Advanced Algorithms
Homework 1

Daniel Grosu, Marijn van der Meer, Frederic Berdoz

April 3, 2020

Exercise 1a. Let $G = (V, E)$ be an undirected graph on n vertices and $nd/2$ edges. Consider the following probabilistic experiment for finding an independent set in G. Delete each vertex in G independently with probability $1 - 1/d$. Compute the expected number of vertices and edges that remain after the deletion process.
Answer. Given $G = (V, E)$ where $|V| = n$ and $|E| = \frac{nd}{2}$, we first find the expected number of vertices and then the number of edges left after the deletion process.

- **Expected number of vertices:** If vertices are deleted with probability $1 - \frac{1}{d}$, then vertices are kept with probability $\frac{1}{d}$. We set X_{v_i}, the indicator variable of whether a vertex v_i was kept after the deletion process. So, $X_{v_i} = 1$ if v_i was not deleted and 0 otherwise. Thus, by setting C_v, the number of vertices left after deletion, we find its expected value as :

$$E[C_v] = E[\sum_{i=1}^{n} X_{v_i}] = \sum_{i=1}^{n} E[X_{v_i}] = \sum_{i=1}^{n} P[X_{v_i} = 1] = \sum_{i=1}^{n} \frac{1}{d} = \frac{n}{d} \tag{1}$$

  Where the second equality follows by linearity of expectation and the third by property of indicator variables.

- **Expected number of edges:** We set C_e, the number of edges left after deletion. Again, we set an indicator variable X_{e_i} that is 1 if an edge was kept and 0 otherwise. To calculate its probability of being 1, we need to take a closer look at what happens to edges:
  An edge is deleted if either one or both of its adjacent vertices are deleted. So, the probability of an edge to still be there after the process is equal to the probability of both its adjacent vertices being kept. For an edge adjacent to vertices v_a and v_b, that probability is :

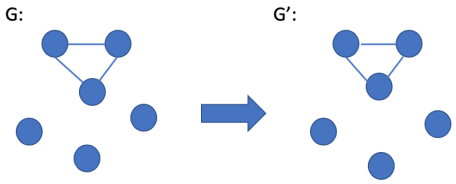
$$P(X_{e_1} = 1) = P(X_{v_a} = 1, X_{v_b} = 1) = P(X_{v_a} = 1)P(X_{v_b} = 1) = \frac{1}{d} * \frac{1}{d} = \frac{1}{d^2} \tag{2}$$

  The second equality comes from the fact that vertices are deleted independently from each-other. From this, it follows that the expected number of edges is :

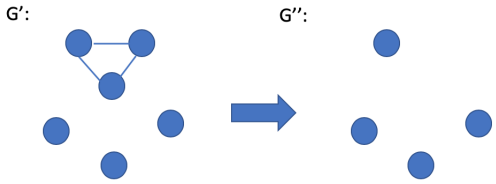
$$E[C_e] = E[\sum_{i=1}^{nd/2} X_{e_i}] = \sum_{i=1}^{nd/2} E[X_{e_i}] = \sum_{i=1}^{nd/2} P[X_{v_i} = 1] = \sum_{i=1}^{nd/2} \frac{1}{d^2} = \frac{n}{2d} \tag{3}$$

Again, we used linearity of expectation and the property of an indicator variable here.

Exercise 1b. From this infer that there is an independent set with at least $\frac{n}{2d}$ vertices in any graph on n vertices and $\frac{nd}{2}$ edges.
Answer. We first check whether the fact of removing vertices with probability $1 - \frac{1}{d}$ is enough to find vertices that form an independent set in G. We show that it is not the case with a counter-example where $n = 6$ and an average degree of $d = 1$:



In this very special case with $d = 1$, the graph G' after the random deletion process where we deleted vertices with probability 0 is identical to G. As G was not an independent set, neither is G'.
So, we need to do something more to G' to assure we find an independent set in G. The easiest way to do this is to follow the algorithm proposed in 1c: for each remaining edge in G', we remove the edge and one of its incident vertices.



We state that the remaining vertices in G'' form an independent set in G. We prove this by assuming the contrary (e.g. the vertices of G'' are not an independent set in G). This means that at least two vertices v_a and v_b are linked by an edge in G. But if v_a and v_b are adjacent in G and still both present in G'', this means that the edge between them was somehow deleted during the first random phase while keeping both vertices, otherwise the second phase would have removed one of them when it deleted their linking edge. This is a contradiction to how G' is built, because edges only disappear if one of its incident vertices is deleted $\perp$
Thus, we can calculate the expected number of vertices remaining in G'' (e.g. the expected size of the constructed independent set). So, by setting $\alpha(G)$ as the size of the independent set we constructed by this two-step process, we find:

$$E[\alpha(G)] \geqslant E[C_v - C_e] = E[C_v] - E[C_e] = \frac{n}{d} - \frac{n}{2d} = \frac{n}{2d} \tag{4}$$

It is a lower bound on the number of independent vertices, because to create G'', we removed a vertex for every edge so the expected number of removed vertices from G' to G'' is at most $\frac{n}{2d}$. In the case where some vertices in G' can have a bigger degree than 1, we see by looking at the example below, that removing the centre vertex immediately leaves the rest as an independent set with more vertices than if we had removed the outer vertices (e.g.

corresponding to the number of edges). So it is possible to obtain an independent set bigger than $C_v - C_e$.

Exercise 1c. Let G be a 3-regular graph. Suppose we would like to turn our randomised experiment into an algorithm as follows. We delete every vertex of G independently with probability 2/3. Then for every edge that remain delete one of its endpoints. Derive an upper bound on the probability that this algorithm finds an independent set smaller than $(1-\epsilon)n/6$.
Answer: Firstly, for the sake of clarity, let's give and recall the definition of some random variables that will be used in the derivation of this upper bound.

- $\alpha(G) = \sum_{i=1}^{n} X_{v_i}$: Size of the independent vertex set returned by the two steps algorithm on the input $G = (V, E)$ with $|V| = n$ and $|E| = \frac{nd}{2}$ (d=3 in our case).
- X_{v_i}: Indicator variable indicating if vertex $v_i \in V, i = 1, ..., n$ is in the independent vertex set returned by algorithm ($X_{v_i} = 1$) or if it has been deleted in the process ($X_{v_i} = 0$).
- $\Delta := n - \alpha(G)$: Number of vertices that are deleted by the algorithm.

In addition to that, recall the following identities and inequalities:

- **Markov's Inequality**: For any positive random variable X and $\epsilon > 0$:

$$P[X > \epsilon] \leqslant \frac{\mathbb{E}[X]}{\epsilon}. \tag{5}$$

- **Chebyshev's Inequality**: For any random variable X and $\epsilon > 0$:

$$P[|X - \mathbb{E}[X]| > \epsilon] \leqslant \frac{\mathrm{Var}[X]}{\epsilon^2}. \tag{6}$$

- For any random variable X:

$$\mathrm{Var}[aX + b] = a^2 \mathrm{Var}[X], \quad \forall a, b \in \mathbb{R}. \tag{7}$$

- For any set of random variable $X_i, i = 1, ..., n$:

$$\mathrm{Var}\left[\sum_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} \mathrm{Var}[X_i] + \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} \mathrm{Cov}[X_i, X_j]. \tag{8}$$

- For two indicator random variable X_v and X_u:

$$\mathrm{Cov}[X_u, X_v] = P[X_u = 1 \cap X_v = 1] - P[X_u = 1]P[X_v = 1]. \tag{9}$$

- Expectancy of the number of deleted vertices : from part 1b., we have that $\mathbb{E}[\alpha(G)] \geqslant \frac{n}{2d}$. So,

$$\mathbb{E}[\Delta] = n - \mathbb{E}[\alpha(G)] \leqslant n - \frac{1}{6}n = \frac{5}{6}n. \tag{10}$$

With this in mind, the goal of this exercise is to find an upper bound on $P[\alpha(G) < (1-\epsilon)\frac{n}{6}]$. We can rewrite this probability as follows:

$$\begin{aligned}
P\left[\alpha(G) < (1-\epsilon)\frac{n}{6}\right] &= P\left[\alpha(G) - n < (1-\epsilon)\frac{n}{6} - n\right] \\
&= P\left[-\Delta < -\frac{5+\epsilon}{6}n\right] \\
&= P\left[\Delta > \frac{5+\epsilon}{6}n\right] \\
&= P\left[\Delta - \mathbb{E}[\Delta] > \frac{5+\epsilon}{6}n - \mathbb{E}[\Delta]\right] \\
&\overset{(10)}{\leqslant} P\left[\Delta - \mathbb{E}[\Delta] > \frac{5+\epsilon}{6}n - \frac{5}{6}n\right] \\
&= P\left[\Delta - \mathbb{E}[\Delta] > \frac{\epsilon n}{6}\right] \\
&\leqslant P\left[|\Delta - \mathbb{E}[\Delta]| > \frac{\epsilon n}{6}\right] \\
&\overset{(6)}{\leqslant} \frac{36}{\epsilon^2 n^2}\mathrm{Var}[\Delta].
\end{aligned}$$

What we need now is an upper bound on the variance of Δ. We have the following:

$$\begin{aligned}
\mathrm{Var}[\Delta] &= \mathrm{Var}[n - \alpha(G)] \\
&\overset{(7)}{=} \mathrm{Var}[\alpha(G)] \\
&= \mathrm{Var}\left[\sum_{i=1}^{n} X_{v_i}\right] \\
&\overset{(8)}{=} \sum_{i=1}^{n} \mathrm{Var}[X_{v_i}] + \sum_{i=1}^{n}\sum_{\substack{j=1 \\ j\neq i}}^{n} \mathrm{Cov}[X_{v_i}, X_{v_j}]
\end{aligned}$$

In order to continue, we need the distribution of the the random variables X_{v_i}. Since this is an indicator variable, it is a Bernoulli distribution and it has only one parameter $0 \leqslant p_i := P[X_{v_i} = 1] \leqslant 1$. We have that

- $\mathrm{Var}[X_{v_i}] = \mathbb{E}[X_{v_i}^2] - \mathbb{E}[X_{v_i}]^2 = p_i - p_i^2 \leqslant 1, \quad \forall i = 1, ..., n.$
- $\mathrm{Cov}[X_{v_i}, X_{v_j}] \overset{(9)}{=} P[X_{v_i} = 1 \cap X_{v_j} = 1] - p_i p_j = P[X_{v_i} = 1 | X_{v_j} = 1]p_j - p_i p_j$. To go further, we make the following observation. In expectation, for any graph $G = (V, E)$ and any vertex $v_i \in V$, knowing that another random vertex $v_j \in V$ has survived the algorithm decreases the probability that v_i also survives, i.e. in average, $P[X_{v_i} =$

$1|X_{v_j} = 1] \leqslant p_i$. This yields that the average of the covariance over the pairs of vertices $\{v_i, v_j\}$ is negative, and thus:

$$\sum_{i=1}^{n} \text{Var}[X_{v_i}] + \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} \text{Cov}[X_{v_i}, X_{v_j}] \leqslant \sum_{i=1}^{n} \text{Var}[X_{v_i}] \leqslant n.$$

Putting all these results together, we have that the following inequality:

$$P\left[\alpha(G) < (1-\epsilon)\frac{n}{6}\right] \leqslant \frac{36}{\epsilon^2 n} \overset{n\to\infty}{\to} 0. \tag{11}$$

This is a good upper bound when n grows but a bad one when ϵ decreases. In order to find an additional upper bound, we use Markov's inequality:

$$\begin{aligned} P\left[\alpha(G) < (1-\epsilon)\frac{n}{6}\right] &= P\left[\Delta > \frac{5+\epsilon}{6}n\right] \\ &\overset{(5)}{\leqslant} \frac{6}{(5+\epsilon)n}\mathbb{E}[\Delta] \\ &\overset{(10)}{\leqslant} \frac{6}{(5+\epsilon)n} \cdot \frac{5}{6}n \\ &= \frac{5}{5+\epsilon} \end{aligned}$$

Finally, we can state the following:

$$P\left[\alpha(G) < (1-\epsilon)\frac{n}{6}\right] \leqslant \min\left\{\frac{5}{5+\epsilon}; \frac{36}{\epsilon^2 n}\right\}.$$

Exercise 2a. Let $G = (V, E)$ be an undirected graph, and let $\mathcal{M} = (E, \mathcal{I})$ denote the graphic matroid of G. Define

$$\mathcal{I}' = \{S \subset E : E \setminus S \text{ contains a base of } \mathcal{I}\}.$$

Prove that $\mathcal{M}' = (E, \mathcal{I}')$ is a matroid.
Answer: Recall the two properties that independent sets must satisfy for them to form a matroid $M = (E; I)$:

$$\text{if } X \subseteq Y \text{ and } Y \in \mathcal{I} \text{ then } X \in \mathcal{I}, \tag{12}$$

$$\text{if } X, Y \in \mathcal{I} \text{ and } |Y| > |X| \text{ then } \exists e \in Y \setminus X : X \cup \{e\} \in \mathcal{I}. \tag{13}$$

Let's show that both of these properties are satisfied.

1. Let $S \in \mathcal{I}'$ and $A \subset E$ such that $A \subseteq S$. Since $S \in \mathcal{I}'$, $E \setminus S$ contains a base of $\mathcal{I}$. Note that $(E \setminus S) \subseteq (E \setminus A)$ because $A \subseteq S$. Therefore $E \setminus A$ also contains at least one base of $\mathcal{I}$ (the same as $E \setminus S$ and maybe other ones). Hence $A \in \mathcal{I}'$.

2. Let $S, T \in \mathcal{I}$ such that $|T| > |S|$, and let $\Delta := T \setminus S \neq \emptyset$.

   First, we show that Δ is not a subset of every base $B_i \in \mathcal{I}$ that are in $E \setminus S$. Otherwise, there would be no element of Δ that can be added to S so that $E \setminus S$ contains a basis of I. We do this by constructing such a case.
   Let $B_T \in E \setminus T$ and $B_S \in E \setminus S$ be two bases of $\mathcal{I}$ (we know B_T and B_S exist because $T, S \in \mathcal{I}'$). If $\Delta \nsubseteq B_S$ then we have a delta that is not a subset of every base $B_i \in \mathcal{I}$ so $B_S \in E \setminus S \cup \Delta$. So assume $\Delta \subseteq B_S$. Recall that all the bases of a matroid have the same size. Therefore $|B_T| = |B_S|$. Now consider $B_{T\setminus S} := B_T \setminus S$. Since at most $|S \setminus T|$ elements of B_T are in S, we have that

   $$|S \setminus T| \geqslant |B_T| - |B_{T\setminus S}| \tag{14}$$

   Furthermore, using property (12), we have that $B_{T\setminus S} \in \mathcal{I}$. We now distinguish two cases:

   - $B_T \cap S = \emptyset$: In this case, $B_T \cap S = \emptyset$ and $\Delta \nsubseteq B_T$ (in fact $\Delta \cap B_T = \emptyset$). Following the same reasoning as above for $\Delta \nsubseteq B_S$, we find an element that can be added to S so that $E \setminus \cup e$ still has a base.
   - $B_T \cap S \neq \emptyset$: In this case, $|B_{T\setminus S}| < |B_S|$. Defining $m = |B_S| - |B_{T\setminus S}|$, we can construct a new base B by adding m new elements from B_S to $B_{T\setminus S}$ using property (13) m times consecutively. In other words:

     $$B = B_{T\setminus S} \cup \left(\bigcup_{j=1}^{m} \{e_j\} \right), \qquad e_j \in B_S \quad j = 1, .., m. \tag{15}$$

     Finally, inequality (14) gives us that $m \leqslant |S \setminus T|$. Thus, since we've added at most $|S \setminus T|$ elements from B_S and since

     $$|B_S \cap \Delta| = |\Delta| = |T \setminus S| = |T| - |T \cap S| > |S| - |T \cap S| = |S \setminus T|,$$

     there exists at least one edge in Δ that has not been added to B, i.e. $\Delta \nsubseteq B$. Again, as above, we have found a base to which Δ is not a complete subset of.

We've therefore shown that there exists at least one base $B \in \mathcal{I}$ such that $\Delta \setminus B \neq \emptyset$. Taking $e \in \Delta \setminus B \subseteq T \setminus S$, we have that $S \cup \{e\} \in \mathcal{I}'$ since $B \subseteq E \setminus (S \cup \{e\})$.

Since properties (12) and (13) hold for $\mathcal{M}' = (E, \mathcal{I}')$, it concludes the proof that $\mathcal{M}'$ is a matroid.

Exercise 2b. Give a polynomial time algorithm that, given an undirected graph $G = (V, E)$, determines if G contains two edge disjoint spanning trees.
Answer: Let $\mathcal{M} = (E, \mathcal{I})$ be the graphic matroid on G and $\mathcal{M}' = (E, \mathcal{I}')$ the matroid defined in exercise 2a. Note that, in order to contain a spanning tree, the size of all the bases of $\mathcal{M}$ must equal $|V| - 1$ (if they are bigger there is a cycle and if they are smaller then at least one vertex is not covered). Also, note that if a base $X \in \mathcal{I}$ is also in $\mathcal{I}'$, then we know, by definition of $\mathcal{I}'$, that there exists at least one base X' which is edge disjoint with X. We can therefore state that there exists two edge disjoint spanning trees in G if and only if

$$\max_{X \in \mathcal{I} \cap \mathcal{I}'} |X| = |V| - 1.$$

This is clearly a matroid intersection problem as we want to find the size of the largest common independent set in $\mathcal{I}$ and $\mathcal{I}'$ and then check if its size equals $|V| - 1$. The main idea of an algorithm that can solve matroid intersection problems (which is strongly inspired by *Edmonds' Matroid Intersection Algorithm*) is presented below:

1. Start with a trivial set $I_0 \in \mathcal{I} \cap \mathcal{I}'$.
2. Find (if they exist) two sets $A \in I_0$ and $B \in E \setminus I_0$ such that $|B| > |A|$ and $I_1 := (I_0 \setminus A) \cup B \in \mathcal{I} \cap \mathcal{I}'$.
3. Repeat step 2) with I_1 to find I_2, then with I_2 to find I_3, and so on, until it is impossible to find the two sets A and B.
4. Return the last set I_n that could be found.

Now that we have the main idea of the algorithm, we still need to do the following in order to adapt it to our problem and to make sure it always solves it:

1. ***Prove that, in our case, we can indeed find a trivial common independent set*** $I_0 \in \mathcal{I} \cap \mathcal{I}'$ ***(or find the solution to our problem by searching for it)***:
   In order to find I_0, we can run GREEDY on the graphic matroid $\mathcal{M} = (E, \mathcal{I})$ with edge weights $w : E \to \mathbb{R}$, $e \mapsto w(e) = 1$. Let I be the set returned by GREEDY. By the properties of matroids, we know that I is a base of $\mathcal{M}$. Several cases can appear:
   - $|I| < |V| - 1$: In this case, we know that G does not contain any spanning trees, let alone two edge disjoint spanning trees.
   - $|I| = |V| - 1$ and $E \setminus I = \emptyset$: In this case, the spanning tree is the whole set of edges E and G cannot contain two edge disjoint spanning trees.
   - $|I| = |V| - 1$ and $E \setminus I \neq \emptyset$: In this case, any set $I_0 = \{e\}$ such that $e \in E \setminus I$ will satisfy $I_0 \in \mathcal{I} \cap \mathcal{I}'$. This is because a set of one edge does not contain a cycle $\Rightarrow I_0 \in \mathcal{I}$, and because by construction, there exist a base in $E \setminus \{e\} \Rightarrow I_0 \in \mathcal{I}'$.
2. ***Prove that the algorithm terminates***:
   Since $|I_{k+1}| > |I_k|, \forall k = 0, ..., n - 1$, and since $|E| < +\infty$, this algorithm will terminate for sure (I_n is at most E).

3. ***Prove that the algorithm returns one of the largest common independent set in*** $\mathcal{I} \cap \mathcal{I}'$:
   By construction, the sets $I_k, \forall k = 0, ..., n$ are common independent sets of the matroid intersection $\mathcal{M} \cap \mathcal{M}'$. Now, assume that I_n the set returned by the above algorithm, i.e. there isn't any sets $A \in I_n$ and $B \in E \setminus I_n$ such that $|B| > |A|$ and $(I_n \setminus A) \cup B \in \mathcal{I} \cap \mathcal{I}'$, and assume that I_n is not the largest common independent set, i.e. I_n is not a base of the matroid intersection so $\exists \hat{I} \in \mathcal{I} \cap \mathcal{I}'$ such that $|I_n| < |\hat{I}|$. Let $A' := I_n \setminus \hat{I}$ and $B' := \hat{I} \setminus I_n$. It is easy to see that A' and B' satisfy the properties that A and B must satisfy, respectively:

   - $A' = I_n \setminus \hat{I} \in I_n$,
   - $B' = \hat{I} \setminus I_n \in E \setminus I_n$,
   - $|B'| - |A'| = |\hat{I} \setminus I_n| - |I_n \setminus \hat{I}| = (|\hat{I}| - |I_n \cap \hat{I}|) - (|I_n| - |I_n \cap \hat{I}|) = |\hat{I}| - |I_n| > 0$,
   - $(I_n \setminus A') \cup B' = (I_n \setminus (I_n \setminus \hat{I})) \cup (\hat{I} \setminus I_n) = (I_n \cap \hat{I}) \cup (\hat{I} \setminus I_n) = \hat{I} \in \mathcal{I} \cap \mathcal{I}'$.

   This leads to a contradiction to the fact that I_n is the set returned by the algorithm (because A and B exists) and concludes the proof that I_n is one of the largest common independent set in $\mathcal{I} \cap \mathcal{I}'$.

4. ***Design a method to find the two sets*** A ***and*** B ***if they exists***:
   In order to find the sets A and B, one can use a similar approach to the one used in the AUGMENTINGPATH algorithm for bipartite perfect matching problems. The goal is to create, at each iteration k of the main algorithm, a directed graph D_I^k whose vertices $V_{D_I^k}$ are the edges of the main graph E, and whose edges $E_{D_I^k}$ are separated in two sets: those that go from $E \setminus I_k$ to E, and those that go from E to $E \setminus I_k$. Let $E^1_{D_I^k}$ and $E^2_{D_I^k}$ be these two disjoint sets of directed edges, respectively. With this definition of D_I^k, one can already see that if we can find a path P_k in D_I^k that goes from $E \setminus I_k$ to $E \setminus I_k$, and if we let A be the set of vertices of P_k that are in I_k and B the set of vertices of P_k that are in $E \setminus I_k$, we have that A and B satisfy the following properties:

   - $A \in I_k$ and $B \in E \setminus I_k$,
   - $|B| > |A|$

   The last thing we have to do is to make sure that $(I_k \setminus A) \cup B \in \mathcal{I} \cap \mathcal{I}'$ by carefully choosing the edges of D_I^k. Let $E^1_{D_I^k}$ and $E^2_{D_I^k}$ be defined as follows:

$$E^1_{D_I^k} := \{(u, v) : u \in E, v \in E \setminus I, (I \setminus \{u\}) \cup \{v\} \in \mathcal{I}\},$$
$$E^2_{D_I^k} := \{(v, u) : u \in E, v \in E \setminus I, (I \setminus \{u\}) \cup \{v\} \in \mathcal{I}'\}.$$

   Moreover, let $V^1_{D_I^k}$ and $V^2_{D_I^k}$ be defined as follows:

$$V^1_{D_I^k} := \{v \in E \setminus I : I \cup \{v\} \in \mathcal{I}\},$$
$$V^2_{D_I^k} := \{v \in E \setminus I : I \cup \{v\} \in \mathcal{I}'\}.$$

   Lastly, let $P_k = v_0 u_1 v_1 ... u_n v_n$ be the shortest path between $V^1_{D_I^k}$ and $V^2_{D_I^k}$ in D_I^k. It can be proved that $A := I_k \cap \{u_i\}_{1 \leqslant i \leqslant n}$ and $B := \{v_i\}_{0 \leqslant i \leqslant n}$ satisfy $(I_k \setminus A) \cup B \in \mathcal{I} \cap \mathcal{I}'$.

This proof is rather technical and will not be exposed here, but it can be found here: Lemma 13.28 [1]. In addition, the shortest path can be found in polynomial time by simply computing the shortest path from every vertices in $V^1_{D^k_I}$ to every vertices in $V^1_{D^k_I}$.

5. ***Design methods to answer the queries*** *"*$I \in \mathcal{I}?$*"* ***and*** *"*$I \in \mathcal{I}'?$*"*:

   - To answer "$I \in \mathcal{I}?$", one can simply run DFS on the graph $G = (V, I)$ and check for any visited vertex v if there exists an adjacent vertex u that has already been visited such that v is not a descendent of u. If such a vertex u is found before DFS terminates, then $I \notin \mathcal{I}$. Else $I \in \mathcal{I}$. $\text{DFS}(V, E)$ runs in time $O(|E| + |V|)$ and the query can therefore be answered in time $O(|E| + |I|)$.
   - To answer "$I \in \mathcal{I}'?$", one can run GREEDY with edge weights $w : (E \setminus I) \to \mathbb{R}$, $e \mapsto w(e) = 1$ on the graphic matroid obtained with the graph $G = (V, E \setminus I)$ (one can use the algorithm above to answer the queries "$I \in \mathcal{I}?$" in GREEDY). If the base returned by GREEDY has size $|V| - 1$, then $I \in \mathcal{I}'$. Else $I \notin \mathcal{I}'$. This can be answered in time $O(|E|(|E| + |V|))$.

Now that we have an algorithm to find a maximal common independent set $I_n \in \mathcal{I} \cap \mathcal{I}'$, we simply need to output

- *"*G *contains two edge disjoint spanning trees"* if $|I_n| = |V| - 1$.
- *"*G *does not contain two edge disjoint spanning trees"* if $|I_n| < |V| - 1$.

Finally, since this algorithm is a combination of loops and polynomial time elementary algorithms (no recursion), it is clear that its runtime is also polynomial.

Exercise 3a. Show that for every $k = 1, ..., n$ one has $price(e_k) \leqslant OPT/(n-k+1)$
Answer. Firstly, we will introduce some notation to capture the partial solutions of each iteration of the algorithm. In the following, by an *iteration* of the algorithm we mean the completion of one repetition of the while loop. Let m be the number of sets chosen by the greedy algorithm and $S_1, \ldots, S_m$ be the sequence of the chosen sets ordered by the time they were added to the solution set. Let $C_k = \cup_{i=1}^{k} S_i$ for $k = 1, \ldots, m$ and $C_0 = \emptyset$ be the partial cover built after the k-th iteration of the algorithm. Let $A_k = S_k \setminus C_{k-1}$ for $k = 1, \ldots, m$ be the set of elements in the universe covered only after adding on the set S_k to the solution.

Proof. Let $S_1^*, \ldots, S_{|OPT|}^* \in \mathcal{F}$ be an optimal cover. For a fixed iteration $j \in \{1, \ldots, m\}$ of the algorithm, using the fact that $\{S_i^*\}$ is a cover, i.e. $U = \bigcup_i S_i^*$, the number of so far uncovered elements can be upper bounded as follows:

$$\begin{aligned} |U \setminus C_{j-1}| &= \left| \left(\bigcup_i S_i^* \right) \setminus C_{j-1} \right| \\ &= \left| \bigcup_i (S_i^* \setminus C_{j-1}) \right| \\ &\leqslant \sum_i |S_i^* \setminus C_{j-1}| \\ &\leqslant \sum_i |S_j \setminus C_{j-1}| = OPT \times |A_j| \end{aligned}$$

where the the last inequality follows from the greedy choice of S_j. As $price(e) = \frac{1}{|A_j|}$ for all $e \in A_j$ and $|U \setminus C_{j-1}| = n - |C_{j-1}|$, it follows from the above inequality that the following upper bound holds for the prices assigned by Greedy:

$$price(e) \leqslant \frac{OPT}{n - |C_{j-1}|}, \forall e \in A_j \text{ and } j = 1, \ldots, m.$$

As the elements of A_j appear to the right of the elements of C_{j-1} in the sequence $e_1, \ldots, e_n$ ordered by the time of addition, it follows that

$$\begin{aligned} e_k \in A_j \Rightarrow k > |C_{j-1}| &\Rightarrow k - 1 \geqslant |C_{j-1}| \\ &\Rightarrow price(e_k) \leqslant \frac{OPT}{n - |C_{j-1}|} \leqslant \frac{OPT}{n-k+1}. \end{aligned}$$

Remark: Already at this stage it is possible to foreshadow the result that will become apparent only at the end of the exercise. Namely, that summing through the prices of all the elements, we find that

$$\sum_{k=1}^{n} price(e_k) \leqslant \sum_{k=1}^{n} \frac{OPT}{n-k+1} = OPT \sum_{k=1}^{n} n\frac{1}{k} = H_n \times OPT.$$

But the sum of all the prices of the elements is equal to the number of sets in the solution:

$$\sum_{i=1}^{n} prices(e_i) = \sum_{j=1}^{m} \sum_{k=1}^{|A_j|} \frac{1}{|A_j|} = \sum_{j=1}^{m} 1 = m$$

Hence: $OPT_{greedy} \leqslant H_n OPT$. ☺

Exercise 3b. The point $(y_1, \ldots, y_e) \in \mathbb{R}^n$ with $y_e = \frac{\text{price}(e)}{H_n}$ is a point inside the feasible polytope of the dual LP if and only if

$$\sum_{e \in S} y_e \leqslant 1 \iff \sum_{e \in S} \text{price}(e) \leqslant H_n, \ \forall\ S \in \mathcal{F}$$

since the $y_e \geqslant 0$ conditions are already trivially satisfied. By the construction of $\text{price}(e)$, the above sum can be written explicitly by summing the number of elements added at every iteration of the algorithm:

$$\sum_{e \in S} \text{price}(e) = \sum_{i=1}^{m} \frac{|S \cap A_i|}{|A_i|} \leqslant \sum_{i=1}^{m} \frac{|S \cap A_i|}{|S \setminus C_{i-1}|} \tag{16}$$

where the last inequality follows from the greedy step that $|S \setminus C_{i-1}| \leqslant |A_i| = |S_i \setminus C_{i-1}|$.

$$\sum_{i=1}^{m} \frac{|S \cap A_i|}{|S \setminus C_{i-1}|} = \sum_{i=1}^{m} \frac{|S \setminus C_{i-1}| - |S \setminus C_i|}{|S \setminus C_{i-1}|} \tag{17a}$$

$$= \sum_{i=1}^{m} \sum_{j=|S \setminus C_i|+1}^{|S \setminus C_{i-1}|} \frac{1}{|S \setminus C_{i-1}|} \tag{17b}$$

$$\leqslant \sum_{i=1}^{m} \sum_{j=|S \setminus C_i|+1}^{|S \setminus C_{i-1}|} \frac{1}{j} \qquad (|S \setminus C_{i-1}| \geqslant j) \tag{17c}$$

$$= \sum_{i=1}^{m} \left(H_{|S \setminus C_{i-1}|} - H_{|S \setminus C_i|} \right) \tag{17d}$$

$$= H_{|S \setminus C_0|} - H_{|S \setminus C_m|} = H_n \tag{17e}$$

Combining Equation 16 and Equation 17 yields

$$\sum_{e \in S} \text{price}(e) \leqslant H_n \implies \sum_{e \in S} y_e = \sum_{e \in S} \text{price}(e)/H_n \leqslant 1$$

proving that $(y_1, \ldots, y_n)$ is a feasible solution.

Exercise 3c. *Proof.* Let x_S, $S \in \mathcal{F}$ be the optimal cover such that $|\text{OPT}| = \sum_{S \in \mathcal{F}} x_S$. Obviously, since x_S is a cover, it is feasible: there exists at least one $S \in \mathcal{F}$ such that $e \in S$ and $x_S = 1$.

In the exercise 3b we have shown that $y_e = \frac{\text{price}(e)}{H_n}$ is a feasible solution of the dual LP. By the Weak Duality theorem, it follows that

$$|\text{OPT}| = \sum_{S \in \mathcal{F}} x_S \geqslant \sum_{e \in U} y_e = \frac{1}{H_n} \sum_{e \in U} \text{price}(e) = \frac{|\text{OPT}^G|}{H_n}$$

Hence the greedy solution is an H_n-approximation of the optimal set cover problem:

$$|\text{OPT}^G| \leqslant H_n \times |\text{OPT}|.$$

☺

Exercise 4a. Give a feasible solution to the LP relaxation of Set Cover on the instance below with value bounded by 2.

$$\begin{aligned} \mathcal{U} = \left\{ x \in \{0,1\}^d : \textstyle\sum_{i=1}^{d} x_i = \frac{d}{2} \right\} \quad & \text{with } d \geqslant 2 \text{ an even integer,} \\ \mathcal{F} = \{S_1, ..., S_m\} \quad & \text{with } m = d, \\ S_i = \{x \in \mathcal{U} : x_i = 1\} & \\ c(S_i) = 1 \quad & \text{for } i = 1, ..., m. \end{aligned} \tag{18}$$

Answer: The LP relaxation of Set Cover is expressed as follows:

$$\textbf{Minimize:} \quad \sum_{i=1}^{m} \alpha_i c(S_i) \tag{19}$$

$$\textbf{Subject to:} \quad \sum_{S_i : x \in S_i} \alpha_i \geqslant 1 \qquad \forall x \in \mathcal{U} \tag{20}$$

$$\alpha_i \in [0,1] \qquad i = 1, ..., m \tag{21}$$

Observe that by construction of $\mathcal{U}$, every element $x \in \mathcal{U}$ belongs to exactly $d/2$ sets S_i. Let's now prove that $\alpha^* \in \mathbb{Q}^m$ with $\alpha_i^* = 2/d$, $i = 1, \ldots, m$ is a feasible solution with a cost bounded by 2. Firstly, using the previous observation, constraint (20) is satisfied because

$$\sum_{S_i : x \in S_i} \alpha_i^* = \sum_{S_i : x \in S_i} \frac{2}{d} = \frac{d}{2} \cdot \frac{2}{d} = 1, \qquad \forall x \in \mathcal{U}. \tag{22}$$

Secondly, since $m = d \geqslant 2$, constraint (21) is satisfied because $0 < \alpha_i^* = 2/d \leqslant 1$, $i = 1, ..., m$. Lastly, the cost (19) of solution α^* is given by

$$\sum_{i=1}^{m} \alpha_i^* \underbrace{c(S_i)}_{=1} = \sum_{i=1}^{m} \alpha_i^* = \sum_{i=1}^{m} \frac{2}{d} = m \cdot \frac{2}{d} = d \cdot \frac{2}{d} = 2. \tag{23}$$

This concludes the proof that α^* is a feasible solution with a cost not bigger than 2.

Exercise 4b. Prove that at least $d/2 + 1$ sets are needed to cover the universe $\mathcal{U}$.
Answer:

- Firstly, let's show that for any collection of sets $\mathcal{G}$ with size $|\mathcal{G}| \leqslant \frac{d}{2}$, there exists at least one element $\hat{x} \in \mathcal{U}$ that is not covered by $\mathcal{G}$:

  Let $I = \{i : S_i \notin \mathcal{G}\}$. If $|\mathcal{G}| \leqslant \frac{d}{2}$, then $|I| \geqslant \frac{d}{2}$ and there exists at least one set $I' \subseteq I$ with size $|I'| = \frac{d}{2}$. Moreover, let $\hat{x} \in \{0,1\}^d$ be defined as follows:

  $$\hat{x}_i = \begin{cases} 1 & \text{if} \quad i \in I', \\ 0 & \text{if} \quad i \in \{1, .., d\} \setminus I'. \end{cases}$$

  Observe that since $|I'| = \frac{d}{2}$, $\hat{x} \in \mathcal{U}$. Finally, by construction of $\hat{x}$ (as it indicates sets that were not chosen), we have that $\hat{x} \notin S_i$, $\forall S_i \in \mathcal{G}$. This concludes the proof that the universe $\mathcal{U}$ cannot be covered with a collection of set $\mathcal{G}$ with size strictly lower than $\frac{d}{2} + 1$.

- Secondly, let's show that a collection of sets $\mathcal{G}$ with size $|\mathcal{G}| \geqslant \frac{d}{2} + 1$ covers every element $x \in \mathcal{U}$:

  As noted previously, every element $x \in \mathcal{U}$ belongs to exactly $\frac{d}{2}$ sets S_i. At least one of these sets must belong to any collection of sets $\mathcal{G}$ with size $|\mathcal{G}| \geqslant \frac{d}{2} + 1$ (because at most $\frac{d}{2} - 1$ sets don't belong to such a $\mathcal{G}$). This concludes the proof.

Exercise 4c. Conclude that the integrality gap of the LP relaxation of Set Cover on a universe $\mathcal{U}$ of size n is $\Omega(\log n)$.
Answer: Let $\mathcal{I}$ be the set of all instances of a Set Cover problem with universe of size n. The integrality gap g of the LP relaxation of Set Cover is defined as follows:

$$g = \max_{I \in \mathcal{I}} \frac{OPT(I)}{OPT_{LP}(I)},$$

where $OPT(I)$ and $OPT_{LP}(I)$ are optimal solutions for the instance I of Set Cover and its LP relaxation, respectively. Let $\hat{I}$ denote the instance of the Set Cover problem discussed in this exercise and defined by (18). We found that $OPT_{LP}(\hat{I}) \leqslant 2$ (Eq. (23)) and that $OPT(\hat{I}) \geqslant \frac{d}{2} + 1$ since any collection of set covering the universe $\mathcal{U}$ must contain at least $\frac{d}{2} + 1$ sets. This yields

$$g = \max_{I \in \mathcal{I}} \frac{OPT(I)}{OPT_{LP}(I)} \geqslant \frac{OPT(\hat{I})}{OPT_{LP}(\hat{I})} \geqslant \frac{\frac{d}{2} + 1}{2} \geqslant \frac{d}{4}. \tag{24}$$

Finally, note that the size n of the universe $\mathcal{U}$ is exactly the number of ways you can choose $\frac{d}{2}$ elements among d, i.e. $n = \binom{d}{d/2}$. However, since $\mathcal{U}$ is a subset of $\{0,1\}^d$, its size n is smaller than 2^d. In other words, $\log(n) \leqslant d$. Using this and Eq. (24), we have that

$$g \geqslant \frac{d}{4} \geqslant \frac{\log(n)}{4} \quad \Rightarrow \quad g \in \Omega(\log(n)).$$

This concludes the proof.

Exercise 5. The code of the implementation is attached in the appendix. The successful submission on the Codeforces Online Judge is:
`https://codeforces.com/group/TUksowRwAk/contest/272033/submission/73203204`
by the contestant **PineMarten**.

Appendix

C++ Code

```cpp
#include <iostream>
#include <fstream>
#include <vector>
#include <algorithm>
#include <set>

using namespace std;

```

```
9
10 void solve(){
11     int n, m;
12     cin >> n >> m;
13
14     auto comparator = [](const auto left, const auto right){
15         if(left.second.size() == right.second.size())
16             return left.first < right.first;
17         return left.second.size() < right.second.size();
18     };
19     vector<pair<int, set<int>>> collection;
20
21     for(int i = 0; i < m; ++i){
22         int sz;
23         cin >> sz;
24         set<int> subset;
25         for(int j = 0; j < sz; ++j){
26             int a;
27             cin >> a;
28             subset.insert(a);
29         }
30         collection.push_back({i + 1, subset});
31     }
32     sort(collection.begin(), collection.end(), comparator);
33
34
35     vector<int> solution;
36     vector<int> inv_price(n + 1, 0);
37
38     int cover_size = 0;
39     while(cover_size < n){
40         auto argmax = *collection.rbegin();
41         collection.pop_back();
42
43         // Add the index to the solution vector
44         solution.push_back(argmax.first);
45         cover_size += argmax.second.size();
46
47         for(auto el: argmax.second){
48             inv_price[el] = argmax.second.size();
49             for(auto &subset: collection){
50                 if(subset.second.find(el) != subset.second.end()){
51                     subset.second.erase(el);
52                 }
53             }
54         }
55         sort(collection.begin(), collection.end(), comparator);
56     }
57
58     double Hn = 0;
59     for(int i = 1; i <= n; ++i){
60         Hn += 1.0L / i;
61     }
62
63     cout << solution.size() << "\n";
64     for(auto el: solution){
65         cout << el << " ";
66     }
67     cout << "\n";
68     for(int i = 1; i <= n; ++i){
```

```
        cout << (1.0L / inv_price[i])/Hn << " ";
    }


}

int main(){
    std::cin.tie(0);
    std::ios_base::sync_with_stdio(0);
    std::cout << std::setprecision(16) << fixed;

    solve();

    return 0;
}
```

References

[1] T.Summary, P.Wilhelm: Greedy Algorithm And Matroid Intersection Algorithm, https://www.mathematik.hu-berlin.de/~wilhelm/greedy-ausarbeitung.pdf