COMS 4236: Computational Complexity (Fall 2018)
Problem Set #2

Wenbo Gao - wg2313@columbia.edu

December 8, 2018

Problem 1

(a)

Problem. Exercises 7.7 on page 271 and Problem 7.14 on page 272 of TC (6 points): Show that NP is closed under union ($\cup$), concatenation ($<>$), and star operation ($*$). (Xi: You may want to use the alternative view of NP as languages decided by polynomial-time verifiers. Can you also show that P is closed under the star operation?)

Solution.

- NP is closed under union ($\cup$).

  Goal. Given $L_1 \in$ NP, $L_2 \in$ NP. Show $L_1 \cup L_2 \in$ NP.

  Proof.

  $$L_1 \in \text{NP} \Rightarrow \exists D_1 \in \text{all NTMs}, L(D_1) = L_1$$
  $$L_2 \in \text{NP} \Rightarrow \exists D_2 \in \text{all NTMs}, L(D_2) = L_2$$

  Construct a NTM, $D(w)$: $\forall w \in \Sigma^*$,

  1. Non-deterministically guess if $w \in L_1$, or $w \in L_2$, or $w \notin L_1 \wedge w \notin L_2$.
  2. If the guess is
     - $w \in L_1$, then simulate D_1 on w and return the result.
     - $w \in L_2$, then simulate D_2 on w and return the result.
     - $w \notin L_1 \wedge w \notin L_2$, then reject

  $\forall w \in L_1 \cup L_2$, then $w \in L_1 \vee w \in L_2$, there is a branch in D that accepts.

  $\forall w \notin L_1 \cup L_2$, then $w \notin L_1 \wedge w \notin L_2$, all branches in D rejects:

  - branch1: $D_1(w) = 0$
  - branch2: $D_2(w) = 0$

– branch3: rejects

Therefore, $L(D) = L_1 \cup L_2$ where D is a NTM, so $L_1 \cup L_2 \in$ NP.

□

- NP is closed under concatenation (<>).

  Goal. Given $L_1 \in$ NP, $L_2 \in$ NP. Show $L_1 <> L_2 \in$ NP.

  Proof.

$$L_1 \in \text{NP} \Rightarrow \exists D_1 \in \text{all NTMs}, L(D_1) = L_1$$
$$L_2 \in \text{NP} \Rightarrow \exists D_2 \in \text{all NTMs}, L(D_2) = L_2$$

  Construct a NTM, $D(w)$: $\forall w \in \Sigma^*$ where $|w| = n$,

  1. Non-deterministically guess a number $n_1 \in \{0, 1, \ldots, n\}$
  2. Let w_1 be the first n_1 characters of w, and w_2 be the rest $(n - n_1)$ characters. Simulate D_1 on w_1 and D_2 on w_2, and return $D_1(w_1) \wedge D_2(w_2)$.

  $\forall w \in L_1 <> L_2$, there is a branch in D that accepts where the choice of n_1 is exactly the length of $w_1 \in L_1$ and $(n-n_1)$ is exactly the length of $w_2 \in L_2$ and $w = w_1 <> w_2$.

  $\forall w \notin L_1 <> L_2$, all branches in D rejects since there's no such a partition of w into w_1 and w_2 where $D_1(w_1) = 1$ and $D_2(w_2) = 1$.

  Therefore, $L(D) = L_1 <> L_2$ where D is a NTM, so $L_1 <> L_2 \in$ NP. □

- NP is closed under star operation ($*$).

  Goal. Given $L_1 \in$ NP. Show $L_1^* \in$ NP.

  Proof.

$$L_1 \in \text{NP} \Rightarrow \exists D_1 \in \text{all NTMs}, L(D_1) = L_1$$

  Construct a NTM, $D(w)$: $\forall w \in \Sigma^*$ where $|w| = n$,
  Non-deterministically guess a number $k \in \{0, 1, \ldots, n\}$

  – if $n \not\equiv 0 (\text{mod } k)$, then reject
  – if $n \equiv 0 (\text{mod } k)$, then let x be the first k characters of w.
  Simulate D_1 on x and return $D_1(x)$.

  $\forall w \in L_1^*$, there is a branch in D that accepts.

  $\forall w \notin L_1^*$, all branches in D rejects.

  Therefore, $L(D) = L_1^*$ where D is a NTM, so $L_1^* \in$ NP. □

(b)

Problem. Exercise 7.11 on page 272 of TC (4 points): Call graph G and H isomorphic if the nodes of G may be reordered so that it is identical to H. Let

$$\text{ISO} = \{(G, H) \mid G \text{ and } H \text{ are isomorphic graphs}\}.$$

Show that ISO $\in$ NP.

Proof. Construct a NTM, $D(w)$: $\forall G(V_G, E_G), H(V_H, E_H) \in$ all graphs,

- If $|V_G| \neq |V_H| \vee |E_G| \neq |E_H|$, reject
- Otherwise, let $|V_G| = |V_H| = n$.
    1. Enumerate all vertices in V_G, and non-deterministically guess a new assignment of index $\in \{1, 2, \ldots, n\}$ to each vertex. (Here I assume vertices in G and H are indexed by numbers from $\{1, 2, \ldots, n\}$)
    2. Once we have the reordered graph V'_G, compare V'_G and H by vertices and edges. If V'_G and H are identical, accept; otherwise, reject.

$\forall (G, H) \in ISO$, there is a branch in D that accepts.
$\forall (G, H) \notin ISO$, all branches in D rejects.
Therefore, $L(D) =$ ISO where D is a NTM, so ISO $\in$ NP. $\square$

Problem 2

(a)

Problem. Problem 8.20 on page 304 of TC (10 points): An undirected graph is bipartite if its nodes may be divided into two sets so that all edges go from a node in one set to a node in the other set. Show that a graph is bipartite if and only if it doesn't contain a cycle that has an odd number of nodes.

Goal. Prove $G(V, E)$ is a bipartite graph $\Leftrightarrow \forall p \subseteq E$, if p is a cycle, then $|p|$ is even.

Proof.

- LHS $\Rightarrow$ RHS
  A cycle is a closed path which starts and ends at the same vertex. Thus, in a bipartite graph, a cycle starts and ends at the same side. In order to get back to the same side, it takes even-number steps, i.e. Left→Right, and then Right→Left, because there's no edge within the same side. So all cycles in a bipartite graph are of even length. Since cycles have the same number of edges as the number of vertices, all cycles in a bipartite graph have even number of vertices.
- LHS $\Leftarrow$ RHS Pick a vertex $v_0 \in V$. Partition V into two sets:
  1. vertices of even-length distance from v_0, including v_0 itself
  2. vertices of odd-length distance from v_0

  Proof by contradiction:
  Assume there exists edges in the same set, then there exists an odd cycle containing v_0 and there is a vertex in the same set as v_0 but of odd-length distance to v_0, which leads to a contradiction. Therefore, there's no edges in the same set and thus this graph is a bipartite graph.

□

(b)

Problem. Let

$$\text{BIPARTITE} = \{G \mid G \text{ is a bipartite graph}\}.$$

Show that BIPARTITE $\in$ NL. (Xi: What is the more natural complexity class to which BIPARTITE belongs? as suggested by the fact mentioned, and what do we know about this class?)

Problem 3

Problem. Show that P $\neq$ SPACE(n). (Hint: Assume P = SPACE(n). Use the space hierarchy theorem to derive a contradition. You may find the function pad, defined in Problem 9.18, to be helpful.)

Problem 4

Problem. Problem 8.15 on page 303 of TC (5 points if you can show the following problem is in PSPACE; 10 points if you can show if it is in P!): The cat-and-mouse game is played by two players, "CAT" and "Mouse", on an arbitrary undirected graph. At a given point each player occupies a node of the graph. The players take turns moving to a node adjacent to the one that they currently occupy. A special node of the graph is called "Hole". Terminal conditions of the game:

- **Cat wins** if the two players ever occupy the same node.
- **Mouse wins** if it reaches the Hole before the preceding happens.
- The game is a **draw** if the two players ever simultaneously reach positions that they previously occupied.

Let

$$\begin{aligned}\text{HAPPY-CAT} = \{(G, c, m, h) \mid & \text{ G,c,m,h are respectively a graph, and} \\ & \text{positions of the Cat, Mouse, and Hole, such that} \\ & \text{Cat has a wining strategy, if Cat moves first.}\}\end{aligned}$$

Show that HAPPY-CAT is in PSPACE (and in P for more points).

Proof. HAPPY-CAT is in P.
Given undirected graph $G(V, E)$, where $|V| = n$.
The state space of the game has $2n^2$ vertices:

$$\begin{aligned}&\text{State} = \\ &\quad \{ \quad \text{Cat\_position} :: V \\ &\quad , \quad \text{Mouse\_position} :: V \\ &\quad , \quad \text{turn\_player} :: \{\text{Cat}, \text{Mouse}\} \\ &\quad \}\end{aligned}$$

$$\#\text{ of states} = |V| * |V| * 2 = n * n * 2 = 2n^2$$

The edges between states are valid state transitions, the number of which is at most $8n^4$.

Each vertex takes $log_2 n$ bits and each edge takes $2log_2 n$ bits, so the entire state graph takes $(2n^2 log_2 n + 16n^4 log_2 n)$ bits which is $O(n^5)$.

Annotate the states by Cat's win condition:

$$\forall V_0 \in V, (\{V_0, V_0, \text{Cat}\}, \text{Cat\_Win}) \text{ and } (\{V_0, V_0, \text{Mouse}\}, \text{Cat\_Win})$$

HAPPY-CAT then can be formalized as a reachability problem:
Given Cat's initial position $V_{Cat\_init} \in V$, Mouse's initial position $V_{Mouse\_init} \in V$, and initial turn player Cat, whether there's a state annotated by Cat_Win can be reached from this initial state.

We can use BFS traversing the state graph backwards from Cat's win states, until reach the initial state and accept, or exhaust all the reachable edges without hitting the initial state and reject. The runtime is bounded by the number of edges which is $O(n^4)$.

Therefore, HAPPY-CAT is in P. □