CS 240 Homework 3

□□□

TOTAL POINTS

6 / 6

QUESTION 1

1 1 / 1

✓ **- 0 pts** Correct

- 0.5 pts Please prove 3-CNF-SAT can be reduced to 4-CNF-SAT

- 0.5 pts Take each clause (□□□□□) and turn it into (□□□□□□□)□(□□□□□□¬□)

- 0.5 pts construction is unclear

- 1 pts 3SAT□□4SAT□□□□□□□□□□□□□□□□□□□

QUESTION 2

2 1 / 1

✓ **- 0 pts** Correct

- 1 pts Click here to replace this description.

- 1 pts This is not a 4-coloring problem, here is to allow conflicts no greater than K.

- 1 pts □□□□□□□□□□□□

- 0.2 pts Make the number of conflicts no more than K? You did not specify the parameters of K as 0.

- 0.1 pts Each edge represents a student?

- 0.1 pts There is a conflict in student in R? □□□□□

- 0.2 pts It is not stated that a student can only take two courses.

- 0.1 pts K--?

- 0.2 pts K coloring problem needs the parameter k = 0 □Refers to the k in problem2□

- 0.2 pts Edges is defined wrongly

- 0.3 pts □□□□problem2□NP□□

- 0.3 pts □□□□□□□□□□

- 0.8 pts Problem2 -> K+k coloring? And 3-coloring -> K coloring? != 3-coloring -> Problem2

- 0.3 pts □□□□□□□□□□□

- 0.5 pts □□□□□□□□□□□

- 0.8 pts □□□□□

- 0.4 pts edges and vertices are defined wrongly

QUESTION 3

3 1 / 1

✓ **- 0 pts** Correct

- 0.3 pts Please show TA cycle is in NP

- 0.7 pts Only show TA cycle is in NP

- 0.2 pts Inaccurate construction

- 0.3 pts mapping function not in polynomial time.

- 1 pts Wrong

- 0.5 pts wrong construction

QUESTION 4

4 1 / 1

✓ **- 0 pts** Correct

- 0.2 pts Without proof that Knapsack is NP

- 0.2 pts Wrong construction of reducing

- 1 pts No answer or totally wrong

- 0.3 pts Only show one direction of equivalence

- 0.1 pts Unclear constuction

- 0.6 pts Not showing the equivalence of answer to two problems

QUESTION 5

5 1 / 1

✓ **+ 1 pts** Correct

- 0.1 pts the value of b is wrong

- 0.1 pts wrong when handling Ax<=b and Ax >=b

+ 0.4 pts wrong, but not totally, and for your hard work

+ 0.2 pts np

+ 0.5 pts hard work

+ 0 pts Wrong, or can not find your solution

+ 0.5 pts construct

+ 0.3 pts proof

- 0.2 pts you can not constraint that x_i = 1-x_j

- 0.1 pts small mistake

QUESTION 6

6 1 / 1

✓ **- 0 pts** Correct

- 0.1 pts "In NP" step is wrong

- 0.1 pts "mention mapping takes polynomial time" step is missing. Or the construction is not from a Independent Set "Instance" to Pairwise Disjoint "instance".

- 0.4 pts "Instance Construction" step is wrong or missing.

- 0.4 pts "Proof Instances Yes/No Equal" step is wrong or missing.

- 1 pts Wrong.

Problem 1

Solution

1. 4-SAT $\in$ NP
   If the clauses Φ with size k is given, and the truth assignment is also given, we can check it by directly evaluate the expression, and if the evaluation answer is True, then it's a true instance, otherwise it's a false instance. It will take $O(k)$ time, which is polynomial, so 4-SAT $\in$ NP.

2. 3-SAT $\leq_p$ 4-SAT

   - Construction
     If an arbitrary 3-SAT problem with size k is given, we write it as $\Phi = \phi_1 \wedge \phi_2 \ldots \wedge \phi_k$, ϕ_1 to ϕ_k are clauses like $x_1 \vee \bar{x}_2 \vee x_3$, we can convert it to a 4-SAT problem by following steps: For each clause ϕ_i, split it two 4-SAT clauses with adding another variable x, for example, $x_1 \vee \bar{x}_2 \vee x_3$, will be transformed to $(x_1 \vee \bar{x}_2 \vee x_3 \vee x) \wedge (x_1 \vee \bar{x}_2 \vee x_3 \vee \bar{x})$. Hence we transformed a 3-SAT problem $\Phi = \phi_1 \wedge \phi_2 \ldots \wedge \phi_k$ to a 4-SAT problem (with size $2k$) $\Phi' = (\phi_1 \wedge x) \wedge (\phi_1 \wedge \bar{x}) \ldots \wedge (\phi_k \wedge x) \wedge (\phi_k \wedge \bar{x})$. It's obviously that if the problem size of 3-SAT is k, then the reduction to 4-SAT will take $O(k)$ time, which is polynomial.
   - Proof
     We need to prove that $x_1, \ldots, x_n$ is a yes instance of Φ' (4-SAT problem) if anc only if $x_1, \ldots, x_n$ is a yes instance of Φ (3-SAT problem).
     - ($\Rightarrow$)
       Give the truth assignment $X = x_1, \ldots, x_n$ of Φ (3-SAT), because $z_1 \vee z_2 \vee z_3 = (z_1 \vee z_2 \vee z_3 \vee x) \wedge (z_1 \vee z_2 \vee z_3 \vee \bar{x})$, it's also the solution of Φ'
     - ($\Leftarrow$)
       Let $X = x_1, \ldots, x_n$ is a truth assignment of Φ'. Notice that for each group we have $(z_1 \vee z_2 \vee z_3 \vee x) \wedge (z_1 \vee z_2 \vee z_3 \vee \bar{x}) = z_1 \vee z_2 \vee z_3$, where $z_1 \vee z_2 \vee z_3$ is the corresponding clause in Φ, so if each group $(z_1 \vee z_2 \vee z_3 \vee x) \wedge (z_1 \vee z_2 \vee z_3 \vee \bar{x})$ is true, then each clause in Φ is also true. Hence, the 3-SAT problem Φ is satisfied.

Problem 2

Solution

1. Course scheduling $\in$ NP
   If an arbitrary solution is given, we can trverse all students' requests and check whether the number of conflicts out of limits in polynomial time and counting the conflicts.

2. 3-Color $\leq_p$ Course scheduling

   - Construction
     If an instance of 3-Color (an undirected graph $G = (V, E)$) is given, we can construct an instance $\{C, S, R, K\}$ of Course scheduling by following steps.
     (a) Let each course $c_1, \ldots, c_k$ corresponds to each vertex $v_1, \ldots, v_k$ in graph(i.e. $C = V$)
     (b) Let $S = \{1, 2, 3\}$, each number represents a color in 3-Color and different time slots in Course scheduling.
     (c) Let $R = \{\{(u, v) = e\}, \forall e \in E\}$, where each edge represents a student, and the endpoints represent the two courses he/she wants to take.
     (d) Finally, let $K = 0$

Problem 2 continued on next page...

1 1 / 1

✓ **- 0 pts** Correct

- 0.5 pts Please prove 3-CNF-SAT can be reduced to 4-CNF-SAT

- 0.5 pts Take each clause (□□□□□)
and turn it into (□□□□□□□)□(□□□□□□¬□)

- 0.5 pts construction is unclear

- 1 pts 3SAT□□4SAT□□□□□□□□□□□□□□□□□□

Problem 1

Solution

1. 4-SAT $\in$ NP
   If the clauses Φ with size k is given, and the truth assignment is also given, we can check it by directly evaluate the expression, and if the evaluation answer is True, then it's a true instance, otherwise it's a false instance. It will take $O(k)$ time, which is polynomial, so 4-SAT $\in$ NP.

2. 3-SAT $\leq_p$ 4-SAT

   - Construction
     If an arbitrary 3-SAT problem with size k is given, we write it as $\Phi = \phi_1 \wedge \phi_2 \ldots \wedge \phi_k$, ϕ_1 to ϕ_k are clauses like $x_1 \vee \bar{x}_2 \vee x_3$, we can convert it to a 4-SAT problem by following steps: For each clause ϕ_i, split it two 4-SAT clauses with adding another variable x, for example, $x_1 \vee \bar{x}_2 \vee x_3$, will be transformed to $(x_1 \vee \bar{x}_2 \vee x_3 \vee x) \wedge (x_1 \vee \bar{x}_2 \vee x_3 \vee \bar{x})$. Hence we transformed a 3-SAT problem $\Phi = \phi_1 \wedge \phi_2 \ldots \wedge \phi_k$ to a 4-SAT problem (with size $2k$) $\Phi' = (\phi_1 \wedge x) \wedge (\phi_1 \wedge \bar{x}) \ldots \wedge (\phi_k \wedge x) \wedge (\phi_k \wedge \bar{x})$. It's obviously that if the problem size of 3-SAT is k, then the reduction to 4-SAT will take $O(k)$ time, which is polynomial.
   - Proof
     We need to prove that $x_1, \ldots, x_n$ is a yes instance of Φ' (4-SAT problem) if anc only if $x_1, \ldots, x_n$ is a yes instance of Φ (3-SAT problem).
     - ($\Rightarrow$)
       Give the truth assignment $X = x_1, \ldots, x_n$ of Φ (3-SAT), because $z_1 \vee z_2 \vee z_3 = (z_1 \vee z_2 \vee z_3 \vee x) \wedge (z_1 \vee z_2 \vee z_3 \vee \bar{x})$, it's also the solution of Φ'
     - ($\Leftarrow$)
       Let $X = x_1, \ldots, x_n$ is a truth assignment of Φ'. Notice that for each group we have $(z_1 \vee z_2 \vee z_3 \vee x) \wedge (z_1 \vee z_2 \vee z_3 \vee \bar{x}) = z_1 \vee z_2 \vee z_3$, where $z_1 \vee z_2 \vee z_3$ is the corresponding clause in Φ, so if each group $(z_1 \vee z_2 \vee z_3 \vee x) \wedge (z_1 \vee z_2 \vee z_3 \vee \bar{x})$ is true, then each clause in Φ is also true. Hence, the 3-SAT problem Φ is satisfied.

Problem 2

Solution

1. Course scheduling $\in$ NP
   If an arbitrary solution is given, we can trverse all students' requests and check whether the number of conflicts out of limits in polynomial time and counting the conflicts.

2. 3-Color $\leq_p$ Course scheduling

   - Construction
     If an instance of 3-Color (an undirected graph $G = (V, E)$) is given, we can construct an instance $\{C, S, R, K\}$ of Course scheduling by following steps.
     (a) Let each course $c_1, \ldots, c_k$ corresponds to each vertex $v_1, \ldots, v_k$ in graph(i.e. $C = V$)
     (b) Let $S = \{1, 2, 3\}$, each number represents a color in 3-Color and different time slots in Course scheduling.
     (c) Let $R = \{\{(u, v) = e\}, \forall e \in E\}$, where each edge represents a student, and the endpoints represent the two courses he/she wants to take.
     (d) Finally, let $K = 0$

Problem 2 continued on next page...

This reduction can be done in polynomial time $O(|V| + |E|)$.

- Proof
  We need to prove that G is a yes instance of 3 Color if and only if that $\{C, S, R, K\}$ is a yes instance of Course scheduling.
  - $(\Rightarrow)$
    Suppose that G is a yes instance, which indicates no adjacent nodes have the same color. That means the two courses taken by any students will never conflict. Then $\{C, S, R, K\}$ is a yes instance of Course scheduling.
  - $(\Leftarrow)$
    Suppose that $\{C, S, R, K\}$ is a yes instance, thus no student will take two courses which are at the same time. So no adjacent nodes will have the same color, then we will see G is a yes instance of 3-Color.

Problem 3

Solution

1. TA-Cycle $\in$ NP
   If an arbitrary solution of TA-Cycle $A_1, A_2, \ldots, A_K$ is given, we can check it in polynomial time by traversing $A_1, A_2, \ldots, A_K$, and check whether A_i and its neighbor has a TA relationship.
2. Directed Hamilton Cycle $\leq_p$ TA-Cycle
   - Construction
     If an instance of Directed Hamilton Cycle (a directed graph $G = (V, E)$) is given, we can construct a instance of TA-Cycle by following steps.
     (a) Let each vertex v_i corresponds to a student A_i (i.e. $A = V$)
     (b) If there is an edge $v_i \to v_j$ in G, then let A_i is TA of A_j

     This reduction can be done in polynomial time $O(|V| + |E|)$.
   - Proof
     We need to prove that G is a yes instance of directed hamilton Cycle iff $A_1, A_2, \ldots, A_k$ is yes instance of TA-Cycle
     - $(\Rightarrow)$
       Suppose that $A_1, A_2, \ldots, A_k$ is an yes instance of TA-Cycle, then without lost of general, assume that $A_1, A_2, \ldots, A_K$ is the TA cycle, which means A_1 is TA of A_2, ... and A_k is TA of A_1. That means in G, there is edges: $v_1 \to v_2$, ... $v_K \to v_1$. Hence, there exists a hamiltonian cycle with length K in G: $v_1, v_2, \ldots, v_K$.
     - $(\Leftarrow)$
       Suppose that G is an yes instance, then there exists a hamiltonian Cycle with length K, without lost of general, assume $v_1, v_2, \ldots, v_K$ is the vertices in hamiltonian cycle, which means there is edges: $v_1 \to v_2$, ... $v_K \to v_1$. Then the corresponding students $A_1, A_2, \ldots, A_K$ is a TA-Cycle.

2 1 / 1

- ✓ **- 0 pts** Correct
- **- 1 pts** Click here to replace this description.
- **- 1 pts** This is not a 4-coloring problem, here is to allow conflicts no greater than K.
- **- 1 pts** □□□□□□□□□□□□
- **- 0.2 pts** Make the number of conflicts no more than K? You did not specify the parameters of K as 0.
- **- 0.1 pts** Each edge represents a student?
- **- 0.1 pts** There is a conflict in student in R? □□□□□
- **- 0.2 pts** It is not stated that a student can only take two courses.
- **- 0.1 pts** K--?
- **- 0.2 pts** K coloring problem needs the parameter k = 0 □Refers to the k in problem2□
- **- 0.2 pts** Edges is defined wrongly
- **- 0.3 pts** □□□□problem2□NP□□
- **- 0.3 pts** □□□□□□□□□□
- **- 0.8 pts** Problem2 -> K+k coloring? And 3-coloring -> K coloring? != 3-coloring -> Problem2
- **- 0.3 pts** □□□□□□□□□□□
- **- 0.5 pts** □□□□□□□□□□□
- **- 0.8 pts** □□□□□
- **- 0.4 pts** edges and vertices are defined wrongly

This reduction can be done in polynomial time $O(|V|+|E|)$.

- Proof

  We need to prove that G is a yes instance of 3 Color if and only if that $\{C, S, R, K\}$ is a yes instance of Course scheduling.

  - ($\Rightarrow$)

    Suppose that G is a yes instance, which indicates no adjacent nodes have the same color. That means the two courses taken by any students will never conflict. Then $\{C, S, R, K\}$ is a yes instance of Course scheduling.

  - ($\Leftarrow$)

    Suppose that $\{C, S, R, K\}$ is a yes instance, thus no student will take two courses which are at the same time. So no adjacent nodes will have the same color, then we will see G is a yes instance of 3-Color.

Problem 3

Solution

1. TA-Cycle $\in$ NP

   If an arbitrary solution of TA-Cycle $A_1, A_2, \ldots, A_K$ is given, we can check it in polynomial time by traversing $A_1, A_2, \ldots, A_K$, and check whether A_i and its neighbor has a TA relationship.

2. Directed Hamilton Cycle $\leq_p$ TA-Cycle

   - Construction

     If an instance of Directed Hamilton Cycle (a directed graph $G=(V,E)$) is given, we can construct a instance of TA-Cycle by following steps.

     (a) Let each vertex v_i corresponds to a student A_i (i.e. $A=V$)

     (b) If there is an edge $v_i \rightarrow v_j$ in G, then let A_i is TA of A_j

     This reduction can be done in polynomial time $O(|V|+|E|)$.

   - Proof

     We need to prove that G is a yes instance of directed hamilton Cycle iff $A_1, A_2, \ldots, A_k$ is yes instance of TA-Cycle

     - ($\Rightarrow$)

       Suppose that $A_1, A_2, \ldots, A_k$ is an yes instance of TA-Cycle, then without lost of general, assume that $A_1, A_2, \ldots, A_K$ is the TA cycle, which means A_1 is TA of A_2, ... and A_k is TA of A_1. That means in G, there is edges: $v_1 \rightarrow v_2$, ... $v_K \rightarrow v_1$. Hence, there exists a hamiltonian cycle with length K in G: $v_1, v_2, \ldots, v_K$.

     - ($\Leftarrow$)

       Suppose that G is an yes instance, then there exists a hamiltonian Cycle with length K, without lost of general, assume $v_1, v_2, \ldots, v_K$ is the vertices in hamiltonian cycle, which means there is edges: $v_1 \rightarrow v_2$, ... $v_K \rightarrow v_1$. Then the corresponding students $A_1, A_2, \ldots, A_K$ is a TA-Cycle.

3 **1 / 1**

✓ **- 0 pts** Correct

- 0.3 pts Please show TA cycle is in NP

- 0.7 pts Only show TA cycle is in NP

- 0.2 pts Inaccurate construction

- 0.3 pts mapping function not in polynomial time.

- 1 pts Wrong

- 0.5 pts wrong construction

Problem 4

Solution

1. Knapsack $\in$ NP
   If an arbitrary solution to knapsack is given, it's obviously that we can check whether the total weight and total value are qualified in polynomial time, by just calculating the sum of the weights or values.

2. Subset sum $\leq_p$ Knapsack

   - Construction
     If an instance of Subset sum (S, t) is given (where t is the target). We can construct an instance of Knapsack (A, C, b, k) by following steps:
     (a) Let $A = \{a_1, ..., a_n\} = S$
     (b) Let $C = \{c_1, ..., c_n\} = S$
     (c) Let $k = b = t$

     This reduction can be done in polynomial time $O(n)$.
   - Proof
     We need to prove that (S, t) is a yes instance of Subset sum if and only if that (A, C, b, k) is a yes instance of Knapsack.
     - $(\Rightarrow)$
       Suppose that (S, t) is a yes instance, thus there exists a subset $S' \in S$ such that $t = \sum_{s \in S'} s$. Because $S = A = C$, the knapsack can be fulfilled by item with weight and value both of t. Hence, (A, C, b, k) is a yes-instance of Knapsack.
     - $(\Leftarrow)$
       Suppose that (A, C, b, k) is a yes-instance, thus we have following equations:
       $$\sum_{s \in S'} s \leq b = t$$
       and
       $$\sum_{s \in S'} s \geq k = t$$
       Hence, we have $t = \sum_{s \in S'} s$. Therefore (S, t) is a yes instance of Subset sum.

Problem 5

Solution

1. Binary quadratic programming problem $\in$ NP
   If an arbitrary solution of binary quadratic programming is given, then we can do the matrix multiplication to get the result of Ax (in $O(mn)$ time), then, determine $Ax \leq b$ in $O(m)$ time. So, checking will take polynomial time $O(mn)$. Hence, binary quadratic programming problem $\in$ NP.

2. 3-SAT $\leq_p$ Binary quadratic programming problem

   - Construction
     If an instance of 3-SAT problem is given, we can construct an instance of Binary quadratic programming problem by following steps: The basic idea is, construct inequalities corresponds to each clause, e.g. $x_1 \vee \bar{x}_2 \vee x_3$ will be transformed to $x_1 + (1 - x_2) + x_3 \geq 1$.

Problem 5 continued on next page...

4 1 / 1

- ✓ **- 0 pts** Correct
- **- 0.2 pts** Without proof that Knapsack is NP
- **- 0.2 pts** Wrong construction of reducing
- **- 1 pts** No answer or totally wrong
- **- 0.3 pts** Only show one direction of equivalence
- **- 0.1 pts** Unclear constuction
- **- 0.6 pts** Not showing the equivalence of answer to two problems

Problem 4

Solution

1. Knapsack $\in$ NP
   If an arbitrary solution to knapsack is given, it's obviously that we can check whether the total weight and total value are qualified in polynomial time, by just calculating the sum of the weights or values.

2. Subset sum $\leq_p$ Knapsack

   - Construction
     If an instance of Subset sum (S, t) is given (where t is the target). We can construct an instance of Knapsack (A, C, b, k) by following steps:
     (a) Let $A = \{a_1, ..., a_n\} = S$
     (b) Let $C = \{c_1, ..., c_n\} = S$
     (c) Let $k = b = t$

     This reduction can be done in polynomial time $O(n)$.
   - Proof
     We need to prove that (S, t) is a yes instance of Subset sum if and only if that (A, C, b, k) is a yes instance of Knapsack.
     - $(\Rightarrow)$
       Suppose that (S, t) is a yes instance, thus there exists a subset $S' \in S$ such that $t = \sum_{s \in S'} s$. Because $S = A = C$, the knapsack can be fulfilled by item with weight and value both of t. Hence, (A, C, b, k) is a yes-instance of Knapsack.
     - $(\Leftarrow)$
       Suppose that (A, C, b, k) is a yes-instance, thus we have following equations:

       $$\sum_{s \in S'} s \leq b = t$$

       and

       $$\sum_{s \in S'} s \geq k = t$$

       Hence, we have $t = \sum_{s \in S'} s$. Therefore (S, t) is a yes instance of Subset sum.

Problem 5

Solution

1. Binary quadratic programming problem $\in$ NP
   If an arbitrary solution of binary quadratic programming is given, then we can do the matrix multiplication to get the result of Ax (in $O(mn)$ time), then, determine $Ax \leq b$ in $O(m)$ time. So, checking will take polynomial time $O(mn)$. Hence, binary quadratic programming problem $\in$ NP.

2. 3-SAT $\leq_p$ Binary quadratic programming problem

   - Construction
     If an instance of 3-SAT problem is given, we can construct an instance of Binary quadratic programming problem by following steps: The basic idea is, construct inequalities corresponds to each clause, e.g. $x_1 \vee \bar{x}_2 \vee x_3$ will be transformed to $x_1 + (1 - x_2) + x_3 \geq 1$.

Problem 5 continued on next page. . .

In more formal terms, for 3-SAT formula Φ, containing n variables and m clauses, construct $m \times n$ matrix A such that:

$$A_{i,j} = \begin{cases} -1, & \text{If variable } j \text{ only occurs without negation in clause } i \\ 1, & \text{If variable } j \text{ only occurs with negation in clause } i \\ 0, & \text{otherwise} \end{cases}$$

Then, constructs b by

$$b_i = -1 + \sum_{j=1}^{n} \max(A_{i,j}, 0)$$

(i.e. $b_i =$ the number of negated literals in clause i) Also, constructs x by:

$$x_i = \begin{cases} 1, & \text{If variable } i \text{ is assigned to True} \\ 0, & \text{If variable } i \text{ is assigned to False} \end{cases}$$

This reduction can be done in polynomial time $O(mn)$

- Proof
  We need to prove that (A, x, b) is a yes instance of Binary quadratic programming problem if and only if Φ is a yes instance of 3-SAT.
  - ($\Rightarrow$)
    If Φ is a yes instance of 3-SAT, whicn means all clauses ϕ_i is satisfied. That means the sum of satisfied literals in clause ϕ_i is ≥ 1. That indicates $y_i \leq b_i$ in Binary quadratic programming problem. If all ϕ_i is satisfied, then we have $\forall i, y_i \leq b_i$, which means $y \leq b$. Hence, (A, x, b) is a yes instance of Binary quadratic programming problem.
  - ($\Leftarrow$)
    If (A, x, b) is a yes instance of Binary quadratic programming problem, then we have $\forall i, y_i \leq b_i$. Then, in 3-SAT problem, we have, the sum of satisfied literals in clause i is ≥ 1, which indicates clause ϕ_i is satisfied. Hence, we can get all clauses are satisfied, hence Φ is a yes instance of 3-SAT problem.

Problem 6

Solution

1. Set packing problem $\in$ NP
   Lemma: we can find the intersection of two sets with size m and n in polynomial time. (By traversing or using a hash set are both OK)
   If an arbitrary solution of Set packing problem is given (with size=k), we can traverse the k sets and calculate $\cap_{i=0}^{k} S_i$, and determine whether the intersection is an Empty set. By the lemma, it can be done in polynomial time.

2. Independent set $\leq_p$ Set packing problem

   - Construction
     If an instance of independent set is given (i.e. a graph $G = (V, E)$ and a set of vertices S), we can construct an instance of Set packing problem by following steps:
     - For each vertex v in S, Initialize $S_v = \{\}$ in Set packing problem.

Problem 6 continued on next page. . .

5 **1 / 1**

✓ **+ 1 pts** Correct

- 0.1 pts the value of b is wrong

- 0.1 pts wrong when handling Ax<=b and Ax >=b

+ 0.4 pts wrong, but not totally, and for your hard work

+ 0.2 pts np

+ 0.5 pts hard work

+ 0 pts Wrong, or can not find your solution

+ 0.5 pts construct

+ 0.3 pts proof

- 0.2 pts you can not constraint that x_i = 1-x_j

- 0.1 pts small mistake

In more formal terms, for 3-SAT formula Φ, containing n variables and m clauses, construct $m \times n$ matrix A such that:

$$A_{i,j} = \begin{cases} -1, & \text{If variable } j \text{ only occurs without negation in clause } i \\ 1, & \text{If variable } j \text{ only occurs with negation in clause } i \\ 0, & \text{otherwise} \end{cases}$$

Then, constructs b by

$$b_i = -1 + \sum_{j=1}^{n} \max(A_{i,j}, 0)$$

(i.e. $b_i =$ the number of negated literals in clause i) Also, constructs x by:

$$x_i = \begin{cases} 1, & \text{If variable } i \text{ is assigned to True} \\ 0, & \text{If variable } i \text{ is assigned to False} \end{cases}$$

This reduction can be done in polynomial time $O(mn)$

- Proof

  We need to prove that (A, x, b) is a yes instance of Binary quadratic programming problem if and only if Φ is a yes instance of 3-SAT.

  - ($\Rightarrow$)

    If Φ is a yes instance of 3-SAT, whicn means all clauses ϕ_i is satisfied. That means the sum of satisfied literals in clause ϕ_i is ≥ 1. That indicates $y_i \leq b_i$ in Binary quadratic programming problem. If all ϕ_i is satisfied, then we have $\forall i, y_i \leq b_i$, which means $y \leq b$. Hence, (A, x, b) is a yes instance of Binary quadratic programming problem.
  - ($\Leftarrow$)

    If (A, x, b) is a yes instance of Binary quadratic programming problem, then we have $\forall i, y_i \leq b_i$. Then, in 3-SAT problem, we have, the sum of satisfied literals in clause i is ≥ 1, which indicates clause ϕ_i is satisfied. Hence, we can get all clauses are satisfied, hence Φ is a yes instance of 3-SAT problem.

Problem 6

Solution

1. Set packing problem $\in$ NP

   Lemma: we can find the intersection of two sets with size m and n in polynomial time. (By traversing or using a hash set are both OK)

   If an arbitrary solution of Set packing problem is given (with size=k), we can traverse the k sets and calculate $\cap_{i=0}^{k} S_i$, and determine whether the intersection is an Empty set. By the lemma, it can be done in polynomial time.

2. Independent set $\leq_p$ Set packing problem

   - Construction

     If an instance of independent set is given (i.e. a graph $G = (V, E)$ and a set of vertices S), we can construct an instance of Set packing problem by following steps:

     - For each vertex v in S, Initialize $S_v = \{\}$ in Set packing problem.

  - For each vertex v in S, add all edges adjacent to v to S_v:
    For all S_i, if there exists an edge from v_i to v_j, then add j to S_i (i.e. $S_i := S_i + \{j$, if there exists an edge from v_i to $v_j\}$) For example, if v_3 connects to v_1 and v_4, then $S_3 = \{1, 4\}$

  This reduction can be done in polynomial time $O(|V| + |E|)$.

- Proof
  We need to prove that G is a yes instance of independent sets if and only if $S_1, \ldots, S_k$ is a yes instance of Set packing problem.
  - $(\Rightarrow)$
    If graph $G = (V, E)$ and a set S is a yes instance of independent set, then for each pair of vertices v_i and v_j, there does not exist edges connects v_i and v_j. Hence the subset $S_1, \ldots, S_k$ will contain no overlaps (pairwise disjoint). Therefore, $S_1, \ldots, S_k$ is a yes instance of Set packing problem.
  - $(\Leftarrow)$
    If $S_1, \ldots, S_k$ is a yes instance of Set packing problem, which means $S_1, \ldots, S_k$ are pairwise disjoint. That indicates for each pair of vertices v_i and v_j in S, there does not exist edges connects v_i and v_j, hence the set of vertices S is an independent set.

6 1 / 1

✓ **- 0 pts** Correct

- 0.1 pts "In NP" step is wrong

- 0.1 pts "mention mapping takes polynomial time" step is missing. Or the construction is not from a Independent Set "Instance" to Pairwise Disjoint "instance".

- 0.4 pts "Instance Construction" step is wrong or missing.

- 0.4 pts "Proof Instances Yes/No Equal" step is wrong or missing.

- 1 pts Wrong.