Homework 3
March 29, 2021
TangLin

PROBLEM 1. Prove there is **PSPACE**-complete problem.

Solution:

We will consider the language **TQBF** which is true quantified boolean formulae with form

$$Q_1x_1Q_2x_2\cdots Q_nx_n\varphi(x_1,x_2,\cdots,x_n)$$

where Q_i is $\forall$ or $\exists$ and φ is a boolean formulae with n variables. So,

$$\mathbf{TQBF}=\{\psi=1|\psi=Q_1x_1Q_2x_2\cdots Q_nx_n\varphi(x_1,x_2,\cdots,x_n)\}$$

First, we show that $\mathbf{TQBF}\in\mathbf{PSPACE}$. Let

$$\psi=Q_1x_1Q_2x_2\cdots Q_nx_n\varphi(x_1,x_2,\cdots,x_n)$$

be a quantified boolean formulae with n variables and the size of φ is m. Obviously, if all the variables are assigned then we need only $O(m)$ space to decide it. Now we will construct a recursive algorithm to decide ψ. We will use notation $\psi|_{x_i=b}$ to denote that replacing all the occurrences of variable x_i with $b\in\{0,1\}$ with dropping quantifier Q_i.
Algorithm will work as follows:

1. if $Q_i=\exists$ then output 1 if and only if at least one of $\psi|_{x_i=0}$ and $\psi|x_i=1$ returns 1.
2. if $Q_i=\forall$ then output 1 if and only if $\psi|_{x_i=0}$ and $\psi|x_i=1$ both return 1.

Obviously, the recursive algorithm conform to the definition of $\exists$ and $\forall$, so it does indeed return the correct answer on any formula ψ.
Whenever we drop a quantifier, we need copy the formula φ once (we can reuse the space to decide $\psi|_{x_i=1}$ and $\psi|_{x_i=0}$).
Let $s_{n,m}$ denote the space of the recursive algorithm uses on ψ with n variables. We can reduce one variable when we drop a quantifier, so

$$s_{n,m}=s_{n-1,m}+O(m)$$

and we can get $s_{n,m}=O(n\cdot m)$, obviously, $n<m$ and we can say that the QBF ψ can be decided in $O(m^2)$ space.
So $\mathbf{TQBF}\in\mathbf{PSPACE}$.

Now we will show that for every $L\in\mathbf{PSPACE}$, $L\leqslant_p\mathbf{TQBF}$. Let M be a TM that can decides L in $S(n)$ space and let $x\in\{0,1\}^n$. First, we show that there is a Boolean formula $\varphi_{M,x}$ such that for every two strings C,C' which

encode the configuration of M, and $\varphi_{M,x}(C, C') = 1$ if and only if C and C' are valid adjacent configurations in the configuration graph of $G_{M,x}$.
We can consider all the possibilities of configurations, the number if $2^{O(S(n))}$, it is too large to be express in CNF or DNF. We can group them by the head position, i.e. we can group them to $O(S(n))$ groups like

C	C'	$\varphi_{M,x}(C, C')$
$\cdots - - - * - - - \cdots$	$\cdots - * - - - - - - - \cdots$	0
$\cdots - - - * - - - \cdots$	$\cdots - - * - - - - - - \cdots$	1
$\vdots$	$\vdots$	$\vdots$
$\cdots - - - - * - - \cdots$	$\cdots - - - - - - - * \cdots$	0
$\cdots - - - - * - - \cdots$	$\cdots - - - - - * - - \cdots$	1
$\vdots$	$\vdots$	$\vdots$
$\vdots$	$\vdots$	$\vdots$

Figure 1: Table of φ,"-" represents a work tape cell and "*" represents the head position.

The first two columns are all possible configurations of M and the third column is the result of φ. For every group we have to decide the all the possible cases, so the Table has $2^{2^{O(S(n))}}$ rows. In every group there will be few rows such that $\varphi = 1$, because M can only move at most 1 step and modify at most 1 cell. In fact, there are at most 6 rows such that $\varphi = 1$, because the head will move to left or right or stay, and the cell can be modified or not, they are at most 6 different cases.
The 6 different cases involve only 1 variable for the work cell, constant number of variables for states which decided by M. So for every group we can build a CNF with constant size, and the for the whole configurations we can get a CNF φ with $O(S(n))$ size.
Assume that C need m space. Now we will use $\varphi_{M,x}$ to come up with a polynomial space quantified boolean formula ψ' such that for every $C, C' \in \{0,1\}^m$, $\psi'(C, C') = 1 \Leftrightarrow$ there is a directed path from C to C' in $G_{M,x}$. Let $C = C_{start}$ and $C' = C_{accept}$, will get a quantified boolean formula $\psi = 1 \Leftrightarrow M$ accepts x. We will define ψ inductively.
Let $\psi_i(C, C')$ be ture iff there is a path of length which does not exceed 2^i from C to C' in $G_{M,x}$. We have $\psi' = \psi_m$ and $\psi_0 = \varphi_{M,x}$. We can know that if $\psi_i = 1$ if and only if there is a configuration C'' such that $\psi_{i-1} = 1$, so we get

$$\psi_i(C, C') = \exists C'' \psi_{i-1}(C, C') \wedge \psi_{i-1}(C'', C).$$

But there occur two ψ_{i-1}, and the size of ψ_m will be 2^m, it is not polynomial, so we have to transform it to an equivalent form. Consider

$$\psi_i = \exists C'' \forall D_1 \forall D_2 \big((D_1 = C \wedge D_2 = C') \vee (D_1 = C' \wedge D_2 = C'')\big) \rightarrow \psi_{i-1}(D_1, D_2).$$

We can easily transform $\to$ to $\wedge, \vee, \neg$, so this form has only one ψ_{i-1}.

$$size(\psi_i) = size(\psi_{i-1}) + O(m)$$

and then

$$size(\psi) = size(\psi_m) = O(m \cdot m) = O(m^2).$$

So we get a QBF with size $O(m^2)$, and the original problem has been reduced to **TQBF**.
So **TQBF** is **PSPACE**-complete.

Problem 2. Prove that the problem satisfiable formula in CNF can reduce by Karp to problem Integer Linear Programming (under a given system of linear inequalities with integer coefficients, find out whether it has a solution in integers).

Solution:

Assume that the CNF has form:

$$\varphi(x_1, x_2, \cdots, x_n) = \bigwedge_i \left(\bigvee_j v_{i_j} \right),$$

where v_{i_j} is some variable x_k or its negation $\neg x_k$. It is easily to express a CNF formula to an integer program:

1. First we add the constraints $0 \leqslant x_i \leqslant 1$ for every i to ensure that the variables can only be assigned with 0 or 1.

2. Then we will express every clause of CNF to an integer inequality. for any variables x_i and x_j

$$\begin{gathered} x_i \vee x_j \Leftrightarrow x_i + x_j \geqslant 1. \\ \neg x_i \vee x_j \Leftrightarrow (1 - x_i) + x_j \geqslant 1. \\ x_i \vee \neg x_j \Leftrightarrow x_i + (1 - x_j) \geqslant 1. \\ \neg x_i \vee \neg x_j \Leftrightarrow (1 - x_i) + (1 - x_j) \geqslant 1. \end{gathered}$$

   So for any number of variables connected by $\vee$ we can get the inequality inductively, because for any formula which are connected by $\vee$ obey the same laws.

3. Then we collect all the clauses and get a system of inequalities, because all the clauses are connected by $\wedge$, and it means that the system has to satisfy all the clause.

We have reduced the SAT problem to Integer Programming.

Obviously, we can valid whether a given vector is a solution to the system in polynomial time, so Integer Programming is in **NP**. And the reduction indicates that SAT $\leqslant_p$ IP, so IP is NP-complete.