ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Final Exam, Advanced Algorithms 2017-2018

- 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 / 15 points	Problem 3 / 15 points	Problem 4 / 15 points	Problem 5 / 22 points	Problem 6 / 23 points

Total / 100

1 *(10 pts)* **Submodular functions.** Let $N = \{1, 2, 3, 4, 5, 6, 7, 8\}$ and consider the submodular function $f : 2^N \to \mathbb{R}$ defined by

$$f(S) = \min(|S|, 5) \qquad \text{for every } S \subseteq N.$$

What is the value of $\hat{f}(0, 1/4, 1/2, 1/4, 3/4, 1/2, 3/4, 1)$ where $\hat{f}$ denotes the Lovász extension of f?

(In this problem you are asked to calculate $\hat{f}(0, 1/4, 1/2, 1/4, 3/4, 1/2, 3/4, 1)$. Recall that you are allowed to refer to material covered in the lecture notes.)

Solution: Recall the definition of the Lováz extension

$$\mathbb{E}_{\sigma \in [0,1]}[f(\{i : x_i \geq \theta\})]\,,$$

where σ is distributed uniformly in the inteval $[0, 1]$. We thus have

$$\begin{aligned}
\hat{f}(0, 1/4, 1/2, 1/4, 3/4, 1/2, 3/4, 1) &= \Pr[\theta \in [0, 1/4]] \cdot f(\{2, 3, 4, 5, 6, 7, 8\}) \\
&\quad + \Pr[\theta \in (1/4, 1/2]] \cdot f(\{3, 5, 6, 7, 8\}) \\
&\quad + \Pr[\theta \in (1/2, 3/4]] \cdot f(\{5, 7, 8\}) \\
&\quad + \Pr[\theta \in (3/4, 1]] \cdot f(\{8\}) \\
&= \Pr[\theta \in [0, 1/4]] \cdot 5 \\
&\quad + \Pr[\theta \in (1/4, 1/2]] \cdot 5 \\
&\quad + \Pr[\theta \in (1/2, 3/4]] \cdot 3 \\
&\quad + \Pr[\theta \in (3/4, 1]] \cdot 1 \\
&= \frac{5 + 5 + 3 + 1}{4} = 3.5\,.
\end{aligned}$$

2 *(15 pts)* **LSH for Manhattan distance.** Recall the Manhattan distance function that we saw in class: for any d-dimensional Boolean vectors $p, q \in \{0,1\}^d$, the Manhattan distance is defined by

$$\text{dist}(p,q) = \|p-q\|_1 = |\{i : p_i \neq q_i\}|\,.$$

Design a locality sensitive hash (LSH) family $\mathcal{H}$ of functions $h : \{0,1\}^d \to \{0,1,2,3\}$ such that for any $p, q \in \{0,1\}^d$,

$$\Pr_{h\sim\mathcal{H}}[h(p) = h(q)] = \left(1 - \frac{\text{dist}(p,q)}{d}\right)^2 .$$

(In this problem you are asked to explain the hash family and show that it satisfies the above property. Recall that you are allowed to refer to material covered in the lecture notes.)

Solution: There are two solutions (the first refers to material covered in class).

Solution 1: In class we designed LSH family $\mathcal{H}_1$ of hash functions $h : \{0,1\}^d \to \{0,1\}$ so that

$$\Pr_{h\sim\mathcal{H}_1}[h(p) = h(q)] = \left(1 - \frac{\text{dist}(p,q)}{d}\right) . \tag{1}$$

Now define, similar to the construction of ANNS data structure, a new hash family $\mathcal{H}$ defined as follows:

- Select h_1 and h_2 uniformly at random from $\mathcal{H}_1$.
- Define $h : \{0,1\}^d \to \{0,1,2,3\}$ by $h(p) = x$ where x equals the number in $\{0,1,2,3\}$ whose binary representation is $h_1(p)h_2(p)$, i.e., $x = 2h_1(p) + h_2(p)$.

For any $p, q \in \{0,1\}^d$ we thus have

$$\begin{aligned}
\Pr_{h\sim\mathcal{H}}[h(p) = h(q)] &= \Pr_{h_1,h_2\in\mathcal{H}_1}[h_1(p) = h_1(q) \wedge h_2(p) = h_2(q)] \\
&= \Pr_{h_1\in\mathcal{H}_1}[h_1(p) = h_1(q)] \Pr_{h_2\in\mathcal{H}_1}[h_2(p) = h_2(q)] \qquad (h_1 \text{ and } h_2 \text{ are selected independently}) \\
&= \left(1 - \frac{\text{dist}(p,q)}{d}\right)^2 . \qquad (\text{by } (1))
\end{aligned}$$

Solution 2: Define $\mathcal{H}$ as follows:

- Select coordinates $i, j \in [d]$ independently uniformly at random.
- Define $h := h_{ij}(p) = x$ where x equals the number in $\{0,1,2,3\}$ whose binary representation is p_ip_j, i.e., $x = 2p_i + p_j$.

We have that

$$\begin{aligned}
\Pr_{h\in\mathcal{H}}[h(p)=h(q)] &= \Pr_{i,j\in[d]}[p_i=q_i \wedge p_j=q_j] \\
&= \Pr_{i\in[d]}[p_i=q_i]\cdot\Pr_{j\in[d]}[p_j=q_j] \qquad \text{(by independence of } i \text{ and } j\text{)} \\
&= \left(\Pr_{i\in[d]}[p_i=q_i]\right)^2 \\
&= \left(\frac{|\{\ell : p_\ell = q_\ell\}|}{d}\right)^2 \\
&= \left(\frac{d-|\{\ell : p_\ell \neq q_\ell\}|}{d}\right)^2 \\
&= \left(1-\frac{\mathrm{dist}(p,q)}{d}\right)^2 .
\end{aligned}$$

3 *(15 pts)* **Estimating cheese consumption of fondue lovers.** Professor Ueli von Gruyères worked hard last year to calculate the yearly cheese consumption of each individual in Switzerland. Specifically, let U be the set of all persons in Switzerland. For each person $i \in U$, Ueli calculated the amount $w_i \in \mathbb{R}_{\geq 0}$ (in grams) of the yearly cheese consumption of person i. However, to help Coop and Migros in their supply-chain management, he needs to calculate the total cheese consumption of those persons that prefer fondue over raclette. That is, if we let $F \subseteq U$ be those that prefer fondue over raclette, then Ueli wants to calculate

$$W_F = \sum_{i \in F} w_i \,.$$

The issue is that Ueli does not know the set F and he does not have the time or energy to ask the preferences of all persons. He therefore designs two estimators that only ask a single person:

Estimator $\mathcal{A}_1$: Let $W = \sum_{i \in U} w_i$. Sample person i with probability $\frac{w_i}{W}$ and output W if i prefers fondue and 0 otherwise.

Estimator $\mathcal{A}_2$: Sample person i with probability $\frac{1}{|U|}$ and output $|U| \cdot w_i$ if i prefers fondue and 0 otherwise.

Let X_1 and X_2 be the random outputs of $\mathcal{A}_1$ and $\mathcal{A}_2$, respectively. Ueli has shown that $\mathcal{A}_1$ and $\mathcal{A}_2$ are unbiased estimators and he has also bounded their variances:

$$\mathbb{E}[X_1] = \mathbb{E}[X_2] = W_F, \qquad \mathrm{Var}[X_1] \leq W^2 \qquad \text{and} \qquad \mathrm{Var}[X_2] \leq |U| \sum_{i \in U} w_i^2 \,.$$

However, Ueli is now stuck because the variances are too high to give any good guarantees for the two estimators. We are therefore going to help Ueli by designing a new estimator with good guarantees while still asking the preferences of relatively few persons.

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

$$\Pr[|Y - W_F| \geq \epsilon W] \leq 1/3 \,. \tag{2}$$

Your estimator should ask at most $3/\epsilon^2$ persons about their preferences.

(In this problem you are asked to (i) design an estimator that asks at most $3/\epsilon^2$ persons about their preferences and (ii) prove that it satisfies the guarantee (2). Recall that you are allowed to refer to material covered in the lecture notes.)

Solution: We define our estimator as follows:

- Let $t = 3/\epsilon^2$.
- Run t independent copies of $\mathcal{A}_1$ to obtain estimates $Z_1, Z_2, \ldots, Z_t$.
- Output $Y = (Z_1 + Z_2 + \ldots + Z_t)/t$.

Since $\mathcal{A}_1$ only asks one person, we have that our estimator only asks $3/\epsilon^2$ persons (as required). We now analyze the guarantee of our estimator. We have

$$\mathbb{E}[Y] = \mathbb{E}\left[(Z_1 + Z_2 + \ldots + Z_t)/t\right] = \frac{1}{t} \sum_{i=1}^{t} \mathbb{E}[Z_i] = W_F \,,$$

where the last equality follows from that $\mathcal{A}_1$ is an unbiased estimator which was given to us in the stament. As seen in the lecture notes, we also have

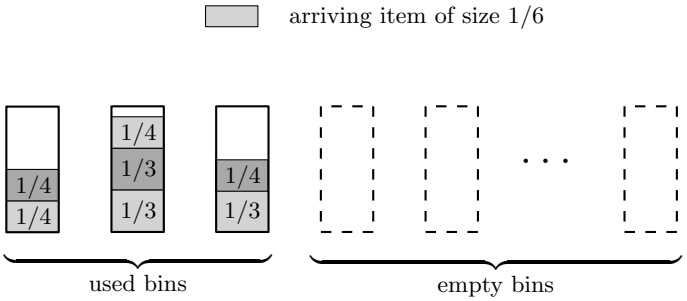
$$\mathrm{Var}[Y] = \mathrm{Var}[X_1]/t$$

and so, by the problem statement, we know $\mathrm{Var}[Y] \leq W^2/t$. We know apply Chebychev's inequality to analyze the guarantee of our estimator:

$$\Pr[|Y - W_F| \geq \epsilon W] \leq \frac{\mathrm{Var}[Y]}{\epsilon^2 W^2} \leq \frac{W^2}{t\epsilon^2 W^2} \leq 1/3\,,$$

where the last inequality is due to the selection $t = 3/\epsilon^2$.

4 *(15 pts)* **Online bin-packing.** Recall the online bin-packing problem that we saw in Exercise Set 10: We are given an unlimited number of bins, each of capacity 1. We get a sequence of items one by one each having a size of at most 1, and are required to place them into bins as we receive them. Our goal is to minimize the number of bins we use, subject to the constraint that no bin should be filled to more than its capacity. An example is as follows:



Here, seven items have already arrived that we have packed in three bins. The newly arriving item of size 1/6 can either be packed in the first bin, third bin, or in a new (previously unused) bin. It cannot be packed in the second bin since $1/3 + 1/3 + 1/4 + 1/6 > 1$. If it is packed in the first or third bin, then we still use three bins, whereas if we pack it in a new bin, then we use four bins.

In this problem you should, assuming that all items have size at most $0 < \epsilon \leq 1$, design and analyze an online algorithm for the online bin-packing problem that uses at most

$$\frac{1}{1-\epsilon}\text{OPT} + 1 \text{ bins}, \tag{3}$$

where OPT denotes the minimum number of bins an optimal packing uses. In the above example, $\epsilon = 1/3$.

(In this problem you are asked to (i) design the online algorithm and (ii) prove that it satisfies the guarantee (3)*. Recall that you are allowed to refer to material covered in the lecture notes.)*

Solution: We consider the following greedy algorithm. At the arrival of an item of size s:

- If it fits, pack the item in an already used bin.
- Otherwise, pack the item in a new bin.

To analyze the algorithm and to prove (3) the following observation is crucial:

Claim 1 *Whenever a new bin is opened, every other bin is packed with items of total size at least* $1 - \epsilon$.

Proof. We only open up a new bin if the arriving item of size $s \leq \epsilon$ does not fit in an already used bin. Since it does not fit in an already used bin, it implies that every such bin is packed with items of total size at least $1 - \epsilon$. □

Notice that the above claim implies that at any point of time, all but 1 bin have load at least $1 - \epsilon$. If our algorithm has opened $m + 1$ bins, the total size of all items packed so far is thus at

least $(1-\epsilon)m$. However, OPT clearly needs $(1-\epsilon)m$ bins to pack these items (since each bin can take items of size at most 1). Hence, $m \leq \frac{\mathrm{OPT}}{1-\epsilon}$ and

$$m + 1 \leq \frac{\mathrm{OPT}}{1-\epsilon} + 1\,,$$

as required.

5 *(consisting of subproblems **a**-**b**, 22 pts)* **Finding large cuts.**

5a *(9 pts)* Consider the following algorithm that takes as input an undirected graph $G = (V, E)$:

> SimpleCut($G = (V, E)$):
>
> 1. Let $\mathcal{H}$ be a 2-universal family of hash functions $h : V \to \{0, 1\}$.
> 2. Select $h \in \mathcal{H}$ at random.
> 3. **return** the vertex set $S = \{v \in V : h(v) = 0\}$.

Prove the following:

In expectation, the set S returned by SimpleCut cuts at least $|E|/2$ edges.

(In this problem you are asked to prove the above statement. Recall that you are allowed to refer to material covered in the lecture notes.)

Solution: Let X_e be the indicator random variable that edge e is cut. Then

$$\mathbb{E}[\#\text{ edges cut}] = \mathbb{E}\left[\sum_{e \in E} X_e\right] = \sum_{e \in E} \mathbb{E}[X_e]\,.$$

We complete the proof by showing that $\mathbb{E}[X_e] \geq 1/2$ for $e = \{u, v\} \in E$. We have

$$\mathbb{E}[X_e] = \Pr[e \text{ is cut}] = \Pr[h(u) \neq h(v)] = 1 - \Pr[h(u) = h(v)] \geq 1/2\,,$$

where the last inequality follows because h was selected at random from a 2-universal hash family $\mathcal{H}$, and thus $\Pr[h(u) = h(v)] \leq 1/2$ for $u \neq v$.

5b *(13 pts)* Design and analyze a polynomial-time algorithm for the following problem:

> **Input:** a vertex set V.
>
> **Output:** vertex subsets $S_1, S_2, \ldots, S_\ell \subseteq V$ with the following property:
>
> For every set of edges $E \subseteq \binom{V}{2}$, there is an $i \in \{1, 2, \ldots, \ell\}$ such that
>
> $$|\{e \in E : |e \cap S_i| = 1\}| \geq |E|/2\,,$$
>
> i.e., S_i cuts at least half the edges in $G = (V, E)$.

We remark that, since your algorithm should run in time polynomial in $n = |V|$, it can output at most polynomially (in n) many vertex sets. We also emphasize that the algorithm does **not** take the edge set E as input.

(In this problem you are asked to (i) design the algorithm, (ii) show that it runs in time polynomial in n, and (iii) prove that the output satisfies the property given in the problem statement. Recall that you are allowed to refer to material covered in the lecture notes.)

Solution: The algorithm is as follows:

1. Construct a 2-universal hash family $\mathcal{H}$ of hash functions $h : V \to \{0, 1\}$.
   As seen in class, we can construct such a family in time polynomial in n, and $\mathcal{H}$ will contain $O(n^2)$ hash functions. Specifically, $\mathcal{H}$ has a hash function for each $a, b \in [p]$ with $a \neq 0$ where p is a prime satisfying $|U| \leq p \leq 2|U|$.
2. Let $h_1, h_2, \ldots, h_\ell$ be the hash functions of $\mathcal{H}$ where $\ell = |\mathcal{H}|$.
3. For each $i = 1, \ldots, \ell$, output the set $S_i = \{v : h_i(v) = 0\}$.

As already noted, the algorithm runs in time polynomial in n and it outputs polynomially many vertex subsets. We proceed to verify the property given in the statement. Consider an arbitrary edge set $E \subseteq \binom{V}{2}$. By the previous subproblem,

$$\begin{aligned}
\frac{|E|}{2} &\leq \mathbb{E}_{h \in \mathcal{H}} \left[\#\text{ edges cut by } \{v : h(v) = 0\}\right] \\
&= \frac{1}{\ell} \sum_{i=1}^{\ell} (\#\text{ edges cut by } \{v : h_i(v) = 0\}) \\
&= \frac{1}{\ell} \sum_{i=1}^{\ell} (\#\text{ edges cut by } S_i)\,.
\end{aligned}$$

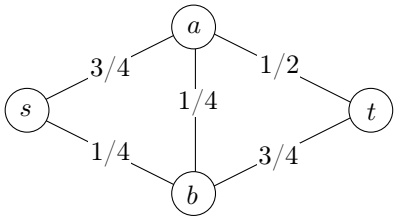
We thus have that the average number of edges that the sets $S_1, S_2, \ldots, S_\ell$ cut is at least $|E|/2$. This implies that at least one of those sets cuts at least $|E|/2$ edges.

6 *(consisting of subproblems **a-b**, 23 pts)* **Finding a min s,t-cut using linear programming.** Consider an undirected graph $G = (V, E)$ and let $s \neq t \in V$. In the minimum (unweighted) s,t-cut problem, we wish to find a set $S \subseteq V$ such that $s \in S$, $t \notin S$ and the number of edges crossing the cut is minimized.

We shall use a linear program to solve this problem. Let P be the set of all paths between s and t in the graph G. The linear program has a variable y_e for each edge $e \in E$ and is defined as follows:

$$\begin{aligned} \text{minimize} \quad & \sum_{e \in E} y_e \\ \text{subject to} \quad & \sum_{e \in p} y_e \geq 1 \quad \forall p \in P, \\ & y_e \geq 0 \qquad \forall e \in E. \end{aligned}$$

For example, consider the following graph where the numbers on the edges depict the y_e-values of a feasible solution to the linear program:



The values on the edges depict a feasible but not optimal solution to the linear program. That it is feasible follows because each y_e is non-negative and $\sum_{e \in p} y_e \geq 1$ for all $p \in P$. Indeed, for the path s, b, a, t we have $y_{\{s,b\}} + y_{\{b,a\}} + y_{\{a,t\}} = 1/4 + 1/4 + 1/2 = 1$, and similar calculations for each path p between s and t show that $\sum_{e \in p} y_e \geq 1$. That the solution is not optimal follows because its value is 2.5 whereas an optimal solution has value 2.

6a *(10 pts)* Let OPT denote the number of edges crossing a minimum s,t-cut and let OPT_{LP} denote the value of an optimal solution the linear program. Prove that $\text{OPT}_{\text{LP}} \leq \text{OPT}$.

(In this problem you are asked to prove $\text{OPT}_{\text{LP}} \leq \text{OPT}$. Recall that you are allowed to refer to material covered in the lecture notes.)

Solution:

Let S be a minimum s,t-cut; then the number of edges cut by S is OPT. We shall exhibit a feasible solution y to the linear program such that value of y is OPT. This then implies that $\text{OPT}_{\text{LP}} \leq \text{OPT}$ as the minimum value of a solution to the linear program is at most the value of y. Define y as follows: for each $e \in E$

$$y_e = \begin{cases} 1 & \text{if } e \text{ is cut by } S, \\ 0 & \text{otherwise.} \end{cases}$$

Notice that, by this definition, $\sum_{e \in E} y_e = \text{OPT}$. We proceed to show that y is a feasible solution:

- for each $e \in E$, we have $y_e \geq 0$;
- for each $p \in P$, we have $\sum_{e \in p} y_e \geq 1$ since any path from s to t must exit the set S. Indeed, S contains s but it does not contain t, and these edges (that have one end point in S and one end point outside of S) have y-value equal to 1.

6b *(13 pts)* Prove that $\text{OPT} \leq \text{OPT}_{\text{LP}}$, where OPT and OPT_{LP} are defined as in **6a**.

Hint: Round a feasible linear programming solution y. In the (randomized) rounding it may be helpful to consider, for each vertex $v \in V$, the length of the shortest path from s to v in the graph where edge $e \in E$ has length y_e. For example, in the graph and linear programming solution depicted in the problem statement, we have that the length of the shortest path from s to a equals $1/2$.

(In this problem you are asked to prove $\text{OPT} \leq \text{OPT}_{\text{LP}}$*. Recall that you are allowed to refer to material covered in the lecture notes.)*

Solution: Let y be an optimal solution to the linear program of value OPT_{LP}. We shall give a randomized rounding that outputs an s,t-cut that in expectation cuts OPT_{LP} edges. Since any cut must cut at least OPT edges, it then follows that $\text{OPT}_{\text{LP}} \geq \text{OPT}$.

We now describe the rounding procedure. For each vertex $v \in V$, define

$$x_v = \text{the length of the shortest path from } s \text{ to } v \text{ in the graph where edge } e \in E \text{ has length } y_e.$$

Notice that $x_s = 0$ and $x_t \geq 1$. That $x_t \geq 1$ is due to the fact that for every path $p \in P$ from s to t we have $\sum_{e \in p} y_e \geq 1$. Furthermore, the variables x satisfy the following key property:

Claim 2 *For every edge $\{u,v\} \in E$, we have $y_{\{u,v\}} \geq |x_v - x_u|$.*

Proof. Name the vertices so that $x_v > x_u$ and suppose toward contradiction that $y_{\{u,v\}} < x_v - x_u$. Let p_u be a shortest path from s to u that has length x_u. And consider the path $p_u + \{u,v\}$. This is a path from s to v that has length $x_u + y_{\{u,v\}}$, which is strictly smaller than x_v (contradicting the definition of x_v to be the length of a *shortest* path). □

We can thus deduce that we have a feasible solution to the linear program:

$$\begin{aligned}
\text{minimize} \quad & \sum_{e \in E} y_e \\
\text{subject to} \quad & \sum_{e \in p} y_{\{u,v\}} \geq x_u - x_v, \quad \text{for every } \{u,v\} \in E \\
& \sum_{e \in p} y_{\{u,v\}} \geq x_v - x_u, \quad \text{for every } \{u,v\} \in E \\
& x_s = 0, x_t \geq 1, \\
& x_v \geq 0, \forall v \in V \\
& y_e \geq 0 \forall e \in E.
\end{aligned}$$

In class, we saw that the above linear program has cost equal to OPT which finishes the proof. For completeness, we give the full argument. The rounding algorithm is as follows:

- Select $\theta \in [0,1]$ uniformly at random.
- Output $S = \{v \in V : x_v < \theta\}$.

Notice that S always contains s (since $x_s = 0$) and never contains t (since $x_t \geq 1$). We now analyze the expected number of edges that are cut. Let X_e be the random indicator variable for the event that edge e is cut. Then

$$\mathbb{E}_\theta[\#\text{ edges cut}] = \mathbb{E}_\theta\left[\sum_{e \in E} X_e\right] = \sum_{e \in E} \mathbb{E}_\theta[X_e] = \sum_{e \in E} \Pr_\theta[\text{e is cut}].$$

We now analyze, for a fixed edge $e = \{u, v\} \in E$, the probability that e is cut over the uniformly at random choice $\theta \in [0, 1]$. Assume without loss of generality that $x_u < x_v$. We then have that e is cut if and only if $\theta \in [x_u, x_v]$ which happens with probability at most $x_v - x_u$ (we say at most because x_v could, in theory, be bigger than 1).

Now, by Claim 2 above, $x_v - x_u \leq y_e$. So we have

$$\mathrm{OPT} \leq \mathbb{E}_\theta[\#\text{ edges cut}] \leq \sum_{e \in E} y_e = \mathrm{OPT}_{\mathrm{LP}} ,$$

as required.