CS240 Homework #4

Zhiqiang Xie 77892769

Problem 1: Stingy SAT

We call the problem described here "Stingy SAT", in formal-language terms:

STINGY-SAT = $\{< \phi, k >: \phi$ is a satisfiable boolean formula with at most k variables are true$\}$

1. STINGY-SAT $\in$ NP.
    - Given an arbitary solution to Stingy SAT, we can evaluate the CNF formula in polynomial time by basic logical operations. And it just takes $O(1)$ time more to check whether the number of variables which are assigned to be true is less or equal to k.
2. SAT $\leq_p$ STINGY-SAT
    - Transformation:
        - Given a SAT formula f with n variables, we choose (f, n) as an instance of Stingy SAT.
    - Proof:

      We now need to prove that f is a yes-instance of SAT if and only if that (f, n) is a yes-instance of Stingy SAT.
        - ($\Rightarrow$) Suppose that f is a yes-instance, and we can see that no more than n variables in the formula can be true since n is the amount of variables. So any satisfying assignment of instance f will be a satisfying assignment of instance (f, n). Therefore, (f, n) is a yes-instance of Stingy SAT.
        - ($\Leftarrow$) Suppose that (f, n) is a yes-instance, any satisfying assignment of it's certainly a satisfying assignment of instance f. Therefore, f is a yes-instance of SAT.

Stingy SAT problem is NP-complete.

Problem 2: Zero Weight Cycle

We call a simple cycle with the sum of its edge weights to be exactly 0 "zero weight cycle", in formal-language terms:

Zero-Weight-Cycle = $\{< G >: G$ is a directed graph containing a zero weight cycle$\}$

1. Zero-Weight-Cycle $\in$ NP.

- Given an arbitary solution to it $G_s \subseteq G$, we can check whether G_s is a zero weight cycle in polynomial time by checking whether the sum of its edge weights is exactly 0.

2. Directed-Hamiltonian-Cycle $\leq_p$ Zero-Weight-Cycle
   - Transformation:
     - Given an instance of Directed-Hamiltonian-Cycle, a directed graph $G = (V, E)$, where $n = |E|, E = \{e_1, e_2, \ldots, e_n\}$, we construct a new directed graph $G' = (V', E')$. G' contains n components $\{G_1, G_2, \ldots, G_n\}$.
       - Where $G_n = (V, E_n)$, E_n means $weight(e_n) = -(|V| - 1)$ and weights of other edges are 1.

       This transformation can be done in polynomial time $(O(n^2))$.
     - Now we set the directed graph G' to be an instance of Zero-Weight-Cycle.
   - Proof:

     We now need to prove that G is a yes-instance of Directed-Hamiltonian-Cycle if and only if that G' is a yes-instance of Zero-Weight-Cycle.
     - $(\Rightarrow)$ Suppose that G is a yes-instance, we'll see that at least one component $G_i \in \{G_1, G_2, \ldots, G_n\}$ contains a zero weight cycle (the hamiltonian cycle). Thus, G' contains at least one zero weight cycle. Therefore G' is a yes-instance of Zero-Weight-Cycle.
     - $(\Leftarrow)$ Suppose that G' is a yes-instance, and we know the only way to produce a zero weight cycle in any component $G_i \in \{G_1, G_2, \ldots, G_n\}$ is a hamiltonian cycle: $0 = (|V| - 1) * 1 + (-(|V| - 1))$. Thus, G is a hamiltonian graph.

       Therefore G is a yes-instance of Directed-Hamiltonian-Cycle.

Zero Weight Cycle problem is NP-complete.

Problem 3: Dominating Set

We call the problem described here "Dominating Set", a *dominating set* of an graph $G = (V, E)$ is a subset $V' \in V$ such that for every $v \in V$, we have $v \in V'$ or at least one of its neighbors $\in V'$, in formal-language terms:

DOMINATING-SET = $\{< G, K >:$ graph G has a dominating set of size $K\}$.

1. DOMINATING-SET $\in$ NP.
   - Given an arbitary solution to it $V' \in V$, we can check whether V' is a dominating set in polynomial time by checking whether for each $v \in V$, $v \in V'$ or at least one of its neighbors $\in V'$. And it just takes $O(1)$ time more to check whether the number of nodes in V' is less or equal to K.
2. VERTEX-COVER $\leq_p$ DOMINATING-SET
   - Transformation:

- Given an instance of VERTEX-COVER, a undirected graph $G = (V, E, k)$, we can construct a new undirected graph $G' = (V', E', k)$:
  - $V' = \{f(v), \forall v \in V, f(e), \forall e \in E\}$, where f is a function to produce a vertex, and $|V'| = |V| + |E|$.
  - $E' = \{g(v, e), \forall e = (v, \_) or (\_, v), g(u, v), \forall u, v \in V u \neq v\}$, where g is a function to produce a edge. Note here is a clique in G' with vertices $\{f(v), \forall v \in V\}$.
  - Now we set the undirected graph G' to be an instance of DOMINATING-SET.

  This transformation can be done in polynomial time $(O(n^2))$.

- Proof:

  We now need to prove that G is a yes-instance of VERTEX-COVER if and only if that G' is a yes-instance of DOMINATING-SET.

  - $(\Rightarrow)$ Suppose that G is a yes-instance, thus we have a set $V_s \in V$ to be a vertex cover with size k'. Then here's a dominating set in G' with size k', because all $f(e) \in V'$ are connected to nodes in $\{f(v), \forall v \in V_s\}$. Where $k' \leq k$.

    Therefore G' is a yes-instance of DOMINATING-SET.

  - $(\Leftarrow)$ Suppose that G' is a yes-instance, thus we have a dominating set $V_d \in V'$ with size k'. If V_d contains any $f(e)$, we can replace it by $f(v)$, where v is either of the endpoints of e. This replacement doesn't destroy the property of dominating set, because any $f(e)$ just connects to two $f(v)$, where v are the endpoints, and every $f(v)$ are connected to each other. Similarly, we can see here's a vertex cover with size k' in G. Where $k' \leq k$.

    Therefore G is a yes-instance of VERTEX-COVER.

Dominating Set problem is NP-complete.

Problem 4: Course Scheduling

We call the problem described here "Course Scheduling ", it's somehow hard to write a formal-language term since there is no need to format it neatly.

1. COURSE-SCHEDULING $\in$ NP.
   - Given an arbitary solution to it, we can check whether the number of conflicts out of limits in polynomial time by checking all students' requests and counting the conflicts.
2. 3-COLOR $\leq_p$ COURSE-SCHEDULING
   - Transformation:
     - Given an instance of 3-COLOR, a undirected graph $G = (V, E)$, we can construct an instance (C, S, R, K) of COURSE-SCHEDULING.
       - $C = V$
       - $S = \{1, 2, 3\}$, each number represents a color and different time.

        - $R = \{\{u, v, (u, v) = e\}, \forall e \in E\}$, where each edge represents a student, and the endpoints represent the two courses he/she takes.
        - $K = 0$

      This transformation can be done in polynomial time $(O(m + n))$.

    - Proof:

      We now 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, thus no adjacent nodes have the same color, which means the two courses taken by any students will never conflict.

        Therefore (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 takes two courses which are at the same time. So we have no adjacent nodes have the same color, recall the definition of 3-COLOR problem we'll see (C, S, R, K) is a yes-instance of COURSE-SCHEDULING.

Course Scheduling problem is NP-complete.

Problem 5: Two Trucks

We call the problem described here "Two Trucks", in formal-language terms:

TWO-TRUCKS = $\{< G, s, K >: G = (V, E)$ is a directed weighted complete graph, s is a starting location $s \in V, k \in Z, G$ has two cycles both start at s inside it with length at most $K, \forall v \in V, v$ is on at least one of the two cycles$\}$

1. TWO-TRUCKS $\in$ NP.
    - Given an arbitary solution to it, we can check whether every location are visited and the length of two cycles out of limits in polynomial time by traversing and counting.
2. TSP $\leq_p$ TWO-TRUCKS
    - Transformation:
      - Given an instance of TSP, a directed weighted complete graph $G = (V, E)$, we can construct a new directed weighted complete graph $G' = (V', E')$. Note all weights of edges are **nonnegative integers**.
        - $V' = V + a$, where a is an additional vertex.
        - $E' = E + \{(v, a), (a, v), \forall v \in V\}$
        - $d(s, a) = K, d(a, s) = 0$, where s is a starting location which doesn't matter.
        - $d(v, a) = d(a, v) = K + 1, \forall v \in \{V - s\}$
      - Now we get the new graph (G', s, K) to be an instance of Two Trucks.

      This transformation can be done in polynomial time $(O(m + n))$.

    - Proof:

We now need to prove that (G, K) is a yes-instance of TSP if and only if that (G', s, K) is a yes-instance of TWO-TRUCKS.

- ($\Rightarrow$) Suppose that (G, K) is a yes-instance, thus we can deploy a truck to run a traverling-salesman tour with cost k', and another to run the cycle path $\{s, a, s\}$ with cost K. Where $k' \leq K$.

  Therefore (G', s, K) is a yes-instance of TWO-TRUCKS.

- ($\Leftarrow$) Suppose that (G', s, K) is a yes-instance, since we can't run a cycle including location a with length less or equal to K except the cycle path $\{s, a, s\}$. One of the two trucks has to run this specfic cycle, so another truck has to run a traverling-salesman tour with cost at most K to make it yes. Thus here's a traverling-salesman tour in graph G with cost at most K.

  Therefore (G, K) is a yes-instance of TSP.

Two Trucks problem is NP-complete.

Problem 5: Knapsack

1. KNAPSACK $\in$ NP.
   - Given an arbitary solution to it, we can check whether the total weight and total value are qualified in Polynomial time by simple calculation.
2. SUBSET-SUM $\leq_p$ KNAPSACK
   - Transformation:
     - Given an instance of SUBSET-SUM (S, t), where t is the target. We can construct an instance of KNAPSACK (A, C, b, k):
       - $A = \{a_1, \ldots, a_n\} = S, b = t$
       - $C = \{c_1, \ldots, c_n\} = S, k = t$
   - Proof:

     We now 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 here exists a subset $S' \in S$ such that $t = \sum_{s \in S'} s$. Since $S = A = C$ even with the same order, the knapsack can be fulfilled by weight and value of t.

       Therefore (A, C, b, k) is a yes-instance of KNAPSACK.

     - ($\Leftarrow$) Suppose that (A, C, b, k) is a yes-instance, thus we have these two inequations
       - $\sum_{s \in S'} s \leq b = t$
       - $\sum_{s \in S'} s \geq k = t$

       Thus, $t = \sum_{s \in S'} s$ holds. Therefore (S, t) is a yes-instance of SUBSET-SUM.

Knapsack problem is NP-complete.