CS3230: Design and Analysis of Algorithms
Semester 2, 2022-23, School of Computing, NUS

Practice Problem Set 5

April 17, 2023

Question 1: Consider the *MAX-2-SAT problem*: Given a 2-CNF formula ϕ with m clauses and an integer $1 \leq k \leq m$, decide whether there is an assignment to the variables that satisfies at least k clauses of ϕ. Show that MAX-2-SAT is NP-complete. [**Hint:** Try a reduction from 3-SAT.]

Answer: First we note that if a truth setting of the variable makes at least k clauses true, then by substituting these truth values we can efficiently check the truth or falsity clause by clause, keeping a counter to hold the number of clauses found to be true so far. Hence *MAX-2-SAT* is in NP.

The following is a reduction from 3SAT to *MAX-2-SAT*. That is, given an instance of 3SAT we construct an instance of MAX-2-SAT so that a satisfying truth assignment of 3SAT can be extended to a satisfying truth assignment of MAX-2-SAT. Note that a satisfying truth assignment is one that makes at least k clauses true in the MAX-2-SAT instance.

Let S be the instance of 3SAT where the clauses are $C_1, ..., C_m$ where $C_i = \{x_i, y_i, z_i\}$, where each x_i, y_i, and z_i represents either a variable or its negation and $1 \leq i \leq m$. From S we build an instance S' of MAX-2-SAT as follows. Each C_i in S corresponds to a clause group C'_i in S' where,

$C'_i = \{(w_i), (x_i), (y_i), (z_i), (\bar{x}_i \vee \bar{y}_i), (\bar{y}_i \vee \bar{z}_i), (\bar{x}_i \vee \bar{z}_i), (\bar{w}_i \vee x_i), (\bar{w}_i \vee y_i), (\bar{w}_i \vee z_i)\}$

where w_i is a new variable and $1 \leq i \leq m$. We set $k = 7m$

Each clause in S' as constructed above has at most two literals. It can be seen that the clauses in S' can be efficiently generated from the clauses in S in polynomial time. We now argue that a satisfying truth assignment of S exists if and only if it can be extended to a satisfying truth assignment for S' appropriately.

Assume that S is satisfiable. Then in a typical clause $C_i = \{x_i, y_i, z_i\}$ either one or two or all the three variables are true.

- Let $x_i = T, y_i = F, z_i = F$. With this truth assignment in S', if $w_i = T$, six clauses of C'_i become true; if $w_i = F$, seven clauses of C'_i become true.
- Let $x_i = T, y_i = T, z_i = F$. This truth assignment in S' together with $w_i = T$ or $w_i = F$,makes seven clauses of C'_i true.
- Let $x_i = T, y_i = T, z_i = T$. With this truth assignment in S', if $w_i = T$, seven clauses of C'_i become true; if $w_i = F$, six clauses of C'_i become true.

In summary, a satisfying truth assignment of S can be extended to a satisfying truth assignment of S' where exactly seven clauses in each clause group get satisfied. Moreover no setting of w_i causes more than seven of the ten clauses to be true in each clause group in S'.

Now assume that S is not satisfiable. Then in at least one clause $C_i = \{x_i, y_i, z_i\}$ in S, we have $x_i = F, y_i = F, z_i = F$. With this truth assignment in S', if $w_i = T$, four clauses of C'_i become true; if $w_i = F$, six clauses of C'_i become true. That is, if S is not satisfiable, no truth setting can make at least seven clauses true in each clause group C'_i. Since 3SAT is NP-complete, we conclude that MAX-2-SAT is also NP-complete.

Question 2: Consider the *Max-Clique problem*: Given an undirected graph $G = (V, E)$ and an integer k decide whether there exists a clique of size at least k in G (i.e., as a subgraph of G). Show that Max-Clique problem is NP-complete. [**Hint:** Try a reduction from the Maximum Independent Set problem.]

Answer: It suffices to show (a) the Max-Clique problem is in NP, and (b) Maximum Independent Set problem(IS) $\leq_p$ Max-Clique problem(CP).

To show (a), given a graph G and a positive integer k, we should be able to verify the certificate in polynomial time. The certificate is a subset V' of the vertices, which comprises the vertices belonging to the clique. We can validate this solution by checking that each pair of vertices belonging to the solution are adjacent, by simply verifying that they share an edge with each other. This can be done in polynomial time.

To show (b), let $G = (V, E)$ and $(G, k) \in IS$ then,

$(G, k) \in IS \iff$ G as independent subset of size at least k

$\iff \overline{G}$ has a clique of size at least k

$\iff (\overline{G}, k) \in CP$

Therefore, any instance of the clique problem can be reduced to an instance of the Independent Set problem. Therefore, Max-Clique problem is NP-complete.

Question 3: We have seen the (undirected) Hamiltonian cycle problem in week 11 tutorial. In a similar way, we can defined the Directed Hamiltonian Cycle problem as follows: Given a directed graph whether there exists a directed cycle that visits each vertex exactly once. There is a slightly intricate poly-time reduction from the 3-SAT problem to the directed Hamiltonian cycle problem (which we will not see in this course, but you may assume that the directed Hamiltonian cycle is NP-complete).

Now we ask you to give a poly-time reduction from the directed Hamiltonian Cycle problem to the (undirected) Hamiltonian Cycle problem. (Note, we have already seen in week 11 lecture that the Hamiltonian Cycle problem is is in NP. So we get that the (undirected) Hamiltonian Cycle problem is NP-complete.)

Answer: Quick intuition: We need a way to inform the undirected Hamiltonian cycle problem of the direction that is present in the directed version. This will be done by representing each node by 3 nodes. Let's dive right into it.

To show $DHC \leq_P UHC$, let's start with the polytime transformation.

Given a graph $G = (V, E)$ for the DHC problem, let's create a $G' = (V', E')$ where G' is an undirected graph and G has a Hamiltonian cycle iff G' has a Hamiltonian cycle. For every vertex $v \in G$, it will be represented in G' as 3 vertices, v', v'', and v''' with undirected edges (v', v'') and (v'', v'''). Then for every directed edge (u, v) in G, it will be represented by the undirected edge (u''', v') in G'. This procedure is clearly a polynomial transformation. Nice! 1/3 done.

Let's move on to the next part, if there is a Hamiltonian cycle in G there is a Hamiltonian cycle in G'. Suppose G has the HC: $v_1, v_2, ..., v_n$, we can quickly see that the a Hamiltonian cycle also exists in G', $v_1', v_1'', v_1''', v_2', v_2'', v_2''', ...v_n', v_n'', v_n'''$. Yay 2/3 of the way there.

For the final part we need to show that if there is a Hamiltonian cycle in G', there is a Hamiltonian cycle in G. Suppose that G' has a HC. Since each of the vertices v_i'' has degree 2, they must be preceded by v_i' and succeeded by v_i''' in this cycle (or the other way around). Therefore, if the vertices v_i'' appear in the cycle in the order $v_1'', v_2'', ..., v_n''$, then it the cycle must look like $v_1', v_1'', v_1''', v_2', v_2'', v_2''', ...v_n', v_n'', v_n'''$ in G' and thus $v_1, v_2, ..., v_n$ in G. Okay some of you guys might be thinking, what if there exists a v_i in the cycle that is traversed in the order v_i''', v_i'', v_i', wouldn't this break our invariant on direction? Good question! Let's say our cycle is $v_1', v_1'', v_1''', v_2', v_2'', v_2''', ...v_i''', v_i'', v_i'...v_n', v_n'', v_n'''$. This means that vertices that represent v_{i-1} G' must also be visited in the order $v_{i-1}''', v_{i-1}'', v_{i-1}'$ because there can only be edges between u' and v''' in G'. So if we extend this, we'll get the same cycle but in the opposite order! NICE! I hope you're convinced now.

Question 4: Give a poly-time reduction from circuit satisfiability to CNF-SAT. [**Hint:** For each gate k introduce a new variable x_k. Then create a sub-formula for each gate. e.g. Suppose for k-th gate the input wires are from (output wire of) i and j-th gate. Then the sub-formula will be $(\overline{x_k} \vee x_i) \wedge (\overline{x_k} \vee x_j) \wedge (x_k \vee \overline{x_i} \vee \overline{x_j})$. Similarly define sub-formulas for OR and NOT gates.]

Answer: We will need to add new variables. Suppose the circuit C has m gates, including input gates, then we introduce new variables $x_1, ..., x_m$, with the intended meaning that variable x_j corresponds to the output of gate j.

We make a formula F which is the AND of $m+1$ sub-expression. There is a sub-expression for every gate j, saying that the value of the variable for that gate is set in accordance to the value of the variables corresponding to inputs for gate j.

We also have a $(m+1)$-th term that says that the output gate outputs 1. There is no sub-expression for the input gates.

For a gate j, which is a NOT applied to the output of gate i, we have the sub-expression
$(x_i \vee x_j) \wedge (\overline{x}_i \vee \overline{x}_j)$

For a gate k, which is an AND applied to the output of gates i and j, we have the sub-expression
$(\overline{x_k} \vee x_i) \wedge (\overline{x_k} \vee x_j) \wedge (x_k \vee \overline{x_i} \vee \overline{x_j})$

We have a similar sub-expression for OR gate.

We now have to show that this reduction works.Suppose C is satisfiable, then consider setting x_j being equal to the output of the j-th gate of C when a satisfying set of values is given in input. Such a setting for $x_1, ..., x_m$ satisfies F.

Suppose F is satisfiable, and give in input to C the part of the assignment to F corresponding to input gates of C. We can prove by induction that the output of gate j in C is also equal to x_j, and therefore the output gate of C outputs 1.

So C is satisfiable if and only if F is satisfiable.

Question 5: Give a poly-time reduction from CNF-SAT to 3-SAT. [**Hint:** For the clauses with more than three literals try to write them as AND of several clauses by introducing new variables.]

Answer: Take a formula F of CNF-SAT. We transform it into a formula F' of 3-SAT such that F' is satisfiable if and only if F is satisfiable.

Each clause of F is transformed into a sub-expression of F'. Clauses of length 3 are left unchanged.

A clause of length 1, such as (x) is changed as follows,
$(x \vee y_1 \vee y_2) \wedge (x \vee y_1 \vee \overline{y}_2) \wedge (x \vee \overline{y}_1 \vee y_2) \wedge (x \vee \overline{y}_1 \vee \overline{y}_2)$
where y_1 and y_2 are two new variables added specifically for the transformation of that clause.

A clause of length 2, such as $(x_1 \vee x_2)$ is changed as follows
$(x_1 \vee x_2 \vee y) \wedge (x_1 \vee x_2 \vee \overline{y})$
where y is a new variable added specifically for the transformation of that clause.

For a clause of length $k \geq 4$, such as $(x_1 \vee ... \vee x_k)$, we change it as follows
$(x_1 \vee x_2 \vee y_1) \wedge (\overline{y}_1 \vee x_3 \vee y_2) \wedge (\overline{y}_2 \vee x_4 \vee y_3) \wedge ... \wedge (\overline{y}_{k-3} \vee x_{k-1} \vee x_k)$
where $y_1, ..., y_{k-3}$ are new variables added specifically for the transformation of that clause.

We now have to prove the correctness of the reduction,

For reductions from length 1 and 2, we can construct the truth table and show that all instances where F is true the output of F' is true, and all instance of false are false.

For reductions from length 4 and above, When F is satisfiable, at least one literal in $x_1, ..., x_k$ is true. If any one of x_1 or x_2 is True; set all additional variables $y_1, ..., y_{k-3}$ to False. Now, first clause of all literals in F' becomes True, therefore F' has a satisfying assignment.

If any one of x_{k-1} or x_k is True, set all additional variables $y_1, ..., y_{k-3}$ to True. Now, last clause of all literals in F' becomes Trues, therefore F' has a satisfying assignment.

If any other literal is true, $x_i$where $i \notin \{1, 2, k-1, k\}$. Now if we make $y_1, ..., y_{i-2}$ True and make $y_{i-1}, ..., y_{k-3}$ False, we satisfy all the clauses in F'. Therefore, there is a satisfying assignment.

For an assignment which makes F false, all literals in $x_1, ..., x_k$ has to be false. In such a case there is no assignment of $y_1, ..., y_{k-3}$ which will make F' true. Therefore, all instances where F is false F' is also false.

Therefore, F' is satisfiable if and only if F is satisfiable.

Question 6: We have seen the Travelling Salesperson Problem (TSP) in the lecture. Now let us consider a different variant of that problem. Suppose given a complete graph with non-negative edge weights and an integer k the objective is to decide whether there is a TSP tour of cost at most k or at least $\alpha(n)k$, for some poly-time computable positive valued function $\alpha(\cdot)$. Show that the Hamiltonian Cycle problem is poly-time reducible to this variant of TSP problem. [**Hint:** Modify the reduction from the Hamiltonian Cycle to the TSP problem discussed in the lecture.]

Answer: The problem is actually really similar to the reduction that we've already done in the tutorial. Let's summarize this problem again:

- Input: k, V, E, and positive function $\alpha(n)$.

- Output yes if minimum TSP tour has cost $\leq k$, output no if minimum TSP tour has cost $> \alpha(n)k$. For the remaining instance, we do not care (i.e., output could be arbitrary).

Let us denote this variant of the TSP as $TSP_{variant}$. Now to show that $HC \leq_P TSP_{variant}$, the reduction is almost identical to the one in our tutorial. Let $G = (V, E)$ be an instance of HC. Build a graph G' on V vertices. For each pair (u, v): If $(u, v) \in E$, $c(u, v) = 1$ else $c(u, v) = \alpha(n) \times n + 1$. Now G has a HC iff G' has a TSP tour of cost at most n. Let's prove it!

1) Note that the reduction above can be done in polynomial time with respect to the input size For each vertices v in V we're adding at most v other edges and assigning weights to them. This means that the reduction step can be done in $O(v^2)$.

2) If G has a HC, then G' has a TSP tour of at most n. I'll omit the forward direction because it's identical to the lecture. Here is the tdlr: HC C exists in G, thus G must exist in G' (by definition of how we defined our reduction). Since C is in G, all the edges of C in G' must have cost 1, thus HC in G' is of cost n.

3) I believe that this is slightly trickier. As from the Yes instance of $TSP_{variant}$, we can only imply that the cost of the tour in G' in $\leq \alpha(n) \times n$. However from this fact, we can quickly see that this means that the tour cannot contain any edges that are not in G because if that were the case the cost of that single edge would already be $\alpha(n) \times n + 1$ leading to a contradiction. This means that all of the edges in the tour in G' must be in G. Now the rest of the proof is identical to that in the lecture notes.

With 1) 2) and 3), we've shown that HC $\leq_P TSP_{variant}$

All the best for your finals!