CS103 Handout 23
Fall 2011 December 5, 2011

Section Handout 8

Problem One: Nondeterministic Algorithms

Below are two problems that are difficult to solve deterministically but are simple to solve nondeterministically. Show how to solve each of these problems using NTMs.

i. In lecture, we saw that the RE languages are closed under union by showing how to build a multitape TM that simulated multiple machines in parallel, accepting if any of them accepted. Devise an alternative construction that uses a single-tape, nondeterministic Turing machine to prove that the RE languages are closed under union. Which proof do you like more?

ii. Let $A_{<4}$ = { ⟨M⟩ | M accepts some string of length less than four }. On Problem Set 8, you proved that this language was recognizable without having nondeterminism at your disposal. Now that you have nondeterminism available, give another proof that $A_{<4}$ is recognizable.

Problem Two: NP

For each of the following languages, show that the language is in NP by designing a polynomial-time verifier.

i. A matching in an undirected graph is a set of edges such that no two edges share an endpoint. Let *MATCHING* = { ⟨G, k⟩ | G is an undirected graph with a matching of size at least k }. Prove that *MATCHING* ∈ NP by designing a polynomial-time verifier for it.

ii. Given a sequence of numbers $x_1, x_2, \ldots, x_n$, an **ascending subsequence** is a subsequence of the original sequence (that is, some elements of the sequence taken in the original order in which they appear) such that each term is larger than the previous term. For example, given the sequence 2, 3, 0, 1, 4, the subsequence **2**, **3**, **4** is an ascending subsequence, as is **0**, **1**, **4**. Let *ASCEND* = { ⟨$x_1, x_2, \ldots, x_n$, k⟩ | There is an ascending subsequence of $x_1 \ldots x_n$ with length at least k. } Prove that *ASCEND* ∈ NP by designing a polynomial-time verifier for it.

Problem Three: NP-Completeness

The independent set problem, as covered in lecture, is specified as follows:

INDSET = { ⟨G, k⟩ | G is an undirected graph that contains an independent set of size k }

As we saw in lecture, INDSET is NP-complete. Using the fact that INDSET is NP-complete, you will prove that the **set packing problem** is NP-complete as well.

In the set packing problem, you are given a list of n sets $S_1, S_2, \ldots, S_n$ along with a number k. The goal is to answer the question

Is there a collection of k sets from the list $S_1, S_2, \ldots, S_n$
such that no element is contained in two of those k sets?

For example, given the sets

{1, 3, 5}, {1, 2, 3}, {2, 4}, {2, 5, 7}, {6}

And the number 3, we would answer "yes" because the collection of sets {1, 3, 5}, {2, 4}, and {6} collectively have no elements in common with one another.

Formally, we define the set covering problem as

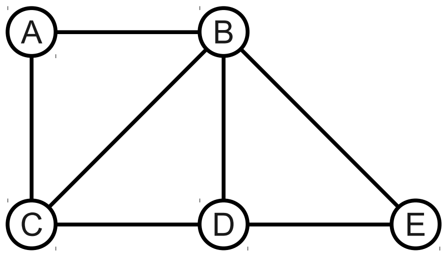
SETPACK = { $\langle S_1, S_2, \ldots, S_n, k\rangle$ | There are k mutually non-overlapping sets in $S_1 \ldots S_n$ }

i. Prove that SETPACK is in NP by designing an NTM that decides it in polynomial time.

To show that SETPACK is NP-complete, we will reduce the IS problem to it. Given a graph G = (V, E), we will construct a family of sets whose elements are the edges in G. There will be one set for each vertex in the graph. Specifically, for each node in $v_i \in V$, we will create a set S_i defined as follows:

$$S_i = \{ \{v_i, v_j\} \mid \{v_i, v_j\} \in E \}$$

That is, the set associated with the vertex v_i is the set of all edges incident to v_i. For example, given this graph:



We would construct the sets

- $S_A = \{ \{A, B\}, \{A, C\} \}$
- $S_B = \{ \{A, B\}, \{B, C\}, \{B, D\}, \{B, E\} \}$
- $S_C = \{ \{A, C\}, \{B, C\}, \{C, D\} \}$
- $S_D = \{ \{B, D\}, \{C, D\}, \{D, E\} \}$
- $S_E = \{ \{B, E\}, \{D, E\} \}$

ii. Using this reduction, prove that SETPACK is NP-complete by reducing INDSET to SETPACK.

Thanks for attending! Good luck on the final exam!