Due: Wednesday, January 30

1. SUBSET SUM is the problem: Given a list of $n$ numbers $A_1, \ldots, A_n$ and a number $T$ (all presented in binary), determine whether there is a subset $S \subseteq [n]$ such that

$$\sum_{i \in S} A_i = T$$

PARTITION is similar to SUBSET SUM except we are not given $T$, and we are to determine whether there is $S \subseteq [n]$ such that

$$\sum_{i \in S} A_i = \sum_{i \notin S} A_i$$

Give an explicit reduction showing SUBSET SUM $\leq_p$ PARTITION (so PARTITION is also NP complete).

Solution:

Given an instance $\langle A_1, \ldots, A_n, T \rangle$ of SUBSET SUM we are to compute (in polynomial time) an instance $\langle B_1, \ldots, B_m \rangle$ of PARTITION such that

$$\exists S \subseteq [n] \sum_{i \in S} A_i = T \iff \exists S' \subseteq [m] \sum_{i \in S'} B_i = \sum_{i \notin S'} B_i \tag{1}$$

Let $A = \sum_{i \in [n]} A_i$.

Remark: Normally the subset sum and partition problems assume that the input numbers $A_1, \ldots, A_n$ are non-negative integers. This is what we assume in the solution below. However if we allowed negative integers, then Case I below suffices – we do not need Case II.

Case I: $2T \geq A$

Let $m = n + 1$, let $B_i = A_i, 1 \leq i \leq n$, and let $B_{n+1} = 2T - A$.

Correctness: We must establish the equivalence (1).

($\Rightarrow$): Assume that the index set $S$ is chosen to satisfy the left equation in (1). If we set $S' = S$ then the right equation is also satisfied, since $(A - T) + (2T - A) = T$.

($\Leftarrow$): Assume that the index set $S'$ satisfies the right equation. Let $S = S'$ if $n + 1 \notin S'$, otherwise let $S = ([n + 1] \setminus S')$. Then $S$ satisfies the left equation.

Case II: $2T < A$

Let $m = n + 1$, let $B_i = A_i, 1 \leq i \leq n$, and let $B_{n+1} = A - 2T$.

Correctness: We must establish the equivalence (1).

($\Rightarrow$): Assume that the index set $S$ is chosen to satisfy the left equation in (1). If we set $S' = S \cup \{n + 1\}$ then the right equation is also satisfied, since $T + (A - 2T) = A - T$.
Assume that the index set \( S' \) satisfies the right equation, so each side sums to \( A - T \). Let \( S = ([n] \setminus S') \) if \( n + 1 \not\in S' \), otherwise let \( S = S' \setminus \{n + 1\} \). Then \( S \) satisfies the left equation, since \( (A - T) - (A - 2T) = T \).

2. Given an undirected graph \( G = (V, E) \), let us say that a set \( V' \subseteq V \) of vertices is very independent if the distance between any two distinct vertices in \( V' \) is at least 3. Define \( VIS = \{\langle G, k \rangle \mid G \text{ is an undirected graph with a very independent set of size } k\} \)

Prove that \( VIS \) is \( NP \) complete.

Solution:
It is easy to see that \( VIS \) is in \( NP \): A certificate showing that \( \langle G, k \rangle \) is a YES instance is a subset \( V' \subseteq V \) such that \( V' \) is a very independent set of size \( k \).

We show \( VIS \) is \( NP \)-hard by showing \( INDSET \leq_p VIS \).

Given an instance \( \langle G, k \rangle \) of \( INDSET \) we want to construct (in polynomial time) an instance \( \langle \hat{G}, \hat{k} \rangle \) such that \( G \) has an independent set of size \( k \) iff \( \hat{G} \) has a very independent set of size \( \hat{k} \).

Suppose \( G = (V, E) \), where the edge set is \( E = \{(u_1, v_1), \ldots, (u_m, v_m)\} \).

To construct \( \hat{G} \), we include all vertices in \( G \), and in addition we introduce a new vertex \( w_i \) for each edge \( e_i \) in \( G \). We replace each edge \( (u_i, v_i) \) in \( G \) by the two edges \( (u_i, w_i) \) and \( (w_i, v_i) \). Finally we need two new vertices \( a, b \), and an edge \( (a, b) \), and an edge \( (w_i, a) \) for \( 1 \leq i \leq m \).

Let \( \hat{k} = k + 1 \).

Correctness:
Suppose \( V' \) is an independent set of \( G \) of size \( k \). It is easy to see that \( V' \cup \{b\} \) is a very independent set of \( \hat{G} \) of size \( k + 1 \).

Conversely, suppose \( U' \) is a very independent set of \( \hat{G} \) of size \( k + 1 \).

Case I: \( U' \) contains no vertex of the form \( w_i \).
Since \( U' \) cannot contain both \( a \) and \( b \), it follows that \( U' \) contains at least \( k \) vertices in the original graph \( G \). These vertices form an independent set in \( G \).

Case II: \( U' \) contains some vertex \( w_i \). Then \( U' \) does not contain \( w_j \) for \( j \neq i \), because the path \( w_i, a, w_j \) has length 2. Also \( U' \) contains neither \( a \) nor \( b \). Hence again \( U' \) contains at least \( k \) vertices in the original graph \( G \), and these form an independent set in \( G \).

3. SET PACKING is the following \( NP \)-complete problem: Given sets \( T_1, \ldots, T_m \), where \( T_i \subseteq \{1, \ldots, n\}, 1 \leq i \leq m \), and given \( k \leq m \), determine if there is a subcollection consisting of \( k \) of the \( T_i \) which are pairwise disjoint. (Assume that each \( T_i \) is presented by a bit string of length \( n \).)

Give an explicit reduction showing \( SET PACKING \leq_p SAT \).

Solution:
Given an instance \( I \) of SET PACKING as described above, we must construct (in polynomial time) a Boolean formula \( \varphi \) such that \( I \) is a YES instance iff \( \varphi \) is satisfiable.
\( \varphi \) has variables \( p_{ij}, 1 \leq i \leq n, 1 \leq j \leq k \). The intended meaning of \( p_{ij} \) is that \( T_i \) is the \( j \)th member of the subcollection.

Then \( \varphi \) is the conjunction of the following formulas:

\[ A_j = (p_{1j} \lor \cdots \lor p_{nj}), 1 \leq j \leq k. \]

\( A_j \) asserts that some \( T_i \) is the \( j \)th member of the subcollection.

\[ B_{ijj'} = (\overline{p_{ij}} \lor \overline{p_{ij'}}), 1 \leq i \leq n, 1 \leq j < j' \leq k. \]

\( B_{ijj'} \) asserts that \( T_i \) cannot be both the \( j \)th member and the \( j' \)th member of the subcollection.

\[ C_{ii'} = (\overline{p_{i1}} \land \cdots \land \overline{p_{ik}}) \lor (\overline{p'_{i1}} \land \cdots \land \overline{p'_{ik}}), \text{ provided } i < i' \text{ and } T_i \cap T_{i'} \neq \emptyset. \]

\( C_{ii'} \) asserts that if \( T_i \) and \( T_{i'} \) are not disjoint, then they cannot both be in the subcollection.

**CORRECTNESS:**

(\( \Rightarrow \)): Assume that \( I \) is a YES instance of SET PACKING. Then there are pairwise disjoint sets \( T_{i_1}, \ldots, T_{i_k} \). Let \( \tau(p_{ij}) = 1 \text{ iff } i = i_j \). Then it is easy to check that the assignment \( \tau \) satisfies all of the above formulas.

(\( \Leftarrow \)):

Let \( \tau \) an assignment which satisfies all of the above formulas. Then by the \( A \)'s and \( B \)'s there are distinct number \( i_1, \ldots, i_k \) in \([n]\) such that \( \tau(p_{i_j}) = 1, 1 \leq j \leq k \). Then by the formulas \( C \), the collection \( T_{i_1}, \ldots, T_{i_k} \) is pairwise disjoint.

4. An NP search problem is given by a polynomial time relation \( R(x, y) \) over \( \{0,1\}^* \times \{0,1\}^* \) and a polynomial \( p(n) \). The problem is: Given \( x \), find \( y \) such that \( |y| \leq p(|x|) \) and \( R(x, y) \), or output ‘NO’ if no such \( y \) exists.

The corresponding NP decision problem is: Given \( x \), determine if such a \( y \) exists.

Here we are interested in polynomial time oracle reductions (see 2.14 on page 65 of the text). We use the notation \( P_1 \xrightarrow{p} P_2 \) to mean problem \( P_1 \) is polynomial time oracle reducible (or ‘Cook reducible’) to \( P_2 \), defined at the bottom of page 8 in the CSC 364 notes *Turing Machines and Reductions*, available on the CSC 2401S web page.

If \( A \)-search is the NP search problem corresponding to the NP decision problem \( A \), then obviously \( A \xrightarrow{p} A \)-search.

The reverse reduction may not be true in general, but if \( A \) is NP-complete, then in fact \( A \)-search \( \xrightarrow{p} A \), so \( A \) and \( A \)-search are p-equivalent.

We say an NP search problem if total if every instance has a solution.

An example of a total NP search problem not known to be solvable in polynomial time is prime factorization: Given an integer \( n \geq 2 \) in binary, find a list of prime numbers whose product is \( n \). (Prime numbers can be recognized in polynomial time.)

A second example is finding a Nash Equilibrium in game theory.

Note that the decision problem associated with a total search problem is trivial, because the answer is always YES. However you are to show that every NP search problem is reducible to some NP decision problem, but probably not equivalent to any NP complete decision problem.
(a) Show that for every \textbf{NP} search problem \( SS \) there is an \textbf{NP} decision problem \( A_{SS} \) such that \( SS \xrightarrow{p} A_{SS} \).

(See the solution to Question 5, PS 3, for CSC 463H, Fall 2012: http://www.cs.toronto.edu/~sacook/csc463h/)

**Solution:**

Let \( R(x, y) \) and \( p(n) \) be the polytime relation and polynomial, respectively, in the definition showing that \( SS \) is an \textbf{NP} search problem.

Define the \textbf{NP} decision problem \( A \) by

\[
A = \{ \langle x, y \rangle \mid \exists z (|yz| \leq p(|x|) \land R(x, yz)) \}
\]

Clearly \( A \in \textbf{NP} \), since the \( z \) serves as the witness.

Here is the reduction showing that \( SS \) is polynomial time reducible to \( A \):

The input is \( x \) (\( \epsilon \) is the empty string) (The idea is to start with \( y = \epsilon \) and use \( A \) to extend \( y \) one bit at a time until \( R(x, y) \) holds.)

\[
\begin{array}{l}
\text{if } \langle x, \epsilon \rangle \notin A \text{ then the Output ‘NO’, exit.} \\
y \leftarrow \epsilon \\
\text{for } i : 1..p(|x|) \quad (*) \\
\quad \text{if } R(x, y) \text{ then Output } y, \text{ exit.} \\
\quad \text{if } \langle x, y0 \rangle \in A \text{ then } y \leftarrow y0 \\
\quad \text{else } y \leftarrow y1 \\
\end{array}
\]

(*) LOOP INVARIANT: \( \langle x, y \rangle \in A \).

Hence \( R(x, y) \) must hold by the time \( y \) has reached length \( p(|x|) \).

(b) Show that if \( A \) is a decision problem (i.e. \( A \subseteq \{0,1\}^* \)) and \( A \xrightarrow{p} TS \) for some total \textbf{NP} search problem \( TS \), then \( A \in \textbf{NP} \cap \text{coNP} \).

**Solution:**

Assume \( A \xrightarrow{p} TS \) for some total \textbf{NP} search problem \( TS \), where \( A \subseteq \{0,1\}^* \). Then there is a polytime oracle TM \( M \) which on input \( x \), makes queries to \( TS \) and determines whether \( x \in A \). The computation of \( M \) on input \( x \) is coded by a string \( C_x \) of length polynomial in \( |x| \), where in particular \( C_x \) codes the answers to all queries made to \( TS \) during the computation. (Since \( M \) operates in polytime, all queries are bounded in length by a polynomial in \( |x| \), and since \( TS \) is a total \textbf{NP} search problem, all answers to these queries must be bounded by a polynomial in the query, and hence by a polynomial in \( |x| \).)

The polytime verifying relation \( R_1 \) showing that \( A \in \textbf{NP} \) is defined by the condition that \( R_1(x, y) \) holds iff \( y = C_x \), where \( C_x \) codes an accepting computation of \( M \) on input \( x \). Note that this can be verified in polytime, since all answers to queries to \( TS \) can be verified in polytime, since \( TS \) is an \textbf{NP} search problem.
Similarly $R_2(x, y)$ is a verifying relation showing $\overline{A} \in \text{NP}$, where now $y$ codes a rejecting computation of $M$ on input $x$.

(c) Conclude that if there exists an \text{NP}-complete decision problem $A$ which is $p$-equivalent to some total \text{NP} search problem $TS$, then $\text{NP} = \text{coNP}$. (Here $p$-equivalent means $A \xrightarrow{p} TS$ and $TS \xrightarrow{p} A$.)

(There is evidence that some total \text{NP} search problems, such as Nash Equilibrium, are not $p$-equivalent to any decision problem.)

\textbf{Solution:}
Suppose that $A$ is \text{NP}-complete, and $A$ is $p$-reducible to a total \text{NP} search problem $TS$. Then by part (b), $A \in \text{NP} \cap \text{coNP}$.

It suffices to show that $\text{NP} \subseteq \text{coNP}$, since then if $A \in \text{coNP}$, it follows that $\overline{A} \in \text{NP}$, so by assumption $\overline{A} \in \text{coNP}$, so $A \in \text{NP}$. Thus $\text{coNP} \subseteq \text{NP}$.

To show $\text{NP} \subseteq \text{coNP}$ we need two simple properties of $\leq_p$:
1. If $L_1 \leq_p L_2$ and $L_2 \in \text{NP}$ then $L_1 \in \text{NP}$.
2. If $L_1 \leq_p L_2$ then the same reduction shows $\overline{L_1} \leq_p \overline{L_2}$.

Let $B$ be any problem in \text{NP}. Since $A$ is \text{NP}-complete, it follows that $B \leq_p A$, so by 2. above, $\overline{B} \leq_p \overline{A}$. Since $A \in \text{coNP}$, it follows that $\overline{A} \in \text{NP}$. Hence by 1. above, $\overline{B} \in \text{NP}$. Hence $B \in \text{coNP}$. 

5