Algorithm Design and Analysis
Assignment 6
Deadline: January 7, 2023

1. (30 points) Given an undirected unweighted graph $G = (V, E)$, prove that it is NP-complete to decide if there is a simple path with length exactly $|V|/2$ (a simple path is a path that do not visit a vertex more than once).

   The problem is clearly in NP, as a simple path with length exactly $|V|/2$ can be served as a certificate. To show it is NP-complete, we present a reduction from HamiltonianPath. Given a HamiltonianPath instance $G = (V, E)$, we construct an instance G' of this problem by adding $|V|$ isolated vertices to the graph G. It is obvious that the HamiltonianPath instance is a yes instance if and only if the constructed instance for this problem is a yes instance. This is because G has a simple path of length $|V|$ if and only if G' has a simple path using exactly half of the vertices in G'.

   Notice that I am assuming the length of the path is defined by the number of the vertices on the path. If it is defined by the number of the edges, we add $|V| - 2$ isolated vertices instead of $|V|$ vertices (for HamiltonianPath instances with $|V| \leq 2$, we can first decide if it is a yes instance or a no instance and then map it to a trivial yes/no instance of this problem).

2. (30 points) Suppose we would like to complete n jobs. Each job i has a release time $r_i \in \mathbb{Z}^+$ and a deadline $d_i \in \mathbb{Z}^+$, and it requires $t_i \in \mathbb{Z}^+$ units of time to complete. Suppose we can only process one job at a time (i.e., we cannot simultaneously process two jobs), and a job must be done within a consecutive time interval (i.e., we cannot pause a job and resume it at a later time). Prove that it is NP-complete to decide if there is a schedule so that all the jobs are completed.

   The problem is in NP, as a scheduling of all the jobs (for example, encoding the starting times of all the jobs) can be served as a certificate for the yes instance.

   To show it is NP-complete, we reduce it from SubsetSum+. Given a SubsetSum+ instance $(S = \{a_1, \dots, a_n\}, k)$, we construct a scheduling instance as follows. We construct n jobs corresponding to $a_1, \dots, a_n$. The i-th job has release time $r_i = 1$ and deadline $d_i = 2 + \sum_{i=1}^n a_i$, and set $t_i = a_i$. We construct the $(n+1)$-th job with $r_{n+1} = k + 1$, $d_{n+1} = k + 2$, and $t_{n+1} = 1$.

   If the SubsetSum+ instance is a yes instance, there exists a subcollection T of S whose elements sum up to k. We can schedule all the $n+1$ jobs properly: the jobs corresponding to the elements in T are completed in the time interval $[1, k+1]$, the $(n+1)$-th job starts at time $k+1$ and ends at time $k+2$, and the jobs corresponding to the elements in $S \setminus T$ are completed in the time interval $[k+2, 2+\sum_{i=1}^n a_i]$.

   If the scheduling instance is a yes instance, we will show that there exists a subcollection T in the SubsetSum+ instance whose elements sum up to k. Firstly, the $(n+1)$-th job must be done in the time interval $[k+1, k+2]$. The remaining jobs needs to be scheduled in one of the two intervals $[1, k+1]$ and $[k+2, 2+\sum_{i=1}^n a_i]$. Two intervals have total length $\sum_{i=1}^n a_i$, and the remaining n jobs need time $\sum_{i=1}^n a_i$. This implies the remaining n jobs must "fit" into the two intervals exactly. Let T' be the subset of the jobs completed in the first interval $[1, k+1]$. Then the subcollection T corresponding to T' has its elements sum up to k.

3. (40 points) An *integer linear program* is similar to a linear program, except that each variable x_i is required to be an integer.

$$\begin{aligned} \text{maximize} \quad & c_1x_1 + \cdots + c_nx_n \\ \text{Subject to} \quad & a_{11}x_1 + \cdots + a_{1n}x_n \leq b_1 \\ & a_{21}x_1 + \cdots + a_{2n}x_n \leq b_2 \\ & \quad \vdots \\ & a_{m1}x_1 + \cdots + a_{mn}x_n \leq b_m \\ & x_1, \ldots, x_n \in \mathbb{Z}_{\geq 0} \end{aligned}$$

We have seen in the class that a linear program can be solved in polynomial time. However, we will see in this question that this is unlikely for integer linear programs.

(a) (30 points) Prove that, for a given input k, deciding if there is a feasible solution $(x_1, \ldots, x_n)$ such that $c_1x_1 + \cdots + c_nx_n \geq k$ is NP-complete.

(b) (10 points) Prove that it is NP-complete to even decide if there is a feasible solution.

(a) It is relatively easy to show that the problem is NP-hard. Question 4(a) in Assignment 5 gives a reduction from VertexCover to our problem. Given a VertexCover instance $(G = (V, E), k)$, we construct the following integer linear program.

$$\begin{aligned} \text{maximize} \quad & \sum_{u \in V} -x_u \\ \text{subject to} \quad & -x_u - x_v \leq -1 && (\forall (u,v) \in E) \\ & x_u \leq 1 && (\forall u \in V) \\ & x_u \in \mathbb{Z}_{\geq 0} && (\forall u \in V) \end{aligned}$$

Since VertexCover is NP-complete, it is NP-hard to decide if there is a feasible solution $(x_u)_{u \in V}$ such that $\sum_{u \in V} -x_u \geq -k$.

It is much harder to prove that the problem is in NP. Naturally, a feasible solution is a certificate to the yes instance and it can be verified if the solution is indeed feasible and if $c_1x_1 + \cdots + c_nx_n \geq k$. However, the input of this problem consists of all those c_i, a_{ij}, b_j. We need to prove that there is a certificate $(x_1, \ldots, x_n)$ having a length that is polynomial in terms of the length of the string encoding those c_i, a_{ij}, b_j. This is not easy to prove. A proof of this claim can be found here:

https://cstheory.stackexchange.com/questions/34337/are-there-integer-programs-with-small-coefficients-that-only-have-large-solution

(b) The NP-hardness follows straightforwardly from part (a) by adding $-c_1x_1 - \cdots - c_nx_n \leq -k$ as an extra constraint.

4. (Bonus 60 points) **[Ultimate Cake Cutting with Yuhao]** Consider the *cake cutting* problem in Question 5 of Assignment 1. Recall that the objective is to allocate the cake to the n agents *fairly* such that each agent believes he receives the average value based on his preference. The formally model is recalled below.

   The cake is modelled by the 1-dimensional interval $[0, 1]$. Each agent i has a *value density function* $f_i : [0, 1] \to \mathbb{R}_{\geq 0}$ representing i's preference over the cake $[0, 1]$. Given an interval $I \subseteq [0, 1]$, agent i's *value* on I is given by the Riemann integral $\int_I f_i(x)dx$. An *allocation* is a collection of n intervals $(I_1, \ldots, I_n)$ such that every pair of intervals I_i and I_j can only intersect at the endpoints, where I_i is the interval allocated to agent i. An allocation $(I_1, \ldots, I_n)$ is *proportional* if $\int_{I_i} f_i(x)dx \geq \frac{1}{n}\int_0^1 f_i(x)dx$ for each agent i, i.e., each agent i thinks he receives the average value based on his value density function.

   In this question, we assume that each value density function f_i is piecewise-constant. That is, for each f_i, the cake $[0, 1]$ can be partitioned into finitely many intervals $[0, x_1), [x_1, x_2), \ldots, [x_{k-1}, x_k), [x_k, 1]$ such that f_i has a constant value on each of these intervals.

   In Question 5 of Assignment 1, we have seen that a proportional allocation always exists and can be computed efficiently by Even-Paz algorithm. In this question, we will require just slightly more than proportionality.

   Suppose Professor Yuhao Zhang is one of the n agents, and we would like to give him slightly more than his proportional value for his excellent service in AI2615. In particular, suppose Yuhao is agent 1. We would like to have an allocation $(I_1, \ldots, I_n)$ such that $\int_{I_1} f_1(x)dx \geq \frac{2}{n}\int_0^1 f_1(x)dx$ for Yuhao, and $\int_{I_i} f_i(x)dx \geq \frac{1}{n}\int_0^1 f_i(x)dx$ for each of the remaining agents i. We call such an allocation *Yuhao-prioritized proportional*.

   Of course, Yuhao-prioritized proportional allocation may not exist. For example, if all the agents have the same value density function $f_i(x) = 1$ on $[0, 1]$, the only proportional allocation is to allocate each agent an interval of length $\frac{1}{n}$, and Yuhao cannot be prioritized. In this question, we study the problem of deciding if Yuhao-prioritized proportional allocation exists for a given cake cutting instance.

   (a) (20 points) Suppose each agent is allowed to be allocated multiple intervals. That is, each I_i can be a union of finitely many intervals in an allocation $(I_1, \ldots, I_n)$. Show that deciding if Yuhao-prioritized proportional allocation exists is in P.

   (b) (40 points) Consider the original setting where each I_i is required to be a connected interval. Prove that deciding if Yuhao-prioritized proportional allocation exists is NP-complete.

   (a) We mark all the points of discontinuity for $f_1, \ldots, f_n$. This partition the cake $[0, 1]$ into multiple intervals, where each f_i is a constant on each of these intervals. Let m

be the total number of these intervals, and denote these intervals by $X_1, \ldots, X_m$. To decide an allocation, we only need to decide how each X_t is allocated to the n agents. Let x_{it} be the length that agent i gets from the interval X_t. Clearly, $\{x_{it}\}_{t=1,\ldots,m;i=1,\ldots,n}$ fully describes an allocation.

Let a_{it} be the value of f_i on X_t. Let $b_i = \frac{1}{n}\sum_{t=1}^{m} a_{it}|X_t|$ be $\frac{1}{n}$ of agent i's value on the entire cake $[0,1]$. Then we solve the following linear program.

$$\begin{aligned}
\text{maximize } & \sum_{t=1}^{m} a_{1t}x_{1t} \\
\text{subject to } & a_{21}x_{21} + \cdots + a_{2m}x_{2m} \geq b_2 \\
& a_{31}x_{31} + \cdots + a_{3m}x_{3m} \geq b_3 \\
& \vdots \\
& a_{n1}x_{n1} + \cdots + a_{nm}x_{nm} \geq b_n \\
& x_{1t} + x_{2t} + \cdots + x_{nt} = |X_t| && \text{(for each } t = 1, \ldots, m) \\
& x_{it} \geq 0 && \text{(for each } i = 1, \ldots, n \text{ and } t = 1, \ldots, m)
\end{aligned}$$

This linear program computes the maximum value Yuhao can receive, subject to that each of the remaining $n-1$ agents receives a value that is at least $\frac{1}{n}$ of his value on the entire cake. To decide if Yuhao-prioritized proportional allocation exists, we only need to see if the optimal solution to the linear program is at least $\frac{2}{n}\sum_{t=1}^{m} a_{1t}|X_t|$. Since a linear program can be solved in polynomial time, this problem is in P.

(b) Our problem of deciding if Yuhao-prioritized proportional allocation exists is clearly in NP, as the allocation itself can be a certificate to a yes instance.

To show it is NP-hard, we will reduce it from Partition+. First of all, we will see that we can assume the cake is represented by $[0, T]$ for some integer T, instead of $[0, 1]$. Notice that $[0, T]$ can be just recaled to $[0, 1]$. For example, say, $T = 100$. If a value density function is 1 on $[0, 0.5]$ and 2 on $(0.5, 1]$, it becomes 1 on $[0, 50]$ and 2 on $[50, 100]$. If an agent receives an interval $[0.3, 0.7]$ on $[0, 1]$, it will become $[30, 70]$ on $[0, 100]$.

Given a Partition+ instance with $a_1, \ldots, a_n$, we construct a cake cutting instance as follows. Let $m = 2n$. Let W be defined such that $2W = \sum_{i=1}^{n} a_i$. The cake is represented by $[0, 2W + 2m]$. There are $N = n + m$ agents indexed by $0, 1, \ldots, n + m - 1$. Let Yuhao be agent 0. Notice that Yuhao needs to get a $2/N$ fraction of the value on the entire cake, while each of the remaining agents needs to get a $1/N$ fraction. The value density functions are defined as follows.

$$f_0(x) = \begin{cases} 1 & x \in [2W+1, 2W+2] \cup [2W+3, 2W+4] \cup \cdots \cup [2W+2m-1, 2W+2m] \\ 0 & \text{otherwise} \end{cases}$$

Agent $1, \dots, n$ correspond to $a_1, \dots, a_n$ in the PARTITION+ instance, and their value density functions are defined as follows.

$$f_i(x) = \begin{cases} 1 & x \in [0, W] \cup [W+1, 2W+1] \\ a_i N - 2W & x \in [2W+4, 2W+5] \\ 0 & \text{otherwise} \end{cases}$$

The last $m-1$ agents' value density functions are defined as follows.

$$f_{n+1}(x) = \begin{cases} 1 & x \in [W, W+1] \\ 0 & \text{otherwise} \end{cases} \qquad f_{n+2}(x) = \begin{cases} 1 & x \in [2W+2, W+3] \\ 0 & \text{otherwise} \end{cases}$$

$$\text{for each } i = 3, \dots, m-1, \;\; f_{n+i} = \begin{cases} 1 & x \in [2W+2i, 2W+2i+1] \\ 0 & \text{otherwise} \end{cases}$$

Here are some observations which will lead to the proof of the NP-hardness of our problem.

1. Due to the existence of agent $n+1, n+2, \dots, n+m-1$, the cake is partitioned into m regions: $[0, W], [W+1, 2W+2], [2W+3, 2W+6], [2W+7, 2W+8], [2W+9, 2W+10], \dots, [2W+2m-1, 2W+2m]$. No agent in $0, 1, \dots, n$ can get an interval that spans across more than one region, for otherwise one of the last $m-1$ agents will receive no value at all.

2. Yuhao (agent 0) has a total value of m on the entire cake, so he must receive value at least value $m \cdot \frac{2}{N} = \frac{4}{3}$. This is only possible if Yuhao takes among his two valued intervals $[2W+3, 2W+4]$ and $[2W+5, 2W+6]$ in the third region (in each of the remaining regions, Yuhao's total value is 1, which is not enough). In this case, since Yuhao needs to get a connected interval, the interval that each of the agents in $1, \dots, n$ has most value on, namely, the "middle" interval $[2W+4, 2W+5]$, must be allocated to Yuhao.

3. For each $i = 1, \dots, n$, agent i total value on the cake is $a_i N$, so he must get a value of a_i to guarantee proportionality. In addition, based on the previous observation, each agent $i \in \{1, \dots, n\}$ must receive an interval in the first or the second region. Therefore, each agent $i \in \{1, \dots, n\}$ must receive a length of at least a_i to guarantee proportionality.

4. If all the agents from $1, \dots, n$ can receive an interval that meets the proportionality requirement, since $\sum_{i=1}^{n} a_i = 2W$, the agents can be partitioned into two sets corresponding to the first two regions such that the intervals received by the agents from each regions has a total length of W. This would imply the PARTITION+ instance is a yes instance.

Now, let us put these observations together and complete the proof.

Suppose the PARTITION+ instance is a yes instance, and there is a valid partition (C_1, C_2) of $a_1, \ldots, a_n$. We will describe a valid Yuhao-prioritized proportional allocation. Yuhao takes $[2W+3, 2W+6]$, which has value 2 and more than $2/N$ of his value on the whole cake. For each agent in $1, \ldots, n$, if agent i corresponds to a number in C_1, let him take a length of a_i from $[0, W]$; otherwise, let him take a length of a_i from $[W+1, 2W+1]$. This is possible as the PARTITION+ instance is a yes instance, and the proportionality requirement is satisfied due to Observation 3. Agent $n+1$ takes $[W, W+1]$. Agent $n+2$ takes $[2W+1, 2W+3]$. For $i = 3, \ldots, m-1$, agent $n+i$ takes $[2W+2i, 2W+2i+2]$. Each of the agents $n+1, \ldots, n+m-1$ gets an interval that is worth his total value on the cake. Therefore, this is a valid Yuhao-prioritized proportional allocation.

Suppose there is a valid Yuhao-prioritized proportional allocation. We will show that the PARTITION+ instance is a yes instance. This follows mostly from Observation 1, 2, 3, and 4.

5. (0 points) For practice and for fun, not for credits.

   (a) Given an undirected graph $G = (V, E)$ with $n = |V|$, decide if G contains a clique with size exactly $n/2$. Prove that this problem is NP-complete.

   (b) Consider the decision version of *Knapsack*. Given a set of n items with weights $w_1, \ldots, w_n \in \mathbb{Z}^+$ and values $v_1, \ldots, v_n \in \mathbb{Z}^+$, a capacity constraint $C \in \mathbb{Z}^+$, and a positive integer $V \in \mathbb{Z}^+$, decide if there exists a subset of items with total weight at most C and total value at least V. Prove that this decision version of Knapsack is NP-complete.

   (c) Given two undirected graphs G and H, decide if H is a subgraph of G. Prove that this problem is NP-complete.

   (d) Given an undirected graph $G = (V, E)$ and an integer k, decide if G has a spanning tree with maximum degree at most k. Prove that this problem is NP-complete.

   (e) Given a ground set $U = \{1, \ldots, n\}$, a collection of its subsets $\mathcal{S} = \{S_1, \ldots, S_m\}$, and a positive integer k, the *set cover* problem asks if we can find a subcollection $\mathcal{T} \subseteq \mathcal{S}$ such that $\bigcup_{S \in \mathcal{T}} S = U$ and $|\mathcal{T}| = k$. Prove that set cover is NP-complete.

   (f) Given a collection of integers (can be negative), decide if there is a subcollection with sum exactly 0. Prove that this problem is NP-complete.

   (g) In an undirected graph $G = (V, E)$, each vertex can be colored either black or white. After an initial color configuration, a vertex will become black if all its neighbors are black, and the updates go on and on until no more update is possible. (Notice that once a vertex is black, it will be black forever.) Now, you are given an initial configuration where all vertices are white, and you need to change k vertices from white to black such that all vertices will eventually become black after updates. Prove that it is NP-complete to decide if this is possible.

   (h) Suppose we want to allocate n items $S = \{1, \ldots, n\}$ to two agents. The two agents may have different values for each item. Let $u_1, u_2, \ldots, u_n$ be agent 1's values for those n items, and $v_1, v_2, \ldots, v_n$ be agent 2's values for those n items. An allocation is a partition (A, B) for S, where A is the set of items allocated to agent 1 and B is the set of items allocated to agent 2. An allocation (A, B) is *envy-free* if, based on each agent's valuation, (s)he believes the set (s)he receives is (weakly) more valuable than the set received by the other agent. Formally, (A, B) is envy-free if

$$\sum_{i \in A} u_i \geq \sum_{j \in B} u_j \qquad \text{agent 1 thinks } A \text{ is more valuable}$$

   and

$$\sum_{i \in B} v_i \geq \sum_{j \in A} v_j \qquad \text{agent 2 thinks } B \text{ is more valuable.}$$

Prove that deciding if an envy-free allocation exists is NP-complete.

(i) Given an undirected graph $G = (V, E)$, the *3-coloring* problem asks if there is a way to color all the vertices by using three colors, say, red, blue and green, such that every two adjacent vertices have different colors. Prove that 3-coloring is NP-complete.

(j) Given a ground set $U = \{1, \ldots, n\}$ and a collection of its subsets $\mathcal{S} = \{S_1, \ldots, S_m\}$, the *exact cover* problem asks if we can find a subcollection $\mathcal{T} \subseteq \mathcal{S}$ such that $\bigcup_{S \in \mathcal{T}} S = U$ and $S_i \cap S_j = \emptyset$ for any $S_i, S_j \in \mathcal{T}$. Prove that exact cover is NP-complete.

I am going to start from the easiest to the hardest.

(c) Difficulty: $*$

The problem is clearly in NP, as the set of vertices in G that form the subgraph H is a certificate.

To show it is NP-complete, we reduce it from Clique. Given a Clique instance $(G = (V, E), k)$, the instance (G', H') of this problem is constructed as $G' = G$ and H' is a complete graph with k vertices. It is straightforward to check that $(G = (V, E), k)$ is a yes instance if and only if (G', H') is a yes instance.

(d) Difficulty: $*$

The problem is clearly in NP, as the set of edges forming the spanning tree with maximum degree at most k is a certificate.

To show it is NP-complete, we reduce it from HamiltonianPath. Given a HamiltonianPath instance $G = (V, E)$, the instance (G', k) of this problem is constructed as $G' = G$ and $k = 2$.

If $G = (V, E)$ is a yes HamiltonianPath, there is a Hamiltonian path in G, and this is also a spanning tree with maximum degree 2, so (G', k) is a yes instance.

If (G', k) is a yes instance, then a spanning tree with maximum degree 2 must also be a Hamiltonian path, so $G = (V, E)$ is a yes HamiltonianPath instance.

(e) Difficulty: $*$

The problem is clearly in NP, as the collection $\mathcal{T}$ can be served as a certificate, and verifying whether $\mathcal{T}$ covers all the elements in U can be clearly done in polynomial time.

To show it is NP-complete, we reduce it from VertexCover. Given a VertexCover instance $(G = (V, E), k)$, we construct a SetCover instance $(U, \mathcal{S}, k')$ as follows. The ground set U contains $|E|$ elements corresponding to the $|E|$ edges in G; $\mathcal{S}$ contains $|V|$ elements corresponding to $|V|$ vertices in G; $k' = k$. In addition, for an edge $e = (u, v)$ in the VertexCover instance, let the element in U representing e be an element of both the subset in $\mathcal{S}$ representing u and the subset representing v. This complete the description of the construction.

It then follows immediately from our construction that G has a vertex cover of size k if and only if we can find a subcollection $\mathcal{T} \subseteq \mathcal{S}$ with $|\mathcal{T}| = k' = k$ that covers U.

(f) Difficulty: $*$

The problem is clearly in NP, as the subcollection of integers with sum 0 is a certificate.

To show it is NP-complete, we reduce it from SubsetSum+. Given a SubsetSum+ instance (S, k), the instance S' of this problem is constructed as $S' = S \cup \{-k\}$.

If (S, k) is a yes SubsetSum+ instance, there exists a subcollection T of S with sum k. Then adding $-k$ to T gives a subcollection of S' with sum 0, which means S' is also a yes instance.

If S' is a yes instance, the subcollection T' with sum 0 must contain $-k$, as $-k$ is the only negative number in S'. Then excluding $-k$ from T' gives a subcollection of S with sum k, which means (S, k) is a yes SubsetSum+ instance.

(h) Difficulty: $**$

This problem is clearly in NP, as the partition (A, B) is a certificate, and envy-freeness can clearly be verified in polynomial time.

To show it is NP-complete, we reduce it from Partition+. Given a Partition+ instance $S = \{a_1, \ldots, a_n\}$, we construct an instance of this problem with $u_i = v_i = a_i$ for $i = 1, \ldots, n$.

It is then easy to verify that envy-freeness can be rewritten as simply

$$\sum_{i \in A} a_i = \sum_{j \in B} a_j.$$

As a result, the Partition+ instance is a yes instance if and only if there exists an envy-free allocation.

(b) Difficulty: $**$

KNAPSACK is clearly in NP, as the subset of items satisfying the requirements is a certificate.

To show it is NP-complete, we reduce it from SUBSETSUM+. Given a SUBSETSUM+ instance $(S = \{a_1, \ldots, a_n\}, k)$, we construct a KNAPSACK instance such that $w_i = v_i = a_i$ for each $i = 1, \ldots, n$ and $C = V = k$.

If the SUBSETSUM+ instance is a yes instance, there exists $T \subseteq S$ such that the numbers in T sum up to k. Choosing the subset of items corresponding to T is a valid KNAPSACK solution, as the total weight and the total value are both exactly $C = V = k$. Thus, the KNAPSACK instance is a yes instance.

If the KNAPSACK instance is a yes instance, there exists a subset of items T such that the total value is at least $V = k$ and the total weight is at most $C = k$. Since $a_i = v_i = w_i$, we have

$$\sum_{i \in T} v_i = \sum_{i \in T} a_i \geq k$$

and

$$\sum_{i \in T} w_i = \sum_{i \in T} a_i \leq k,$$

which implies

$$\sum_{i \in T} a_i = k.$$

Thus, the SUBSETSUM+ instance is a yes instance.

(a) Difficulty: $**$

The problem is clearly in NP, as the set of $n/2$ vertices that form a clique is a certificate.

To show that the problem is NP-complete, we reduce it from CLIQUE. Given a CLIQUE instance $(G = (V, E), k)$, we construct the following half-clique instance G':

- If $k < \frac{|V|}{2}$, G' is obtained by adding $|V| - 2k$ extra vertices, connect those extra vertices to each other, and then connect each extra vertex to all vertices in V.
- If $k = \frac{|V|}{2}$, $G' = G$.
- If $k > \frac{|V|}{2}$, G' is obtained by adding $2k - |V|$ extra *isolated* vertices.

It is straightforward to check that G has a k-clique if and only if G' has a $\frac{n}{2}$-clique. The details are omitted here.

(g) Difficulty: $**$

The problem is clearly in NP, as the set of vertices whose colors are initially changed to black is a certificate.

To show it is NP-complete, we reduce it from VertexCover. Here is an important observation. Let S be the subset of vertices whose colors are changed to black initially. We will prove that all vertices will eventually turn to black *if and only if* S is a vertex cover of this graph.

If S is a vertex cover, then for each white vertex u, all its neighbors must be black, or in other words, in S. Otherwise, if a neighbor v of u is not in S, then (u, v) is not covered by S and S cannot be a vertex cover. Given that all the neighbors for each white vertex are black, each white vertex will become black in the next update.

If S is not a vertex cover, then there exists an edge (u, v) that is not covered by S. This means u and v are both white. By our rule of update, u and v can never become black: it is never possible that all neighbors of u are black, nor it is possible for v.

Therefore, the reduction is simple. For the VertexCover instance (G, k), the instance for this problem is also (G, k). The remaining details are omitted.

(i) Difficulty: $* * *$

[Solution 1: Textbook Solution]

3-coloring is clearly in NP, as a color assignment of all vertices is a certificate for a yes instance. To show it is NP-complete, we reduce it from 3-SAT.

Given a 3-SAT instance ϕ, we construct a 3-coloring instance $G = (V, E)$ as follows. We will name the three colors T, F, N which stand for "true", "false", "neutral". The reason for this naming will be apparent soon. Firstly, we construct three vertices named t, f, n and three edges $\{t, f\}, \{f, n\}, \{n, t\}$. It is easy to see these 3 vertices, forming a triangle, must be assigned different colors. Without loss of generality, we assume $c(t) = T, c(f) = F, c(n) = N$, where $c : V \to \{T, F, N\}$ is the color assignment function. We call this triangle the "palette": in the remaining part of the graph, whenever we want to enforce that a vertex cannot be assigned a particular color, we connect it to one of the three vertices in the palette.

For each variable x_i in ϕ, we construct two vertices $x_i, \neg x_i$, and three edges $(x_i, \neg x_i)$, (x_i, n), $(\neg x_i, n)$. This ensures that it must be either $c(x_i) = T, c(\neg x_i) = F$ or $c(x_i) = F, c(\neg x_i) = T$. The former case corresponds to assigning `true` to variable x_i, and the latter case corresponds to assigning `false` to variable x_i.

After we have constructed the gadgets simulating the Boolean assignment for variables, we need to create a gadget to simulate each clause m. We use $m = (x_i \vee \neg x_j \vee x_k)$ as an example to illustrate the reduction. Firstly, we create a vertex m and connect it to the two vertices n, f, so that we can only have $c(m) = T$ (i.e., the clause must be evaluated to `true`). Then, we need to create a gadget that connects the three vertices x_i, $\neg x_j$ and x_k (constructed in the previous step) as "inputs", and connects vertex m at the other end as "output". The gadget must simulate the logical OR operation.

The gadget shown on the left-hand side in Fig. 1 simulates the logical OR operation for two inputs: if both two input vertices are assigned F, then the output vertex cannot be assigned T, for otherwise there is no valid way to assign colors for vertices A and B; if at least one input vertex is assigned T, say, Input 1 is assigned T, then it is possible to assign the output vertex T, since it is always valid to assign F to A and N to B. The gadget for the clause $m = (x_i \vee \neg x_j \vee x_k)$ can then be constructed by applying two OR gadgets, shown on the right-hand side of Fig. 1. We do this for all the clauses.

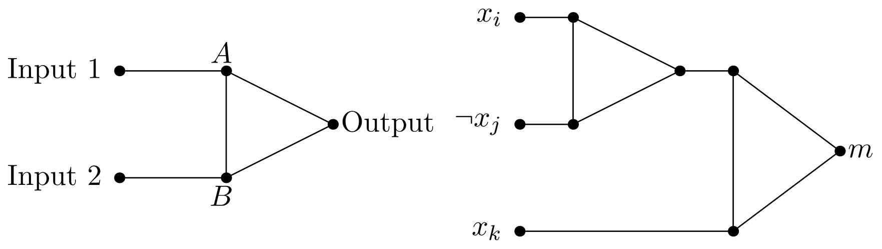

Figure 1: The OR gadget (left-hand side) and the gadget for the clause $m = (x_i \vee \bar{x}_j \vee x_k)$ (right-hand side).

The remaining part of the proof is straightforward. We have shown that G simulate assignments to ϕ. Thus, ϕ is satisfiable if and only if there is a valid color assignment to vertices in G. Finally, it is easy to verify that the construction of G can be done in a polynomial time.

[Solution 2: Wenqian Wang's Solution]

The proof for 3-coloring is in NP is the same as before. To show it is NP-complete, we introduce an intermediate problem NotAllEqual-3SAT.

Problem 1 (NotAllEqual-3SAT). Given a 3-CNF Boolean formula ϕ, decide if there is an assignment such that each clause contains at least one true literal and one false literal.

Lemma 2. NotAllEqual-3SAT *is* NP*-complete.*

Proof. The problem is clearly in NP, and we will present a reduction from 3SAT to NotAllEqual-3SAT. Given a 3SAT instance ϕ, we construct a NotAllEqual-3SAT instance ϕ' as follows. Firstly, introduce a new Boolean variable w. Then, for each clause $C_j = (\ell_1 \vee \ell_2 \vee \ell_3)$ in ϕ, we introduce two more variables y_j, z_j and use three clauses in ϕ' to represent C_j:

$$(w \vee \ell_1 \vee y_j) \wedge (\neg y_j \vee \ell_2 \vee z_j) \wedge (\neg z_j \vee \ell_3 \vee w). \tag{1}$$

Notice that the variable w is used for all triples of three clauses, while y_j, z_j are only used for the j-th triple.

If ϕ is a yes 3SAT instance, we will show that ϕ' is also a yes instance by showing each of the three clauses in (1) contains a `true` and a `false`. We will set $w =$ `false`. Since ϕ is a yes 3SAT instance, we know at least one of ℓ_1, ℓ_2, ℓ_3 is `true`. If $\ell_1 =$ `true`, we can set $y_j =$ `false` and $z_j =$ `false`. If $\ell_2 =$ `true`, we can set $y_j =$ `true` and $z_j =$ `false`. If $\ell_3 =$ `true`, we can set $y_j =$ `true` and $z_j =$ `true`. In each scenario, it can be check that each of the three clauses in (1) contains a `true` and a `false`. Thus, ϕ' is a yes NotAllEqual-3SAT instance.

If ϕ' is a yes NotAllEqual-3SAT instance, we aim to show that at least one of ℓ_1, ℓ_2, ℓ_3 must be true, which will imply ϕ is a yes 3SAT instance. Firstly, we assume without loss of generality that $w =$ `false`. If $w =$ `true`, we can flip the values for all variables, which will also be a valid assignment. Suppose for the sake of contradiction that $\ell_1 = \ell_2 = \ell_3 =$ `false`. To ensure the first and the third clauses in (1) contain at least one `true`, we need to set $y_j =$ `true` and $z_j =$ `false`. In this case, the second clause does not contain any `true`, which is a contradiction. $\square$

Lemma 3. *NotAllEqual-3SAT $\leq_k$ 3-coloring.*

Proof. The main idea here is that a not-all-equal gadget is much easier to construct. The figure on the left-hand side of Fig. 2 demonstrates a not-all-equal gadget.

It is easy to check that, if all the three inputs have the same value, or the same color, the three middle vertices v_1, v_2, v_3 can only choose from two colors, which is impossible. On the other hand, if the three inputs are not all equal, then we can properly choose colors for v_1, v_2, v_3. For example, if Input 1 is T and Input 2 is F, regardless of the value of Input 3, assigning $c(v_1) = F, c(v_2) = T, c(v_3) = N$ is always valid. Thus, this gadget faithfully performs the not-all-equal job.

Owing to the simplicity of the not-all-equal gadget, we do not need a triangle as a palette as before. All we need is a vertex n that is assumed (without loss of generality) to be colored with N. We connect n to the vertex representing x_i and the vertex representing $\neg x_i$. The figure on the right-hand side of Fig. 2 demonstrates these features.

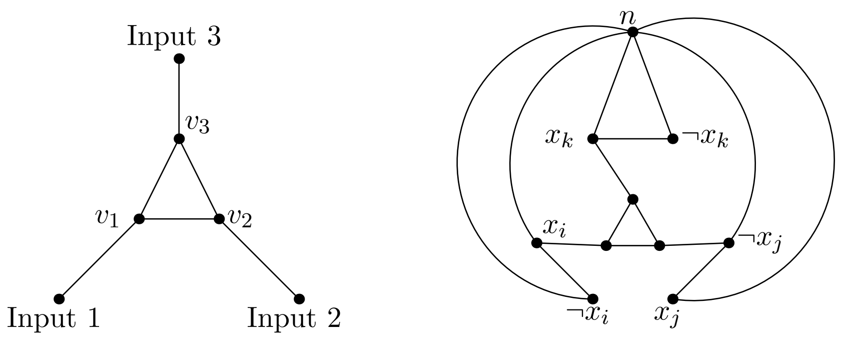

Figure 2: The not-all-equal gadget (left-hand side); the gadget for the clause $m = (x_i \vee \bar{x}_j \vee x_k)$ and how it is connected with other vertices (right-hand side).

The remaining parts of the proof are straightforward, and thus omitted. □

Lemma 2 and Lemma 3 immediately imply 3-coloring is NP-complete.

(j) Difficulty: $* * *$

[Solution 1: Biaoshuai Tao's Solution]

The problem is clearly in NP, as a subcollection of subsets that exactly covers U is a certificate.

To show it is NP-complete, we reduce it from 3-coloring. Given a 3-coloring instance $G = (V, E)$, we construct an exact cover instance as follows.

The ground set U is constructed as follows.

- For each vertex $u \in V$, we construct an element $e_u \in U$.
- For each edge $(u, v) \in E$, we construct three elements $e^r_{uv}, e^g_{uv}, e^b_{uv} \in U$, where the superscripts r, g, b stand for red, green and blue respectively.

The collection $\mathcal{S}$ is constructed as follows:

- For each vertex $u \in V$, we construct three subsets $S^r_u, S^g_u, S^b_u \in \mathcal{S}$ defined as follows.
  - include $e_u \in S^r_u$
  - include $e^r_{uv} \in S^r_u$ for each u's neighbor v.
  - S^g_u and S^b_u are defined similarly.
- For each edge $(u, v) \in E$, construct three subsets $S^r_{uv}, S^g_{uv}, S^b_{uv} \in \mathcal{S}$ such that they contains $e^r_{uv}, e^g_{uv}, e^b_{uv}$ respectively. Notice that each of $S^r_{uv}, S^g_{uv}, S^b_{uv}$ contains only one elements.

For a demonstrating example, suppose the 3-coloring instance is just a triangle with three vertices u, v, w. We have $U = \{e_u, e_v, e_w, e^r_{uv}, e^g_{uv}, e^b_{uv}, e^r_{vw}, e^g_{vw}, e^b_{vw}, e^r_{wu}, e^g_{wu}, e^b_{wu}\}$. The collection $\mathcal{S}$ contains 18 subsets:

$$S^r_u = \{e_u, e^r_{uv}, e^r_{wu}\}, S^g_u = \{e_u, e^g_{uv}, e^g_{wu}\}, S^b_u = \{e_u, e^b_{uv}, e^b_{wu}\}$$

$$S^r_v = \{e_v, e^r_{uv}, e^r_{vw}\}, S^g_v = \{e_v, e^g_{uv}, e^g_{vw}\}, S^b_v = \{e_v, e^b_{uv}, e^b_{vw}\}$$

$$S^r_w = \{e_w, e^r_{vw}, e^r_{wu}\}, S^g_w = \{e_w, e^g_{vw}, e^g_{wu}\}, S^b_w = \{e_w, e^b_{vw}, e^b_{wu}\}$$

$$S^r_{uv} = \{e^r_{uv}\}, S^g_{uv} = \{e^g_{uv}\}, S^b_{uv} = \{e^b_{uv}\}, S^r_{vw} = \{e^r_{vw}\}, S^g_{vw} = \{e^g_{vw}\}, S^b_{vw} = \{e^b_{vw}\}, S^r_{wu} = \{e^r_{wu}\},$$

$$S^g_{wu} = \{e^g_{wu}\}, S^b_{wu} = \{e^b_{wu}\}.$$

This whole construction can clearly be done in polynomial time.

If the 3-coloring instance is a yes instance, let $c : V \to \{r, g, b\}$ be a valid coloring. We further assign a color to an edge (u, v), denoted by $c(u, v)$, such that $c(u, v)$ is different from $c(u)$ and $c(v)$. Given a valid coloring of vertices, each edge is uniquely colored. For example, if $c(u) = r$ and $c(v) = b$, then we have $c(u, v) = g$. To show the exact cover instance we constructed is a yes instance, we consider the following subcollection $\mathcal{T} \subseteq \mathcal{S}$:

- For each vertex u, include $S^{c(u)}_u$ to $\mathcal{T}$;
- For each edge (u, v), include $S^{c(u,v)}_{uv}$ to $\mathcal{T}$.

We will prove that $\mathcal{T}$ is an exact cover.

For each element e_u corresponding to vertex u in G, it is covered by exactly one of $S^{c(u)}_u$. For each element e^r_{uv} corresponding to edge (u, v) in G, it is covered by S^r_u if $c(u) = r$, it is covered by S^r_v if $c(v) = r$, and it is covered by S^r_{uv} if $c(u, v) = r$. Since exactly one of $c(u) = r, c(v) = r, c(u, v) = r$ can happen, e^r_{uv} is covered by exactly one subset. The same holds for e^g_{uv} and e^b_{uv}. We conclude that the exact cover instance we constructed is a yes instance.

Now, suppose the exact cover instance is a yes instance. We will show that the 3-coloring instance is a yes instance. Let $\mathcal{T}$ be a valid solution to the exact cover instance. Then exactly one of S^r_u, S^g_u, S^b_u is included in $\mathcal{T}$, as these are the three subsets that contains e_u. This corresponds to a coloring $c : V \to \{r, g, b\}$: if S^r_u is included, we assign $c(u) = r$; if S^g_u is included, we assign $c(u) = g$; if S^b_u is included, we assign $c(u) = b$. It remains to show that c is a valid 3-coloring. Suppose for the sake of contradiction that there exists $(u, v) \in E$ with $c(u) = c(v)$. Assume $c(u) = c(v) = r$ without loss of generality. Then we have included both S^r_u and S^r_v in $\mathcal{T}$. In this case, the element e^r_{uv} is covered by both S^r_u and S^r_v, which contradicts to that $\mathcal{T}$ is an exact cover.

[Solution 2: Jiaxin Song's Solution]

The proof that exact cover is in NP is the same as before.

To show it is NP-complete, we reduce it from 3SAT. Given a 3SAT instance ϕ, we construct an exact cover instance as follows.

The ground set U is constructed as follows:

- For each variable x_i, we construct an element $e_{x_i} \in U$.
- For each clause $C_j = (\ell_1 \vee \ell_2 \vee \ell_3)$, we construct four elements $e_{C_j}, e_{C_j}^{\ell_1}, e_{C_j}^{\ell_2}, e_{C_j}^{\ell_3} \in U$.

The collection $\mathcal{S}$ is constructed as follows:

- For each variable x_i, construct two subsets $S_{x_i}^T, S_{x_i}^F \in \mathcal{S}$ where
  - $S_{x_i}^T = \{e_{x_i}\} \cup \{e_{C_j}^{x_i} : x_i \text{ is a literal in } C_j\}$;
  - $S_{x_i}^F = \{e_{x_i}\} \cup \{e_{C_j}^{\neg x_i} : \neg x_i \text{ is a literal in } C_j\}$.
- For each clause $C_j = (\ell_1 \vee \ell_2 \vee \ell_3)$, construct seven subsets $S_{C_j}^{TTT}, S_{C_j}^{TTF}, S_{C_j}^{TFT}, S_{C_j}^{FTT}, S_{C_j}^{FFT}, S_{C_j}^{FTF}, S_{C_j}^{TFF} \in \mathcal{S}$ where
  - $S_{C_j}^{TTT} = \{e_{C_j}\}$
  - $S_{C_j}^{TTF} = \{e_{C_j}, e_{C_j}^{\ell_3}\}$
  - $S_{C_j}^{TFT} = \{e_{C_j}, e_{C_j}^{\ell_2}\}$
  - $S_{C_j}^{FTT} = \{e_{C_j}, e_{C_j}^{\ell_1}\}$
  - $S_{C_j}^{FFT} = \{e_{C_j}, e_{C_j}^{\ell_1}, e_{C_j}^{\ell_2}\}$
  - $S_{C_j}^{FTF} = \{e_{C_j}, e_{C_j}^{\ell_1}, e_{C_j}^{\ell_3}\}$
  - $S_{C_j}^{TFF} = \{e_{C_j}, e_{C_j}^{\ell_2}, e_{C_j}^{\ell_3}\}$

For a demonstrating example, suppose ϕ consist of only one clause $C_1 = (x_1 \vee \neg x_2 \vee x_3)$. The ground set is $U = \{e_{x_1}, e_{x_2}, e_{x_3}, e_{C_1}, e_{C_1}^{x_1}, e_{C_1}^{\neg x_2}, e_{C_1}^{x_3}\}$. The collection $\mathcal{S}$ consists of the following 13 subsets:

$$S_{x_1}^T = \{e_{x_1}, e_{C_1}^{x_1}\}, S_{x_1}^F = \{e_{x_1}\}, S_{x_2}^T = \{e_{x_2}\}, S_{x_2}^F = \{e_{x_2}, e_{C_1}^{\neg x_2}\}, S_{x_3}^T = \{e_{x_3}, e_{C_3}^{x_3}\}, S_{x_3}^F = \{e_{x_3}\}$$

$$S_{C_1}^{TTT} = \{e_{C_1}\}, S_{C_1}^{TTF} = \{e_{C_1}, e_{C_1}^{x_3}\}, S_{C_1}^{TFT} = \{e_{C_1}, e_{C_1}^{\neg x_2}\}, S_{C_1}^{FTT} = \{e_{C_1}, e_{C_1}^{x_1}\}$$

$$S_{C_1}^{FFT} = \{e_{C_1}, e_{C_1}^{x_1}, e_{C_1}^{\neg x_2}\}, S_{C_1}^{FTF} = \{e_{C_1}, e_{C_1}^{x_1}, e_{C_1}^{x_3}\}, S_{C_1}^{TFF} = \{e_{C_1}, e_{C_1}^{\neg x_2}, e_{C_1}^{x_3}\}$$

The whole construction can clearly be done in polynomial time.

Suppose ϕ is a yes 3SAT instance. We will construct an exact cover $\mathcal{T}$ to show that the exact cover instance is a yes instance. For each x_i, if $x_i = \texttt{true}$, we include $S_{x_i}^T \in \mathcal{T}$; otherwise, include $S_{x_i}^F \in \mathcal{T}$. For each clause C_j, we check the values of the three literals, and include the corresponding subset $S_{C_j}^{XXX}$ in $\mathcal{T}$. For example, for the clause

$C_1 = (x_1 \vee \neg x_2 \vee x_3)$, if the assignment is $x_1 = x_2 = x_3 = \texttt{true}$, the first and the third literals are `true` and the second is `false`, and in this case we will include $S_{C_1}^{TFT}$. We will show $\mathcal{T}$ is an exact cover.

Firstly, each "variable element" e_{x_i} is covered exactly once by either $S_{x_i}^T$ or $S_{x_i}^F$. Secondly, each "clause element" e_{C_j} is covered exactly once since we have selected exactly one of those seven $S_{C_j}^{XXX}$. Lastly, for each element $e_{C_j}^{x_i}$, it is covered exactly once: if we have not selected $S_{x_i}^T$, based on our construction of $\mathcal{T}$, it will be covered by the subset $S_{C_j}^{XXX}$ we selected; if we have selected $S_{x_i}^T$, it will not be covered by the subset $S_{C_j}^{XXX}$ we selected. The same analysis holds for each element $e_{C_j}^{\neg x_i}$: it will be covered by either $S_{x_i}^F$ or the subset $S_{C_j}^{XXX}$ we selected, but not both. We conclude that $\mathcal{T}$ is an exact cover.

Now, suppose we have an exact cover $\mathcal{T}$. We will show that ϕ is satisfiable. For each variable x_i, exactly one of $S_{x_i}^T$ and $S_{x_i}^F$ must be selected to cover e_{x_i}. This gives us a natural assignment to $x_1, \ldots, x_n$. We will show that this is a satisfiable assignment. Suppose for the sake of contradiction that there is a clause C_j where the values of all the three literals are `false`. Using the same example $C_1 = (x_1 \vee \neg x_2 \vee x_3)$. Suppose $x_1 = x_3 = \texttt{false}$ and $x_2 = \texttt{true}$. Then we have selected $S_{x_1}^F, S_{x_2}^T, S_{x_3}^F$. The three elements $e_{C_1}^{x_1}, e_{C_1}^{\neg x_2}, e_{C_1}^{x_3}$ are not covered by any of $S_{x_1}^F, S_{x_2}^T, S_{x_3}^F$, so they need to be covered by those $S_{C_1}^{XXX}$. We can only select one of those $S_{C_1}^{XXX}$: if we select more than one of those, the element e_{C_1} will be covered more than once. On the other hand, only one $S_{C_1}^{XXX}$ cannot cover all the three elements $e_{C_1}^{x_1}, e_{C_1}^{\neg x_2}, e_{C_1}^{x_3}$ by our construction (we have not included "$S_{C_1}^{FFF}$" as a subset in $\mathcal{S}$). Therefore, $\mathcal{T}$ cannot be an exact cover, which leads to a contradiction.