Chapter 7, Part 2

Classes P and NP
The Complexity Class P

Juris Hartmanis and Dick Stearns [1965] : proposed computational complexity — measuring complexity of problems by the number of steps (or the number of cells) expended in the worst case under the TM model.

Fundamental results in the Hartmanis-Stearns paper:

1. **Time Hierarchy Theorem** (see Section 9.1) \ldots

   \[
   \text{TIME}(t(n)) \neq \text{TIME}(t(n)^2) \text{ for all reasonable } t(n)
   \]

2. **Linear Speed-up Theorem** \ldots

   \[
   \text{TIME}(t(n)) = \text{TIME}(ct(n)) \text{ for all } c > 0 \text{ and all reasonable } t(n)
   \]

A better hierarchy theorem is proven by Harry Lewis and Stearns.
Alan Cobham [1964], Jack Edmonds [1965], and Michael Rabin [1966] suggested the "polynomial time" as a broad classification of problems that are solvable in a reasonable amount of time

\[ P = \bigcup_{k>0} \text{TIME}(n^k) \]

Why polynomial, why not, say \( n^3 \) ?

Because the "polynomial time" is invariant under the model of computation

**NP** is the **nondeterministic counterpart of P**

\[ NP = \bigcup_{k>0} \text{NTIME}(n^k) \]
The Path Problem

**Input** A directed graph $G = (V, E)$ and $s, t$, $1 \leq s, t \leq |V|$

**Question** Does the graph has a directed path from $s$ to $t$?

We define $PATH$ to be the set of all positive instances $\langle G, s, t \rangle$ to the Path Problem.
The Path Problem

An encoding of a graph can be its adjacency matrix \((a_{ij})\): for every \(i, j, 1 \leq i, j \leq n\), \(a_{ij} = 1\) if \((i, j) \in E\) and 0 otherwise.

The entire encoding can be

\[
0^n \# a_1 a_2 \cdots a_n \# 0^s \# 1^t,
\]

where \(a_1, a_2, \ldots, a_n\) are the rows of the adjacency matrix.
A Polynomial Time Algorithm for \textit{PATH}

Let $G = (V, E)$ be an instance of \textit{PATH}, $n = |V|$, and $A$ the adjacency matrix of $G$.

For each $k \geq 1$, let $A^{(k)}$ be the $k$-th power of the matrix $A$, where $\lor$ and $\land$ replace $+$ and $\times$. 
Let $G = (V, E)$ be an instance of $PATH$, $n = |V|$, and $A$ the adjacency matrix of $G$.

For each $k \geq 1$, let $A^{(k)}$ be the $k$-th power of the matrix $A$, where $\lor$ and $\land$ replace $+$ and $\times$.

Then for every $k \geq 1$ and every $i, j$, $1 \leq i, j \leq n$, the $(i, j)$th entry of $A^{(k)}$ is a $1$ if and only if there is a directed path from $i$ to $j$ of length at most $k$ in $G$. 

**Algorithm for** \( PATH \)

- Compute \( B = A^{(n)} \) by iterative multiplication; that is, compute \( A^{(1)} = A, A^{(2)}, A^{(3)}, A^{(4)}, \ldots \) by multiplying \( A \) by the previous matrix.
- \textbf{if} the \((s, t)\)th entry of \( B = 1 \) \textbf{accept} ; \textbf{else reject}
Running Time on a Multi-tape Turing Machine

- We have only to compute the transpose of $A^{(i)}$.
  - The initial one can be obtained by transposing the input matrix $A$, which requires $O(n^3)$ steps.
  - For the other matrices, that is done by controlling the order in which the entries are computed.
- There are $n^2$ entries per matrix.
- There are $n - 1$ matrix multiplications.
- $2n$ bits are examined to compute an entry of one product.

Thus, the running time is $O(n^4)$ step algorithm
The Relative Primality Problem

**Input**  Integers \( x, y \geq 1 \).

**Question**  Are \( x \) and \( y \) relatively prime to each other, i.e., \( \gcd(x, y) = 1 \)?

Define \( RELPRIME \) to be the set of all positive instances \( \langle x, y \rangle \) of the Relative Primality Problem.

**Note:** \( x \) and \( y \) should not be encoded in unary.
A Polynomial Time Algorithm for RELPRIME

Use the Euclidean Algorithm: On input \( \langle x, y \rangle \):

1. repeat \( x \leftarrow x \mod y; \) swap \( x \) and \( y \); until \( y = 0 \)
2. output \( x \)
How Quickly Does $x$ Decrease?

Suppose $x$ has value $u$, $y$ has value $v$, $v \leq u$, after one iteration of the above algorithm, the value of $x$ becomes $u'$ and the value of $y$ becomes $v'$. and after another iteration of the above algorithm, the value of $x$ becomes $u''$ and the value of $y$ becomes $v''$.

We have:

- $u' = v$,
- if $v > u/2$, then $v' = u \mod v = u - v < u/2$;
- if $v \leq u/2$, then $v' \leq v - 1 < u/2$.

So, we have

- $u'' = v' < u/2$,
- $v'' < u'/2 = v/2$.

This implies that in two iterations, both $x$ and $y$ will be less than half of what they are now.
Running Time Analysis

If $\max\{|x|, |y|\} = n$, then the running time is $O(n^3)$.

(*) if the Euclid algorithm on $\langle x, y \rangle$ outputs 1 then accept; else reject

The Running Time Analysis: $O(n^3)$. 
Theorem. *Every context-free language is in \( \mathcal{P} \).*
Theorem. Every context-free language is in \( P \).

Proof Let \( L \) be context-free. Let \( G \) be a CNF grammar for \( L \). Suppose \( w = w_1 \cdots w_n \) be a string whose membership in \( L \) we are testing.

The case when \( w = \epsilon \) is easy: we accept if and only if \( S \rightarrow \epsilon \) is a rule in \( G \).
The Nonempty Case

So, assume \( w \neq \epsilon \) and let \( w_1, \ldots , w_n \) be the symbols of \( w \).

For each \( i, j, 1 \leq i \leq j \leq n \), let \( t(i, j) \) be the set of all variables from which \( w_i \cdots w_j \) can be produced.
The Nonempty Case

So, assume \( w \neq \epsilon \).

For each \( i, j, 1 \leq i \leq j \leq n \), let \( t(i, j) \) be the set of all variables from which \( w_i \cdots w_j \) can be produced.

We can compute \( t(i, j) \) for all \( i, j, 1 \leq i \leq j \leq n \), using dynamic programming.

Then test the membership by examining whether \( S \in t(1, n) \).
Dynamic Programming for Computing the Table

Set $t(i, i) \leftarrow$ the set of all $A$ such that $A \rightarrow w_i$ is in $G$. Then execute the following:

```
for $\ell = 2$ to $n$
    for $i = 1$ to $n - \ell + 1$
        $j = i + \ell - 1$; $t(i, j) = \emptyset$;
        for $k = i$ to $j - 1$
            if $\exists A, B \in t(i, k), C \in t(k + 1, j)$
                such that $A \rightarrow BC$ is in $G$
                then add $A$ to $t(i, j)$
```

The running time is $O(n^3)$ since $\ell, i, \text{ and } k$ have at most $n$ possible values.

The size of $t(i, j)$ is at most the number of variables of $G$, but that is a constant since $G$ is fixed.
Examples of NP Languages

The Hamilton Path Problem

**Input**  A directed graph \( G = (V, E) \) and \( s, t \in V, s \neq t \)

**Question**  Is there a Hamilton Path from \( s \) to \( t \) in \( G \), i.e., a directed path from \( s \) to \( t \) that visits all the nodes exactly once?

Define \( HAMPATH \) to be the set of all positive instances \( \langle G, s, t \rangle \) to the Hamilton Path Problem.
The Compositeness Problem

**Input**  Integer $x \geq 1$

**Question**  Does $x$ a composite number, i.e., have an integer divisor other than 1 and $x$?

Define $COMPOSITES$ to be the set of all composite numbers $x$. 

A verifier of a language $A$ is an algorithm $V$ such that

$$A = \{w \mid V \text{ accepts } \langle w, c \rangle \text{ for some } c \}.$$  

That is, a verifier is an algorithm that takes two inputs $w$ and $c$ and decides whether to accept or reject in such a way that:

- If $w \in A$, there is an auxiliary input $c$ that makes the verifier accept and
- If $w \notin A$, there is no auxiliary input $c$ that makes the verifier accept.

For a fixed $V$, the string $c$ witnessing to $w \in A$ (that is, one such that $V$ accepts $\langle w, c \rangle$) is called a certificate or a proof.
An Alternative Definition of NP

We will measure the time of $V$ in terms of the length of $w$.

**Definition.** (alternate) **NP** is the class of languages that have polynomial time verifiers.

This means that polynomial time verifies reject (input, proof-candidate) pairs in which the proof candidate is exceedingly long.
Theorem. The alternative definition is equivalent to the first definition of \textbf{NP}.

\textbf{Proof} (Sketch) If \(L\) has a polynomial time verifier, then we can construct a nondeterministic Turing machine that nondeterministic guesses a proof of length bounded by some fixed polynomial and then verifies the proof.

If \(L\) is accepted by a polynomial time nondeterministic Turing machine, we can use the accepting computation paths of the machine as the proofs of membership.
Membership of *HAMPATH in NP*

Define a certificate for each $\langle G, s, t \rangle \in HAMPATH$ to be any sequence $\langle v_1, ..., v_n \rangle$ of nodes such that

(i) for every $i$, $1 \leq i \leq n$, $i = v_j$ for some $j$,
(ii) $s = v_1$,
(iii) $t = v_n$, and
(iv) for every $i$, $1 \leq i \leq n - 1$, $(v_i, v_{i+1}) \in E$.

A correct certificate can be of length $O(n \log n)$ and verification can be done in $O(n^3)$ steps.
Membership in NP

Define a certificate for each $x \in COMPOSITES$ to be any number $y$ such that $y$ divides $x$ and $1 < y < x$. Then a correct certificate can be of length $O(n)$.
The Clique Problem

**Input** A graph $G = (V, E)$ and $k \geq 1$.

**Question** Does $G$ contain a complete graph of size $\geq k$?

Define $CLIQUE$ to be the set of all positive instances $\langle G, k \rangle$ to the Clique Problem.
Membership in \textbf{NP}

\textbf{Theorem.} \textit{CLIQUE} \textbf{is in NP.}

\textbf{Proof} \ (Sketch) Define a certificate for an instance \(\langle G, k \rangle\), where \(G\) is an \(n\) node graph, to be an \(n\) bit sequence \(c = c_1 \cdots c_n\) such that:

\textbf{Exactly} \(k\) \textbf{of} \(c_1, \ldots, c_n\) \textbf{are 1s} \textbf{and} \textbf{for every} \(i, j, 1 \leq i < j \leq n\), \(\textbf{if} \ c_i = c_j = 1\), \textbf{then} \((i, j) \in E\)

Then verification can be done in \(O(n^3)\) steps.
More Problems in NP: Subset Sum

The Subset Sum Problem

**Input** integers $x_1, \ldots, x_k$ and $t$

**Question** Is there a subset of $\{x_1, \ldots, x_k\}$ that adds up to $t$?

Define $\textit{SUBSET-SUM}$ to be the set of all positive instances $\langle S, t \rangle$ to the Subset Sum Problem.
Theorem. \textit{SUBSET-SUM is in NP.}

Proof (Sketch) Define a certificate for an instance $\langle S, t \rangle$ with $|S| = n$ in \textit{SUBSET-SUM} to be an $n$ bit sequence such that

$$\sum_{i=1}^{n} c_{i}x_{i} = t$$

Then verification can be done in $O(n^2)$ steps.