Computational Models, Spring 2014 Exrecise #6

Turing Machines, $\mathcal{P}$ and $\mathcal{NP}$

1. Show that the following languages are NP-Complete

   (a) $Partition := \{x_1, \dots x_n \mid \exists S \subseteq [1 \dots n], \sum_{i \in S} x_i = \sum_{i \notin S} x_i\}$
   Solution We show by a reduction SubsetSum $\leq_P$ Partition:
   Given $x_1, \dots x_n$, let us denote $\sum_{i=1 \dots n} x_i = k$.
   $f(x_1, \dots x_n, t) = x_1, \dots x_n, t + k, 2k - t$

   (b) $XS := \{x_1, \dots x_n \mid \exists S \subseteq [1 \dots n], \sum_{i \in S} x_i + |\bar{S}| = \sum_{i \in \bar{S}} x_i + |S|\}$
   Solution We show by a reduction Partition $\leq_P$ XS:
   Note that:
   $\sum_{i \in S} x_i + |\bar{S}| = \sum_{i \in \bar{S}} x_i + |S|$
   $\Leftrightarrow$
   $\sum_{i \in S} x_i - |S| = \sum_{i \in \bar{S}} x_i - |\bar{S}|$
   $\Leftrightarrow$
   $\sum_{i \in S}(x_i - 1) = \sum_{i \in \bar{S}}(x_i - 1)$

   Thus our reduction is:
   $f(x_1, \dots x_n) = (x_1 - 1), \dots (x_n - 1)$

2. Show that both of these decision problems are in $\mathcal{P}$

   (a) DNF-Satisfability: Given a DNF formula is it satisfiable.
   Solution A disjunction is the boolean OR of its clauses, and so it is satisfiable if any of its clauses can be satisfied. A clause is the boolean AND of some number of literal terms. Any clause is satisfiable unless it contains both some literal x and the negation of that literal $\bar{x}$. Thus an algorithm to determine DNF-Satisfiability would loop through the clauses and check if there is a clause which is satisfiable. This is clearly in $\mathcal{P}$.

   (b) CNF-Tautology: Given a CNF formula, is it a tautology (a tautology is a formula which is true in every possible assignment)?
   Solution There are two methods of proving this:

   - A CNF formula is the boolean AND of its clauses. Thus it is a tautology if and only if every clause is a tautology. A clause is the boolean OR of some number of literals. Thus a clause is only a tautology if and only if it has some literal x and the negation of that literal $\bar{x}$. This can easily be checked by scanning through the clauses, and so CNF-Tautology is in $\mathcal{P}$.
   - A more clever method of proving this is to note that by DeMorgan's law the negation of a DNF formula is a CNF formula. Further, note that if X is a DNF tautology then its negation is a CNF contradiction, i.e. it is unsatisfiable. Thus we can solve this problem by negating the input and using our solution to the previous problem.

3. Consider the following algorithm to solve the vertex cover problem. First, we generate all size-k subsets of the vertices. There are $O(n^k)$ of them. Then we check whether any of the resulting subgraphs is complete. Why is this not a polynomial-time algorithm?

   Solution For it to be a polynomial time algorithm, it must have runtime which is polynomial for any input. However k and n both depend on the input, so n^k is not a polynomial. For example, consider the input (G = (V,E), n) where —V— = n in this case the algorithm generates $O(n^n)$ sets.

4. For the following decision problems, determine whether they are in $\mathcal{P}$ or in $\mathcal{NPC}$ (assuming $\mathcal{P} \neq \mathcal{NP}$). Prove your answer.

(a) Input: sets $A_1 \dots A_n$, and a number k.
Question: does there exist a set C of size k, such that for every $1 \leq i \leq n$ $A_i \cap C \neq \emptyset$?

NPC. Reduction from VC. Given input G = (V, E), k for VC, we construct a set A_i for every $\{v_1v_2\} \in$E and let $A_i = \{v_1v_2\}$ We keep the same k.

(b) Input: a 3CNF formula ψ Question: does there exist an assignment that satisfies ψ and gives **True** for exactly 10 variables?

P. We check all possible sets of 10 variables.

(c) Input: a 3CNF formula ψ
Question: do there exist at least two assignments that satisfy ψ?

NPC. Reduction from 3SAT. Given input ψ , we add a clause $(z \vee \bar{z} \vee \bar{z})$ where z is a new variable.

(d) Input: graph G.
Question: does there exist a Hamiltonian path in G (between any pair of vertices)?

Reduction from UHAMPATH to Hamiltonian path. Given input G, s, t we construct a graph G' by adding two new vertices u_1, u_2 to G and connecting u_1 to s and u_2 to t.

(e) Input: graph G and a number k.
Question: does there exist a simple path in G of length $\geq k$?

NPC. Reduction from previous item. Given input $G = (V, E)$, we construct the input $G, |V| - 1$.

(f) Input: graph G and a number k.
Question: is there a Vertex-Cover S in G of size k and an IndependentSet, T, of size $\frac{k}{2}$, such that $T \subseteq S$?

NPC. Reduction from VC. Given input G, k for VC, we construct input G', k' for our problem. G' is obtained from G by adding it a distinct simple cycle of length 2k. k' = 2k.

5. Prove that if $\mathcal{P} = \mathcal{NP}$ then every language in $\mathcal{P}$, except $\emptyset$ and Σ^* is $\mathcal{NPC}$. Why can't $\emptyset$ and Σ^* be $\mathcal{NPC}$.

Let A be any language in NP and let B be another language not equal to $\emptyset$ or Σ^*. Then there exist strings $x \in B$ and $y \notin B$. To reduce an instance w of A to that of B, we just check in polynomial time if $w \in A$. If yes, we output x and y when $w \notin A$.
The languages $\emptyset$ and Σ^* cannot be NP-complete, because to reduce a language A to a language B, we need to map instances in A to instances in B and those outside A to outside B. However, for $B = \emptyset$, there are no instances in B (and none outside B for $B = \Sigma^*$) which means there cannot be such a reduction from any language $A \neq \emptyset, \sigma^*$.

6. Consider the following language:

$$L = \{\#1^n\#x_1, \dots x_n \mid \exists i, j \ s.t. x_i = x_j\}$$

. A possible way to implement a Turing Machine that accepts the language is as follows: "Choose (non-deterministically) two indices i and j $(0 < i, j \leq n)$ and write them on a second tape. Now check (using a deterministic TM) if (1) $i \neq j$ and (2) $x_i = x_j$."

In this exercise we will *formally* implement the first part: On a *two-taped, non-deterministic* Turing Machine with input $\#1^n\#$ $(n > 1)$, choose non-deterministically i and j such that $0 < i, j \leq n$ and write on the second tape $1^i\#1^j$ and accept.

If this helps you, you may use a model that allows the head to stay put as well as moving right and left.

Solution We will use a two taped, non-deterministic TM that allows the head to stay put as well as moving right and left.

- $\delta(q_0, (\#, \sqcup)) = \{(q_{i_1}, (\#, \sqcup), (R, S))\}$
  start writing 1^i
- $\delta(q_{i_1}, (1, \sqcup)) = \{(q_{i_2}, (1, 1), (R, R))\}$
  ensure that $i > 0$
- $\delta(q_{i_2}, (1, \sqcup)) = \{(q_{i_2}, (1, 1), (R, R)), (q_{i_2}, (1, \sqcup), (R, S))\}$
  non-deterministically choose if to write 1 or not on the second tape
- $\delta(q_{i_2}, (\sqcup, \sqcup)) = \{(q_L, (\sqcup, \#), (L, R))\}$
  mark end of 1^i
- $\delta(q_L, (1, \sqcup)) = \{(q_L, (1, \sqcup), (L, S))\}$
  go left
- $\delta(q_L, (\#, \sqcup)) = \{(q_{j_1}, (\#, 1), (R, R))\}$
  start writing q^j
- $\delta(q_{j_1}, (1, \sqcup)) = \{(q_{j_2}, (1, 1), (R, R))\}$
  ensure that $i > 0$
- $\delta(q_{j_2}, (1, \sqcup)) = \{(q_{j_2}, (1, 1), (R, R)), (q_{j_2}, (1, \sqcup), (R, S))\}$
  non-deterministically choose if to write 1 or not on the second tape
- $\delta(q_j, (\sqcup, \sqcup)) = \{(q_a, (\sqcup, \sqcup), (S, S))\}$
  terminate

7. We wish to understand the *encoding* of graphs

   (a) Suggest an efficient way to encode an undirected graph $G = (V, E)$ as a string $< G >$. Describe the complexity of your encoding as a function of edges and vertices. Draw several graphs and demonstrate the encoding on the graphs.

   (b) Write a program (python or scheme) that takes as an input an encoding $< G >$ of a graph and an integer k and returns true if the sum of the degrees of the vertices equals to k.