/3\,,$$

by the selection of $t = 6/\epsilon^2$.

(This page is intentionally left blank.)

6 *(consisting of subproblems* ***a-b****, 23 pts)* **Submodular vertex cover.** In this problem, we give a 2-approximation algorithm for the submodular vertex cover problem which is a generalization of the classic vertex cover problem seen in class. We first, in subproblem **(a)**, give a new rounding for the classic vertex cover problem and then give the algorithm for the more general problem in subproblem **(b)**.

6a *(11 pts)* Recall that a vertex cover instance is specified by an undirected graph $G = (V, E)$ and non-negative vertex-weights $w : V \to \mathbb{R}_+$. The task is to find a vertex cover $S \subseteq V$ of minimum total weight $\sum_{i\in S} w(i)$, where a subset $S \subseteq V$ of the vertices is said to be a vertex cover if for every edge $\{i, j\} \in E$, $i \in S$ or $j \in S$. The natural LP relaxation (as seen in class) is as follows:

$$\begin{array}{lll} \textbf{minimize} & \displaystyle\sum_{i\in V} w(i)x_i & \\ \textbf{subject to} & x_i + x_j \geq 1 & \text{for } \{i,j\} \in E \\ & x_i \geq 0 & \text{for } i \in V \end{array}$$

Given a fractional solution x to the above linear program, a natural approach to solve the vertex cover problem is to give a rounding algorithm. Indeed, in class we analyzed a simple rounding scheme: output the vertex cover $S = \{i \in V : x_i \geq 1/2\}$. We proved that $w(S) \leq 2\sum_{i\in V} w(i)x_i$.

In this subproblem, your task is to prove that the following alternative randomized rounding scheme give the same guarantee in expectation. The randomized rounding scheme is as follows:

- Select $t \in [0, 1/2]$ uniformly at random.
- Output $S_t = \{i \in V : x_i \geq t\}$.

Prove (i) that the output S_t is a feasible vertex cover solution (for any $t \in [0, 1/2]$) and (ii) that $\mathbb{E}[\sum_{i\in S_t} w(i)] \leq 2 \cdot \sum_{i\in V} w(i)x_i$ where the expectation is over the random choice of t. We remark that you *cannot* say that x is half-integral as x may not be an extreme point solution to the linear program.

(In this problem you are asked to prove that the randomized rounding scheme (i) always outputs a feasible solution and (ii) the expected cost of the output solution is at most twice the cost of the linear programming solution. Recall that you are allowed to refer to material covered in the lecture notes.)

Solution:

First, the output is always feasible since we always include all vertices with $x_i \geq 1/2$ which is a feasible vertex cover as seen in class. We proceed to analyze the approximation guarantee. Let X_i be the indicator random variable that i is in the output vertex cover. Then $\Pr[X_i = 1]$ is equal to the probability that $t \leq x_i$ which is 1 if $x_i \geq 1/2$ and otherwise it is $x_i/(1/2) = 2x_i$. We thus always have that $\Pr[X_i = 1] \leq 2x_i$. Hence,

$$\mathbb{E}[\sum_{i\in S_t} w(i)] = \mathbb{E}[\sum_{i\in V} X_i w(i)] = \sum_{i\in V} \mathbb{E}[X_i]w(i) \leq 2\sum_{i\in V} x_i w(i)\,.$$

(This page is intentionally left blank.)

6b *(12 pts)* Design and analyze a *deterministic* 2-approximation algorithm for the submodular vertex cover problem:

> **Input:** An undirected graph $G = (V, E)$ and a non-negative submodular function $f : 2^V \to \mathbb{R}_+$ on the vertex subsets.
>
> **Output:** A vertex cover $S \subseteq V$ that minimizes $f(S)$.

We remark that the classic vertex cover problem is the special case when f is the linear function $f(S) = \sum_{i \in S} w(i)$ for some non-negative vertex weights w.

A randomized 2-approximation algorithm will be given partial credits and to your help you may use the following fact without proving it.

> **Fact**. Let $V = \{1, 2, \ldots, n\}$ and let $\hat{f} : [0,1]^n \to \mathbb{R}_+$ denote the Lovász extension of f. There is a deterministic polynomial-time algorithm that minimizes $\hat{f}(x)$ subject to $x_i + x_j \geq 1$ for all $\{i, j\} \in E$ and $x_i \in [0,1]$ for all $i \in V$.

(In this problem you are asked to (i) design the algorithm, (ii) show that it runs in polynomial-time, and (iii) prove that the value of the found solution is at most twice the value of an optimal solution. You are allowed to use the above fact without any proof. For full score your algorithm should be deterministic but randomized solutions will be given partial credits. Recall that you are allowed to refer to material covered in the lecture notes.)

Solution: We use the same rounding scheme as in the first subproblem. We then have that the expected cost of our solution is

$$2 \int_0^{1/2} f(\{i : x_i \geq t\}) dt\,.$$

On the other hand,

$$\hat{f}(x) = \int_0^1 f(\{i : x_i \geq t\}) dt = \int_0^{1/2} f(\{i : x_i \geq t\}) dt + \int_{1/2}^1 f(\{i : x_i \geq t\}) dt$$

which by non-negative is at least

$$\int_0^{1/2} f(\{i : x_i \geq t\}) dt\,.$$

Our output is thus at most twice the lower bound in expectation. The algorithm can easily be derandomized by trying all relevant t's (at most $n + 1$ many and select the best one).

(This page is intentionally left blank.)