ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Homework I, Advanced Algorithms 2019

Due on Friday March 29 at 17:00 (upload one solution per group on moodle). Solutions to many homework problems, including problems on this set, are available on the Internet, either in exactly the same formulation or with some minor perturbation. It is *not acceptable* to copy such solutions. It is hard to make strict rules on what information from the Internet you may use and hence whenever in doubt contact Ola Svensson. You are, however, allowed to discuss problems in groups of up to three students; it is sufficient to hand in one solution per group.

1 *(20 pts)* Let $G = (V, E)$ be an undirected graph where every vertex $v \in V$ has at most $\Delta \in \mathbb{N}$ neighbors. We will analyze the following randomized algorithm that assigns a "color" $c(v) \in \{1, 2, \ldots, \Delta\}$ to every vertex:

For each vertex v, select $c(v) \in \{1, 2, \ldots, \Delta\}$ uniformly at random and independently of other vertices. In other words, v receives a color $i \in \{1, 2, \ldots, \Delta\}$ with probability $1/\Delta$.

1a *(5 pts)* For a coloring $c : V \to \{1, 2, \ldots, \Delta\}$, we say that an edge $\{u, v\} \in E$ is *monochromatic* if $c(u) = c(v)$. Show that the expected number of monochromatic edges in the coloring c output by our randomized algorithm equals $\frac{|E|}{\Delta}$.

Solution: For any edge $e \in E$ let X_e be a random variable such that $X_e = 1$ if edge e is monochromatic and $X_e = 0$ otherwise. We define $X = \sum_{e \in E} X_e$. Note that X is the number of monochromatic edges. Thus we have

$$
\begin{aligned}
\mathbb{E}[X] &= \sum_{e \in E} X_e \\
&= \sum_{e \in E} \mathbb{E}[X_e] && \text{By linearity of expectation} \\
&= \sum_{e \in E} 1 \cdot \mathbb{P}[X_e = 1] \\
&= \sum_{\{u,v\} \in E} \mathbb{P}[c(u) = c(v)] && \text{By definition of } X_e \\
&= \sum_{\{u,v\} \in E} \sum_{i=1}^{\Delta} \mathbb{P}[c(u) = i] \cdot \mathbb{P}[c(u) = i] && \text{Since } c(u) \text{ and } c(v) \text{ are independent} \\
&= \sum_{\{u,v\} \in E} \sum_{i=1}^{\Delta} \frac{1}{\Delta} \cdot \frac{1}{\Delta} && \text{Since } c(u) \text{ and } c(v) \text{ are picked uniformly at random} \\
&= |E| \cdot \Delta \cdot \frac{1}{\Delta^2} \\
&= \frac{|E|}{\Delta}.
\end{aligned}
$$

1b *(15 pts)* For a vertex v, define $N(v) = \{u \in V : \{u, v\} \in E\}$ to be the set of neighbors of v. For a coloring $c : V \to \{1, 2, \ldots, \Delta\}$, we say that a vertex v is *good* if $c(v) \neq c(u)$ for all neighbors $u \in N(V)$. Show that the expected number of good vertices in the coloring c output by our randomized algorithm is at least $\left(1 - \frac{1}{\Delta}\right)^{\Delta} |V|$.

Solution: Let Y_v be a random variable such that $Y_v = 1$ if v is good and $Y_v = 0$ otherwise. Let $Y = \sum_{v \in V} Y_v$, hence, Y is the number of good vertices. Therefore we have

$$
\begin{aligned}
\mathbb{E}[Y] &= \sum_{v \in V} Y_v \\
&= \sum_{v \in V} \mathbb{E}[Y_v] && \text{By linearity of expectation} \\
&= \sum_{v \in V} 1 \cdot \mathbb{P}[Y_v = 1] \\
&= \sum_{v \in V} \mathbb{P}[\forall u \in N(v), c(u) \neq c(v)] && \text{By definition of } Y_v \\
&= \sum_{v \in V} \prod_{u \in N(v)} \mathbb{P}[c(u) \neq c(v)] && \text{Since colors are picked independently} \\
&= \sum_{v \in V} \prod_{u \in N(v)} \left(1 - \frac{1}{\Delta}\right) && \text{There are } \Delta - 1 \text{ options for color of } c(u) \\
&= \sum_{v \in V} \left(1 - \frac{1}{\Delta}\right)^{|N(v)|} \\
&\geq \sum_{v \in V} \left(1 - \frac{1}{\Delta}\right)^{\Delta} && \text{Since } |N(v)| \leq \Delta \\
&= |V| \cdot \left(1 - \frac{1}{\Delta}\right)^{\Delta}.
\end{aligned}
$$

2 *(20 pts)* Consider a general (not necessarily bipartite) graph $G = (V, E)$. Let $(V, \mathcal{I})$ be the matroid with ground set V and

$$\mathcal{I} = \{U \subseteq V : \ G \text{ has a matching in which every vertex of } U \text{ is matched}\}.$$

Recall that we say that a vertex v is matched by a matching M if there is an edge in M incident to v. Show that $(V, \mathcal{I})$ is indeed a matroid by verifying the two axioms.

Solution: We will verify that $\mathcal{I}$ satisfies both axioms.

First axiom If $U_1 \in \mathcal{I}$ and $U_2 \subseteq U_1$ then there is a matching of G where every vertex of U_1 is matched. In particular every vertex of U_2 is matched by it, so $U_2 \in \mathcal{I}$

Second Axiom We have $U_1, U_2 \in \mathcal{I}$ and $|U_1| > |U_2|$ and we want to show that there exists some vertex $v \in U_1$ such that $U_2 \cup \{v\} \in \mathcal{I}$. Let M_1 and M_2 the be matchings of U_1 and U_2 respectively. If there is a vertex in $U_1 \setminus U_2$ that is matched by M_2 then we can just add this

vertex to U_2 without changing M_2 and we are done. So assume that there is no such vertex. Take the symmetric difference $M_1 \Delta M_2$. In $M_1 \Delta M_2$ every vertex has degree at most 2 so it consists of paths and cycles. Note that the paths are alternating with respect to M_1 and M_2.

Take a vertex v in $U_1 \setminus U_2$. Since there is no edge touching v in M_2, it has degree 1 in $M_1 \Delta M_2$ so it is the beginning of a path P. Suppose that it ends at some vertex u. If $u \in V \setminus U_2$ then $M_2 \Delta P$ is a matching where every vertex of U_2 is still matched and v is matched, so $U_2 \cup \{v\} \in \mathcal{I}$ as we want.

If $u \in U_2$, then $M_2 \Delta P$ does not match u, so it would not work. However, u cannot end in $U_2 \cap U_1$ because every vertex there has degree 0 or 2 (in $M_1 \Delta M_2$). Also u can be in $U_2 \setminus U_1$, but it cannot be so for all vertices v in $U_1 \setminus U_2$, because $|U_1 \setminus U_2| > |U_2 \setminus U_1|$. Thus there must be a vertex v in $U_1 \setminus U_2$ such that its corresponding path in $M_1 \Delta M_2$ ends outside U_2.

3 *(20 pts)* **Primal-dual algorithm for the weighted 3-Uniform Vertex Cover problem.** A k-uniform hypergraph is defined by a tuple (V, E) where V is the set of vertices and every hyperedge $e \in E$ is a subset of V of cardinality k. We remark that a 2-uniform hypergraph is simply a graph. Here we will consider the weighted vertex cover problem on 3-uniform hypergraphs:

Input: A 3-uniform hypergraph $G = (V, E)$ with vertex weights $w : V \to \mathbb{R}_+$.

Output: A vertex cover $C \subseteq V$ minimizing the weight $w(C) = \sum_{v \in C} w(v)$ subject to that every edge $e \in E$ is covered, i.e., $e \cap C \neq \emptyset$.

The LP relaxation and its dual is similar to that of the vertex cover problem on graphs:

(Primal) LP Relaxation

$$\begin{aligned} \textbf{minimize} \quad & \sum_{v \in V} w(v) x_v \\ \textbf{subject to} \quad & x_u + x_v + x_w \geq 1 \quad \text{for } \{u, v, w\} \in E \\ & x_v \geq 0 \quad \text{for } v \in V \end{aligned}$$

(Dual)

$$\begin{aligned} \textbf{maximize} \quad & \sum_{e \in E} y_e \\ \textbf{subject to} \quad & \sum_{e \in E : v \in e} y_e \leq w(v) \text{ for } v \in V \\ & y_e \geq 0 \qquad \text{for } e \in E \end{aligned}$$

The Vertex Cover problem (also on 3-uniform hypergraphs) is NP-hard and therefore we do not expect an efficient (polynomial-time) algorithm that finds exact solutions. In this problem, we will analyze and implement a simple and very fast primal-dual algorithm that achieves the best-known guarantees.

The primal-dual algorithm works as follows. It will maintain a feasible dual solution y that initially is set to $y_e = 0$ for every $e \in E$. It will then iteratively improve the dual solution until the set $C = \{v \in V : \sum_{e \in E : v \in e} y_e = w(v)\}$ forms a vertex cover. Note that C consists of those vertices whose constraints in the dual are tight. The formal description of the algorithm is as follows:

1. Initialize the dual solution y to be $y_e = 0$ for every $e \in E$.
2. While $C = \{v \in V : \sum_{e \in E : v \in e} y_e = w(v)\}$ is not a vertex cover:
   - Select an edge $\{u, v, w\} \in E$ that is not covered by C, i.e., $C \cap \{u, v, w\} = \emptyset$.
   - Increase $y_{\{u,v,w\}}$ until one of the dual constraints (corresponding to u, v or w) becomes tight.
3. Return $C = \{v \in V : \sum_{e \in E : v \in e} y_e = w(v)\}$.

Show that the primal-dual algorithm has an approximation guarantee of 3. That is, show that the returned vertex cover C has weight $\sum_{v \in C} w(v)$ at most three times the weight of an optimal solution.

Solution: Note that, by construction, the output C is a vertex cover and the computed dual solution is feasible. Let y be the dual solution when the primal-dual algorithm terminates. We have the following:

$$\begin{aligned}
\sum_{v \in C} w(v) &= \sum_{v \in C} \sum_{u:\{u,v,w\} \in E} y_{\{u,v,w\}} && \text{(by the definition of set } C\text{)} \\
&\leq \sum_{v \in V} \sum_{u:\{u,v,w\} \in E} y_{\{u,v,w\}} && \text{(because } y_e\text{'s are non-negative and } C \subseteq V\text{)} \\
&= 3 \cdot \sum_{e \in E} y_e && \text{(each edge is counted three times, once per each end-point)} \\
&\leq 3 \cdot LP_{OPT} && \text{(by the weak-duality theorem)} \\
&\leq 3 \cdot OPT. && \text{(because the LP is a relaxation)}
\end{aligned}$$

4 *(20 pts)* **Interval packing.** Given a set of unit intervals $[a_i, a_i + 1]$ for each $1 \leq i \leq n$ and a weight function on intervals, the *maximum weight unit interval packing* problem asks for a subset J of intervals of maximum weight such that no intervals in J overlap.

Formulate a linear program for the maximum weight unit interval packing problem and show that any extreme point of your linear program is integral.

Solution:

Suppose that weights are denoted by w_i for all $i \in [n]$. Then, we can write the following linear programming:

$$\textbf{maximize} \quad \sum_{i \in [n]} w_i \cdot x_i$$

$$\textbf{subject to} \quad \sum_{i:\ a_i \in [a_j, a_j+1]} x_i \leq 1 \quad \forall j \in [n]$$

$$x_j \in [0,1] \quad \forall j \in [n]$$

Now, we prove that the extreme points of this linear programming are integral. Without loss of generality, assume that $a_1 \leq a_2 \leq \cdots \leq a_n$. We index each constraint $\sum_{i:\ a_i \in [a_j, a_j+1]} x_i \leq 1$ by j. Let $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ be a feasible solution which is not integral. From now on let $A = \{i \in [n] : x_i \in (0,1)\}$. Now, let $k_1 \in A$ be the smallest number such that k_1'th constraint is tight, i.e., $\sum_{i:\ a_i \in [a_{k_1}, a_{k_1}+1]} x_i = 1$. Now, we need the following critical fact:

Fact: If $k \in A$ is such that $\sum_{i: a_i \in [a_k, a_k+1]} x_i = 1$ then $\exists k' \in A$ such that $k' > k$ and $a_{k'} \in [a_k, a_k + 1]$. Informally, this fact says that if you have a fractional x_k in a condition which is tight, you should have at least another fractional $x_{k'}$ in that constraint too.

Color k_1'th interval in red (suppose that all the intervals are black initially). Now look at k_1'th constraint, and find the furthest to the right interval which is included in this constraint and color it in red. Now find the smallest $k_2 \in A$ such that $k_2 > k_1$ and k_2'th constraint is tight. Now look at the intervals included in this constraint. If there are two red intervals in the constraint, do nothing and proceed to the next k_3 and repeat. If there is just one red interval in the constraint, color the furthest interval to the right which is in this constraint in red (note that this interval cannot be red before this step, since all intervals have length 1 and this is a new constraint which has not been examined yet) and then proceed to the next k_3 and repeat. If there is no red intervals in the constraint, stop.

After the above mentioned process stops, do the following. Let $R = \{r_1, r_2, \ldots\}$ be the set of indices corresponding to red intervals and for each $j > i$, interval corresponding to r_j is to the right of interval corresponding to r_i. For some small enough $\epsilon > 0$, for $j \in \{r_1, r_3, r_5, \ldots\}$ do the following

$$y_j = x_j - \epsilon$$
$$z_j = x_j + \epsilon$$

and for $j \in \{r_2, r_4, r_6, \ldots\}$ do the following

$$y_j = x_j + \epsilon$$
$$z_j = x_j - \epsilon$$

and for $j \in [n] \setminus \{r_1, r_2, \dots\}$

$$y_j = x_j$$
$$z_j = x_j$$

Note that since any tight constraint had exactly two or zero red intervals, both $\mathbf{y} = (y_1, y_2, \dots)$ and $\mathbf{z} = (z_1, z_2, \dots)$ are feasible, and $\mathbf{x} = \frac{\mathbf{y}+\mathbf{z}}{2}$, so it cannot be an extreme point. So, all extreme points must be integral. (For an example see Figure 1)

5 *(20 pts)* **Implementation.** The objective of this problem is to successfully solve the problem *Hiking Trails* on our online judge. You will find detailed instructions on how to do this on Moodle.

Solution: Note that this is the weighted vertex cover problem on 3-uniform hypergraphs (huts are vertices, trails are edges). One algorithm that can be used is given in Problem 3 (the budgets correspond to the dual values).

Listing 1: Implementation of the primal-dual algorithm for weighted vertex cover in Python 3

```
def get_ints(): # read a line of integers as an array
    return [int(x) for x in input().split(' ')]

# read input
n, m = get_ints()
c = get_ints()
trails = [get_ints() for i in range(m)]
budgets = [0] * m

# c[i] will serve as slack of the dual constraint corresponding to i-th
# hut/vertex (at first it is the cost, since budgets are zero)

# primal-dual algorithm
for i, (u, v, w) in enumerate(trails):                  # for every edge
    if c[u - 1] > 0 and c[v - 1] > 0 and c[w - 1] > 0:  # that is not covered
        # increase budget as much as possible
        budgets[i] = min(c[u - 1], c[v - 1], c[w - 1])
        # update slack of the constraints
        c[v - 1] -= budgets[i]
        c[u - 1] -= budgets[i]
        c[w - 1] -= budgets[i]

# print output; we take those huts whose slacks have been brought down
# to 0 (tight vertices)
print(c.count(0))
for i in range(n):
    if c[i] == 0:
        print(i+1, end=' ')
print()
print(*budgets)
```

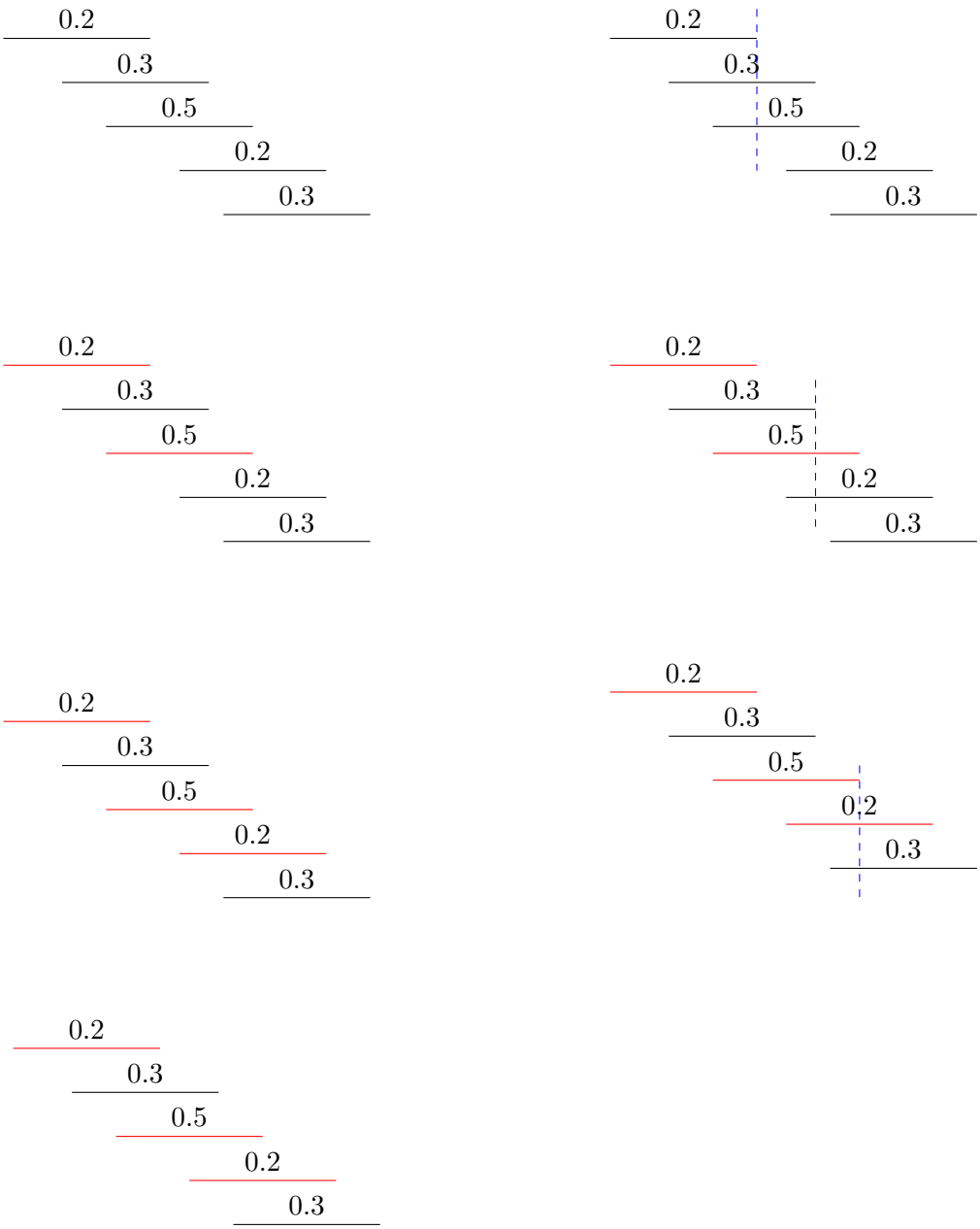

Figure 1: Example of proof of Question 4