Problem 1 (18 points).

Indicate *true* or *false* for each of the following statements. (You do not need to describe middle steps.)

1. Let G be a network and e be an edge of G. If increasing the capacity of e by 2 results in the increase of the value of the maximum flow by 1, then e must not be in *any* minimum cut of G.
2. If P = NP, then we must have NP = co-NP. *(If you think it remains open, indicate "false".)*
3. Assume that in a random experiment the probability of succeeding is p. Then the expected number of independent repeat of such experiments until succeeding k times is k/p.
4. For any $k \geq 2$, any instance of k-SAT (i.e., each clause has exactly k distinct literals) consisting of exactly $2 \cdot k + 1$ clauses must be satisfiable (i.e., there exists an assignment for each variable such that all clauses are true).
5. Let $G = (V, E)$ be an arbitrary undirected graph. Let $V_1 \subseteq V$ be one minimum vertex cover of G. Let $V_2 \subseteq V$ be one maximum independent set of G. Then we must have $|V_1| + |V_2| = |V|$.
6. If we assume that in the Knapsack problem the *weight* of every item is at most n^2, where n is the number of items, then this problem can be solved in polynomial-time.

Solution.

1. False. Counter-example: $G = (V, E)$, $V = \{s, a, t\}$, edges include (s, a) with capacity of 2, and (a, t) with capacity of 1. The minimum cut of G consists of one cut-edge (a, t). If $e = (a, t)$, then increasing its capacity by 2 results in the increase of the value of the maximum flow of G by 1. But e is in one minimum cut of G.
2. True.
3. True. We proved that, the expected number of independent repeat of such experiments until succeeding once is $1/p$. Succeeding k times can be regarded as k continuous succeeding events, and hence the expected total such experiments is k/p.
4. False. For general k, any k-SAT instance with at most $2^k - 1$ clauses must be satisfiable. When $k = 2$, not every instance with 5 clauses is satisfiable.
5. True. This is because V_1 is a vertex cover if and only if $V \setminus V_1$ is an independent set.
6. True. We have an optimal DP algorithm runs in $O(nW)$ time. We can design the following algorithm for this problem: if $W \geq n^3$, return all n items; otherwise, run this DP algorithm. The overall running time is $O(n^4)$.

Problem 2 (24 points).

A company has n branches $\{X_1, X_2, \cdots, X_n\}$ in one city, and it plans to move some of them to another city. Branch X_i will cost a_i for renting per year if it stays in the original city, and will cost b_i for renting per year if X_i is moved to the new city. The traveling cost will be c_{ij} per year between X_i and X_j if they are in the same city (no matter the original or the new one), and will cost d_{ij} per year if X_i and X_j are not in the same city. It is guaranteed that $c_{ij} \leq d_{ij}$ for all $1 \leq i \neq j \leq n$. Given positive costs a_i, b_i, c_{ij}, and d_{ij}, $1 \leq i \neq j \leq n$, design a polynomial-time algorithm to decide which branches should be moved to the new city such that the total

cost (renting for all branches plus traveling for all pairs of branches) per year is minimized. (*Hint:* consider reducing this problem to a network flow problem.)

Solution. We build an undirected network $G = (V,E)$, where $V = \{X_1, X_2, \cdots, X_n\} \cup \{s,t\}$. We add the following edges to the network. For every branch X_i, we add undirected edge (s, X_i) with capacity of b_i. For every branch X_i, we add undirected edge (X_i, t) with capacity of a_i. For every pair of branches (X_i, X_j), we add undirected edge (X_i, X_j) with capacity of $d_{ij} - c_{ij}$. (Note that we can equivalently transform this undirected network to a directed network by transforming every undirected edge (u,v) to two directed edges (u,v) and (v,u) with the same capacity.)

We run any max-flow algorithm to get the maximum-flow and minimum *s*-*t* cut of G. Let (S^*, T^*) be the minimum *s*-*t* cut of G. We return $T^* \setminus \{t\}$, i.e., move the corresponding branches in $T^* \setminus \{t\}$ to another city.

We now prove that the above algorithm is optimal. Let (S,T) be an arbitrary *s*-*t* cut of G. Consider the capacity of (S,T).

$$\begin{aligned}
c(S,T) &= \sum_{X_i \in S} a_i + \sum_{X_j \in T} b_j + \sum_{X_i \in S} \sum_{X_j \in T} (d_{ij} - c_{ij}) \\
&= \sum_{X_i \in S} a_i + \sum_{X_j \in T} b_j + \sum_{X_i \in S} \sum_{X_j \in T} d_{ij} - \sum_{X_i \in S} \sum_{X_j \in T} c_{ij} \\
&= \sum_{X_i \in S} a_i + \sum_{X_j \in T} b_j + \sum_{X_i \in S} \sum_{X_j \in T} d_{ij} - \left(\sum_{1 \le i \ne j \le n} c_{ij} - \sum_{X_i \ne X_j \in S} c_{ij} - \sum_{X_i \ne X_j \in T} c_{ij} \right) \\
&= \sum_{X_i \in S} a_i + \sum_{X_j \in T} b_j + \sum_{X_i \in S} \sum_{X_j \in T} d_{ij} + \sum_{X_i \ne X_j \in S} c_{ij} + \sum_{X_i \ne X_j \in T} c_{ij} - \sum_{1 \le i \ne j \le n} c_{ij}
\end{aligned}$$

Define $f(S,T) = \sum_{X_i \in S} a_i + \sum_{X_j \in T} b_j + \sum_{X_i \in S} \sum_{X_j \in T} d_{ij}$. We then have

$$c(S,T) = f(S,T) - \sum_{1 \le i \ne j \le n} c_{ij}$$

Notice that $f(S,T)$ is exactly the total cost of moving branches in $T \setminus \{t\}$ to another city. Since $\sum_{1 \le i \ne j \le n} c_{ij}$ is a constant (not to do with S or T), we have that the capacity of minimum *s*-*t* cut gives the minimized total costs. This prove that $T^* \setminus \{t\}$ is the optimal solution.

Problem 3 (26 points).

Let $G = (V,E)$ be an undirected graph. We say two vertices $u, v \in V$ *conflict* with each other, if $(u,v) \in E$, or there exists another vertex $w \in V$ such that $(u,w) \in E$ and $(v,w) \in E$. Given $G = (V,E)$, the problem is to find $V_1 \subseteq V$ such that no pair of vertices in V_1 conflict and that $|V_1|$ is maximized.

1. **(13 points.)** Prove that this problem is polynomial-time reducible to the maximum-independent-set problem, that is, design an algorithm for this problem in which you can use (up to polynomial times) a solver for the maximum-independent-set problem, and prove its correctness.

2. **(13 points.)** Prove that the maximum-independent-set problem is polynomial-time reducible to this problem, that is, design an algorithm for the maximum-independent-set problem in which you can use (up to polynomial times) a solver for this problem, and prove its correctness.

Solution.

1. Let $G = (V, E)$ be any instance of this problem. Now we construct an instance $G' = (V', E')$ of the maximum-independent-set problem. We set $V' = V$. For any pair of vertices $u, v \in V$, we consider two cases: if we have $(u, v) \in E$, then we add (u, v) to E'; if not, we enumerate all other vertices $w \in V \setminus \{u, v\}$ and examine whether $(u, w) \in E$ and $(v, w) \in E$, and if such w exists we add (u, v) to E'.

   We use the solver for maximum-independent-set to find the maximum independent-set of G'. Let it be V_1^*. We now show that V_1^* is the optimal solution of the original problem. In fact, the above construction directly implies that, in G two vertices u and v conflict if and only if in G' there exists one edge $(u, v) \in E'$. Hence, in G a subset of vertices $V_1 \subseteq V$ without conflicting if and only if in G' the same subset of vertices V_1 is an independent-set.

2. Let $G = (V, E)$ be an instance of the maximum-independent-set problem. Without loss of generality, we assume that G is connected; otherwise we run the following algorithm on each of its connected componeents. Now we construct an instance $G' = (V', E')$ of the original problem. For first add V to V'. Then for any edge $e \in E$, we add a *middle* vertex v_e to V'. (That is, $|V'| = |V| + |E|$.) For any edge $e = (u, v) \in E$, we add two edges, (u, v_e) and (v_e, v) to E'. (These are equivalent to construct G' from G by adding an extra vertex in the middle of every edge of G.) We eventually add edge (v_{e_1}, v_{e_2}) to G' for every pair of middle vertices v_{e_1} and v_{e_2}.

   We then use the solver for the original problem to get the maximized subset of G' in which no pair of vertices conflict. Let V_1^* be the subset returned by this solver. We first consider the case that $|V_1^*| \geq 2$. We now prove that, if $|V_1^*| \geq 2$, then none of the vertices in V_1^* is middle vertex. Suppose conversely that there exists $v_e \in V_1^*$. Let $u \in V_1^*$ be another vertex. Since we assume that G is connected, u is adjacent to at least one edge; let e_1 be such an edge that is adjacent to u in G. Then in G' there is a middle edge v_{e_1}. According to our construction of G', there exists edge (u, v_{e_1}) in G', and there exists edge (v_e, v_{e_1}) in G'. Hence, u and v_{e_1} conflict in G', a contradiction to the fact that both v_e and u are in V_1^*. Following this claim, we konw that $V_1^* \subseteq V$. We then claim that, for any two vertices $u, v \in V_1^*$, we must have that $(u, v) \notin E$. In fact, if there exists $e = (u, v) \in E$, then according to our construction, there exists (u, v_e) and (v, v_e) in E' and hence u and v conflict in G' and therefore they cannot be both in V_1^*. Following this claim, we know that V_1^* is an independent set of G, i.e., $|V_1^*|$ is a lower bound of the maximum independent set of G. We now prove that, $|V_1^*|$ also gives an upper bound of the maximum independent set of G. This is because, if $(u, v) \notin E$, then u and v do not conflict in G' (based on our construction). In other words, any independent set in G is also a subset of G' without conflicting. Hence, $|V_1^*|$ is also an upper bound of the optimal solution of G. Combined, we have that V_1^* is one maximum independent-set of G.

   The second case is that $|V_1^*| = 1$. In this case, based on the last claim we proved above, we know that the maximum independent set of G contains only 1 vertex. Hence, in this case, the algorithm simply return an arbitrary vertex of G.

Problem 4 (32 points).

You are given an undirected graph $G = (V, E)$, and you need to color each vertex with one of the given four colors. We say an edge $(u, v) \in E$ is *satisfied* if u and v are assigned different colors. Given $G = (V, E)$, the problem is to color all vertices so that the number of satisfied edges is maximized.

1. **(16 points)** Design a randomized $4/3$-approximation algorithm for this problem, that is, the *expected* number of satisfied edges returned by your algorithm should be at least $3/4$ fraction of the number of the satisfied edges in the optimal solution. Prove your algorithm can achieve such randomized approximation ratio. Your algorithm should run in polynomial-time.

2. **(16 points)** Design a (deterministic) $4/3$-approximation algorithm for this problem, that is, the number of satisfied edges returned by your algorithm should be at least $3/4$ fraction of the number of the satisfied edges in the optimal solution. Prove your algorithm can achieve such approximation ratio. Your algorithm should run in polynomial-time.

Solution.

1. The algorithm independently assigns one of the 4 colors with probability of 1/4 for each vertex. We now show its expected performance. Let Z be the random variable indicates the total number of satisfied edges. Let Z_e be the binary random variable indicates whether edge $e \in E$ is satisfied. We have that $Z = \sum_{e \in E} Z_e$ and therefore $\mathbb{E}(Z) = \sum_{e \in E} \mathbb{E}(Z_e)$. Further, edge $e = (u, v)$ is satisfied if and only if u and v are colored differently, and with probability of $4 \cdot 1/4 \cdot 1/4$, they are colored the same. Hence, $\Pr(Z_e = 1) = 1 - \Pr(Z_e = 0) = 1 - 1/4 = 3/4$. Combined, $\mathbb{E}(Z) = \sum_{e \in E} \mathbb{E}(Z_e) = 3 \cdot |E|/4$. As the optimal solution satisfies at most $|E|$ edges, this algorithm is a randomized $4/3$-approximation algorithm.

2. We can use the conditional expectation technique to derandomize above randomized algorithm to get a deterministic approximation algorithm with the same approximation ratio. Let $\{c_1, c_2, c_3, c_4\}$ be the four given colors. Number all vertices as $V = \{v_1, v_2, \cdots, v_n\}$. The derandomization algorithm is given below.

   Algorithm DERAND
   for $i = 1$ to n
   compute $A_1 = \mathbb{E}(Z \mid v_1 = v_1^*, v_2 = v_2^*, \cdots, v_{i-1} = v_{i-1}^*, v_i = c_1)$
   compute $A_2 = \mathbb{E}(Z \mid v_1 = v_1^*, v_2 = v_2^*, \cdots, v_{i-1} = v_{i-1}^*, v_i = c_2)$
   compute $A_3 = \mathbb{E}(Z \mid v_1 = v_1^*, v_2 = v_2^*, \cdots, v_{i-1} = v_{i-1}^*, v_i = c_3)$
   compute $A_4 = \mathbb{E}(Z \mid v_1 = v_1^*, v_2 = v_2^*, \cdots, v_{i-1} = v_{i-1}^*, v_i = c_4)$
   among the above 4 values find the largest one A_k and set $v_i^* = c_k$
   end for
   return $\{v_1^*, v_2^*, \cdots, v_n^*\}$
   end DERAND

   To compute the conditional expectation, for example $A_1 = \mathbb{E}(X \mid v_1 = v_1^*, v_2 = v_2^*, \cdots, v_{i-1} = v_{i-1}^*, v_i = c_1)$, we use $A_1 = \sum_{e \in E} \mathbb{E}(Z_e \mid v_1 = v_1^*, v_2 = v_2^*, \cdots, v_{i-1} = v_{i-1}^*, v_i = c_1)$. Let $e = (u, v)$. If both u and v are colored, i.e., $u, v \in \{v_1, v_2, \cdots, v_i\}$, then we have $\mathbb{E}(Z_e \mid v_1 = v_1^*, v_2 = v_2^*, \cdots, v_{i-1} = v_{i-1}^*, v_i = c_1) = 1$ if they are colored differently, and equals to 0 if not. If either u or v is not colored yet, then we have that $\mathbb{E}(Z_e \mid v_1 = v_1^*, v_2 = v_2^*, \cdots, v_{i-1} = v_{i-1}^*, v_i = c_1) = 3/4$.

   Using the same argument (in the MAX-3SAT problem), we can prove that this algorithm keeps the invariant that after i-th iteration, $\mathbb{E}(Z \mid v_1 = v_1^*, v_2 = v_2^*, \cdots, v_{i-1} = v_{i-1}^*, v_i = v_i^*) \geq \mathbb{E}(Z) = 3 \cdot |E|/4$. Hence after n iterations (i.e., when the algorithm terminates), all vertices are colored (with $v_1^*, v_2^*, \cdots, v_n^*$), and the number of satisfied edges is at least $3 \cdot |E|/4$.