3P17/4P17 COMPUTABILITY 2014-2015 Lecture Handout 14

A note on input form checks

Note that in Handout 13, we did not explicitly state the algorithm do the standard input form checks and reject if the string was not of the form we care about. We shall dispense with it here as well.

The class NP via non-deterministic polynomial time Turing machines

Define $\mathbf{NTIME}(f(n))$ to be the collection of languages decided by non-deterministic Turing machines in time $O(f(n))$.

Define **NP** as follows:

NP is the collection of languages that are decided by non-deterministic polynomial time Turing machines.

In other words,

$$\mathbf{NP} = \bigcup_{k=1}^{\infty} \mathbf{NTIME}(n^k).$$

Example: CLIQUE

A **clique** in an undirected graph G is a subgraph such that every pair of nodes in the subgraph is connected by an edge. A **k-clique** is a clique with k nodes (and therefore $\binom{k}{2}$ edges).

We define the following language:

$$\text{CLIQUE} = \{\langle G, k\rangle : G \text{ is an undirected graph with a } k\text{-clique}\}.$$

Theorem 1. *CLIQUE* $\in$ ***NP***.

Proof. The following is a polynomial time non-deterministic TM for deciding CLIQUE:

N
On input $\langle G, k\rangle$ where G is a graph:

(1) Check G has at least k nodes, reject if it does not.

(2) Non-deterministically select a subset c of k nodes of G.

(3) Test if G contains edges between all nodes in c.

(4) If so, accept, otherwise, reject.

The correctness is clear.

Step (1) is executed once and only involves a single scan through the input and a comparison with k. It can therefore be done in polynomial time.

Selecting a subset of the nodes can be done in polynomial time since it only involves scanning through the input and choosing or rejecting nodes to include. So an execution of line (2) can be done in polynomial time.

Checking if each pair of nodes in c has an edge in G involves $O(n^2)$ number of possible edges to check, where n is the number of nodes in in G. Each check involves scanning through the input, which can be done in polynomial time.

□

Example: HAMPATH

A ***Hamiltonian path*** in a directed graph G is a directed path that visits each node exactly once.

Define the language

$$\text{HAMTPATH} = \{\langle G, s, t\rangle : G \text{ is a directed graph with a Hamiltonian path from } s \text{ to } t\}.$$

Theorem 2. *HAMPATH* $\in$ ***NP***.

Proof. The following is a polynomial time NDTM that decides HAMPATH:

N
On input $\langle G, s, t\rangle$ where G is a directed graph with nodes s and t,

(1) Write a list of n numbers $p_1, p_2, \ldots, p_n$ where n is the number of nodes in G. Each number is non-deterministically selected from to be between 1 and n.

(2) Check for repetitions in the list. If any are found, reject.

(3) Check that $p_1 = s$ and $p_n = t$. If either fail, reject.

(4) For each $i = 1, 2, \ldots, n-1$, check if (p_i, p_{i+1}) is a directed edge in G. If any are not, reject.

(5) Accept.

This algorithm is correct, meaning that it does indeed decide HAMPATH: If there is a directed Hamiltonian path from s to t in G, it will be a sequence of vertices $p_1, p_2, \ldots, p_n$ where $p_1 = s, p_n = t$ and each (p_i, p_{i+1}) is an edge in G. Since this sequence will get generated by branch of the computation tree, then that branch will accept.

If, on the other hand, there is no directed Hamiltonian path from s to t in G, then all sequences will lead to rejects, because they will either not start and end at s and t respectively, or G will not have the required edges.

The time complexity of$N$ is also polynomial in $|\langle G, s, t\rangle|$:

-(1) is executed once and takes polynomial time since it just needs to determine what n is then choose n numbers.
-(2) is executed once and performs $O(n^2)$ comparisons between the numbers. Each comparison involves scanning once through the list of numbers, and so this step contributes polynomial steps overall.
-(3) is executed once and involves a couple of checks, hence contributes polynomial steps overall.
-(4) checks at most n edges and each check involves scanning through the tape. It is executed once and therefore contributes polynomial steps overall. □

Note, "Each number is non-deterministically selected from to be between 1 and n" means that every possible number gets chosen by the TM. N branches into n copies to select p_1, each of those copies branch into n copies to select p_2, and so on. Then some branch of the computation tree will have $p_1 = 6, p_2 = 20, \ldots, p_{100} = 17$, for example.

When we say, for step (4), for example, that it is "executed once" we mean in a particular branch which gets to step (4). Step (4) will, in fact be executed on every branch that reaches that stage. The Turing machines that branch off from each other don't "know" about each other. As far as they are concerned, the other machines don't exist and they are the only ones executing step (4).