U.C. Berkeley — CS172: Automata, Computability and Complexity Solutions to Problem Set 8
Professor Luca Trevisan 4/20/2007

Solutions to Problem Set 8

1. Define $SETCOVER$ to be
   $\{(U, S_1, \ldots, S_m, k) \mid \forall i,\ S_i \subset U$, and there is $I \subseteq \{1, \ldots, m\}$ with $|I| = k$ and $U = \bigcup_{i \in I} \{S_i\}$

   Show that $SETCOVER$ is **NP**-complete.

   [20 points]

   Solution: SET-COVER can be seen as a generalization of VERTEX COVER. For a given graph G, consider each vertex as a set of the edges incident upon it. Formally, we define the sets $S_u = \{(u, v) \in E \mid v \in V\}$ for all $u \in V$. Also, we take $U = E$ and let k be the same as in the vertex cover instance. Then a SET COVER of this family of sets corresponds exactly to picking vertices (sets) such that at least one vertex corresponding to each edge is picked (i.e. at least one set containing every element is picked). Hence, the graph G has a vertex cover of size at most k if and only if the above instance has a set cover of size k. (Note that if there is a cover of size less than k, then there is also one of size exactly k since we can always add a few extra sets.)

   To show that the problem is in **NP**, it suffices to note that given an $I \subseteq \{1, \ldots, m\}$, we can verify in polynomial time that $|I| = k$ and $U = \bigcup_{i \in I} \{S_i\}$.

2. Define the language

   $$\text{ShortestPath} = \{(G, k, s, t) | \text{ the shortest path from } s \text{ to } t \text{ in } G \text{ has length } k\}$$

   (a) Prove that ShortestPath is in **NL**.

   (b) Prove that ShortestPath is in **L** if and only if **L** = **NL**.

   [40 points]

   Solution:

   (a) Solution 1: We construct a **NL**-machine for ShortestPath as follows: on input $\langle G, k, s, t \rangle$, first compute r_{k-1} (the number of vertices reachable from s in at most $k-1$ steps). Then, on input $\langle G = (V, E), k, s, t \rangle$ and r_{k-1} on the work tape,

```
$d \leftarrow 0$
$flag \leftarrow FALSE$
for all $w \in V$ do
   $p \leftarrow s$
   for $i \leftarrow 1$ to $k - 1$ do
      non-deterministically pick a neighbor $q$ of $p$
   if $p = w$ then
      $d \leftarrow d + 1$
      if $w = t$ reject
      if $w$ is a neighbor of $t$ then
         $flag \leftarrow TRUE$
if $d < r_{k-1}$ reject
if flag then accept else reject
```

SOLUTION 2: Observe that **NL**is closed under intersection, and that SHORTESTPATH $= L_1 \cap L_2$ where $L_1 = \{\langle G, k, s, t\rangle \mid$ there is a path from s to t of length at most $k\}$ and $L_1 = \{\langle G, k, s, t\rangle \mid$ there is no path from s to t of length at most $k-1\}$. On the other hand, it is clear that $L_1 \in$ **NL** and that $L_2 \in$ **coNL** $=$ **NL**.

(b) SOLUTION 1: It suffices to prove that $\overline{PATH} \leq_L$ SHORTESTPATH, since **NL** $=$ **coNL** and $\overline{PATH}$ is **coNL**-complete. Given an instance $\langle G, s, t\rangle$ of $PATH$, the log-space transducer for this reduction outputs $\langle G', n+1, s, t\rangle$ where n is the number of vertices in G, and G' is constructed from G by adding n new vertices and a path from s to t of length $n+1$ that goes through these new vertices.

SOLUTION 2: If SHORTESTPATH $\in$ **L**, then we can solve $PATH$ in logarithmic space by invoking the logarithmic space machine for SHORTESTPATH for k from 0 to the number of vertices in the graph.

3. (Sipser 8.9) A *ladder* is a sequence of strings $s_1, s_2, \ldots, s_k$, wherein every string differs from the preceding one in exactly one character. For example the following is a ladder of English words, starting with "head" and ending with "free": head, hear, near, fear, bear, beer, deer, deed, feed, feet, fret, free.

Let $LADDER_{\text{DFA}} = \{\langle M, s, t\rangle \mid M$ is a DFA and $L(M)$ contains a ladder of strings, starting with s and ending with $t\}$. Show that $LADDER_{\text{DFA}}$ is in **PSPACE**.
[30 points]

SOLUTION: It suffices to show that $LADDER \in$ **NPSPACE**. The idea is as follows: given $\langle M, s, t\rangle$, reject if $|s| \neq |t|$. Otherwise, consider a graph G of exponential size whose vertices are indexed by strings in $\Sigma^{|s|}$, and there is a directed edge from w_1 to w_2 iff w_1 and w_2 differ in exactly one character, and $w_1, w_2 \in L(M)$. Then, $\langle M, s, t\rangle \in LADDER$ iff there is a path from s to t in G. This we can check in **NPSPACE** by guessing the path (akin to the **NL** algorithm for $PATH$), and at each step, storing only the name of current vertex (which is a string in $\Sigma^{|s|}$. To guess the path, at vertex w_1, we will nondeterministic select a new vertex w_2 that differs from w_1 in exactly one character, and verify that M accepts w_2. (We can ensure that the machine always halts by keeping a counter and incrementing it with each guess, and rejecting when the counter hits $|\Sigma|^{|s|}$. This is because if there exists a chain from s to t, then there exists one of length at most $|\Sigma|^{|s|}$ by removing loops.)