Solutions: Homework 8

Prepared by Chien-Chung Huang

1.

This algorithm for SAT proposes only one solution with exponential complexity. However, we don't know whether any *other* solution with lower complexity exsits or not. If there is (so far we are not sure), then $P = NP$. If we want the above claim about $P \neq NP$ to be true, we have to show that this solution is the only one and it is implssible to solve SAT in any other way.

2

To show that DOUBLESAT is NP-complete, We have to show (1) DOUBLESAT is in NP (2) every A in NP is polynomail time reducible to DOUBLESAT.

To show DOUBLESAT is in NP, we first construct a non-deterministic Turing machine NTM to solve DOUBLESAT.
NTM:
"On input $< \phi >$
(0) Set the counter value to 0.
(1) Nondeterministically decide a set of truth values for $x_1, x_2 ... x_n$ which appear in $< \phi >$.
(2) Decide whether this set of truth values can satisfy the boolean equation.
If yes, increment the counter.
(3) If the value of the counter is larger or equal to 2, ACCEPT, else REJECT. "
NTM obviously can execute in polynomial time.

We now try to show that 3SAT is polynomail time reducible to CLIQUE.
Let ϕ be a formula with k clauses such as

$$\phi = (a_1 \cup b_1 \cup c_1) \cap (a_2 \cup b_2 \cup c_2) \cap ... \cap (a_k \cup b_k \cup c_k)$$

We build another boolean equation ϕ' such that

$$\phi' = (a_1 \cup b_1 \cup c_1 \cup d_1) \cap (a_1 \cup b_1 \cup c_1 \cup \overline{d_1}) \cap (a_1 \cup b_1 \cup c_1 \cup d_2) \cap (a_1 \cup b_1 \cup c_1 \cup \overline{d_2}) ... (a_k \cup b_k \cup c_k \cup d_k) \cap (a_k \cup b_k \cup c_k \cup \overline{d_k})$$

We claim that ϕ is satisfiable iff ϕ' has more than two set of truth values that satisfy it.

If ϕ is satisfiable, every clause $(a_i \cup b_i \cup c_i)$ must be 1. Thus, $(a_i \cup b_i \cup c_i \cup d_i)$ and $(a_i \cup b_i \cup c_i \cup \overline{d_i})$ must both be 1 too, no matter d_i be 0 or 1. We conclude that ϕ' must have at least two set of assigments.

If, on the other hand, ϕ' is satisfiable, then $(a_i \cup b_i \cup c_i \cup d_i)$ and $(a_i \cup b_i \cup c_i \cup \overline{d_i})$ must both be 1. Since one of d_i and $\overline{d_i}$ must be 0, it means it is not possible that all a_i, b_i, c_i are 0s, otherwise, ϕ' is not satisfiable. If not all a_i, b_i, c_i are 0s, then the i's clause in ϕ must be 1. As a result, ϕ must be satisfiable.

3 **3.1** Here is an easy example. We set V= 100, $W = 1.5$. $w_1 = 1, w_2 = 1, w_3 = 0.7, w_4 = 0.7, v_1 = 99, v_2 = 80, v_3 = 51, v_4 = 51$. Using this algorithm, we only pick up object 1 and then will give up. However, we can achieve the goal by picking up object 3 and 4.

3.2 KNAPSACK problem belongs to NP, as we can nondeterministically choose a set of objects and veriry whether this set fulfill the requirement $\sum_{i\in S} w_i \leq W \cap \sum_{i\in S} v_i \geq V$.

We try to reduce SUBSET-SUM problem into KNAPSACK. SUBSET-SUM states that $\{< S, t > | S = \{x_1, ..., x_k\}$ and for some $\{y_1, ...y_l\} \subseteq \{x_1, ...x_k\}$, we have $\sum y_i = t\}$.

We create another KNAPSACK problem as follows:

$$\left\{ < 2k, x_1...x_k, x_1, ...x_k, t, t >: \exists S \subset \{1, ...2k\} \left(\sum_{i\in S} x_i \leq t \bigcap \sum_{i\in S} x_i \geq t \right) \right\}$$

We claim that if we know the solution of the SUBSET-SUM problem we also can solve the created KNAPSACK problem and vice versa. The reason is that the above KNAPSACK problem can only be solved wehn $\sum_{i\in S} x_i = t$.

4 **Problem 7.22a** For any clause in a $\neq$-assignment, we know two literals of them, say, x_1, x_2 have different values, i.e., 0 and 1. The negation of this $\neq$-assignment will make this very clause $\overline{x_1} \cup \overline{x_2} \cup \overline{x_3} = 1 \cup 0 \cup \overline{x_3}$. $\overline{x_3}$ can only be 1 or 0. Thus the negation of the $\neq$-assignment is also an $\neq$-assignment.

Problem 7.22b To show 3SAT is polynomial time reducible to $\neq$-SAT, we have to show that there exists a polynomial time computable funtion $f : \Sigma^* \to \Sigma^*$ exists which for every w, $w \in A \Leftrightarrow f(w) \in B$.
Such a function is already provided, i.e., by transforming each clause $(y_1 \cup y_2 \cup y_3)$ into two clauses $(y_1 \cup y_2 \cup z_i) \cap (\overline{z_i} \cup y_3 \cup b)$. This transformation obviously can be done in polynomial time. What we need to prove then is to show that $(y_1 \cup y_2 \cup y_3) = 1$ iff $(y_1 \cup y_2 \cup z_i) = 1$ and $(\overline{z_i} \cup y_3 \cup b) = 1$. and at least two out of the literals have different values.

We can prove this by showing exhaustively all possible combination of truth values.

y_1	y_2	y_3	z_i	b
1	1	1	0	0
1	1	0	0	0/1
0	1	1	1	0/1
1	0	1	1	0/1
0	0	1	1	0/1
1	0	0	0	0/1
0	1	0	0	0/1

Based on the above table, we know the truth values of a 3-SAT problem, we also can use it solve $\neq$SAT, and vice versa.

Problem 7.22c to show that $\neq$SAT is NP, we construct the following nondeterministic machine.
NTM:
"On input $< \phi >$
(1) Nondeterministically decide a set of truth values for $x_1, x_2...x_n$ which appear in $< \phi >$.
(2) Test whehter this set of truth values satisfies ϕ.
(3) If yes, ACCEPT, else REJECT. "
NTM obviously can execute in polynomial time.

From (b), we know that 3-SAT, which is known to be NP, can be reduced to ≠SAT. Thus, we conclude that ≠SAT is NP-complete.

Problem 7.23 MAX-CUT is NP. We can nondeterministically choose a set of k (or more) edges and decide whether all the nodes can be divided into disjoint sets by the chosen edges.

We now try to reduce ≠SAT into MAX-CUT. Given ϕ, for each variable x, we create a collection of $3k$ nodes labelled with x and another $3k$ nodes labeled with $\overline{x}$, where k is number of clauses in ϕ. All nodes labelled x are connected with all nodes labeled $\overline{x}$.
Besides, we connect three nodes lableded with the literals appearing in the same clause in ϕ. When connecting the nodes appearing in the same clause, we have to make sure that the same node is used at most once.

We now claim that if ϕ has a solution iff we know the maximum number of edges that can be cut from this graph.

We first show if we know the solution of ϕ, we can decide the maximum number of cuts.

For each clause, we know there are at three literals (corresponding to three nodes in the graph). One literal have a different value from the other two. We cut the two edges connecting the node with the different value. Besides, we also cut all edges linking the nodes with its complement nodes.

We claim that every edge that is cut connects two nodes, one assigned vaue 0 and the other 1. There are two cases to consider:

(1) The edge connecting a node x and its complment $\overline{x}$, which obviousy are assigned different values.
(2) The edge connects two nodes appearinging in the same clause. We have explicitly choose those edges connecting different values. So this is obviously true.

On the other direction, we claim that if we know the max number of edges to be cut, we know how to assign the values to the problem ≠SAT. Using the strategy when we choose the edges to be cut in reverse order, we assign the same value to the edge that is not cut in an triangle (three literals in the same clause) and different values to those which are cut.