Classes P and NP

The Complexity Class P

Juris Hartmanis and Richard Stearns [1965] : defined *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 Sections 9.1 – 9.10) $\cdots$

   $\mathrm{TIME}(t(n)) \neq \mathrm{TIME}(t(n)^2)$ **for all reasonable** $t(n)$

2. **Linear Speed-up Theorem** $\cdots$

   $\mathrm{TIME}(t(n)) = \mathrm{TIME}(ct(n))$ **for all** $c > 0$ **and all reasonable** $t(n)$

A better hierarchy theorem is proven by Harry Lewis and Stearns

The Complexity Class P (continued)

Alan Cobham [1964], Jack Edmonds [1965], and Michael Rabin [1966] suggested the **"polynomial time"** class as a broad class of problems that are solvable in a *reasonable amount of time*

$$\mathrm{P} = \bigcup_{k>0} \mathrm{TIME}(n^k)$$

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**

$$\mathrm{NP} = \bigcup_{k>0} \mathrm{NTIME}(n^k)$$

Problems in P

The Path Problem

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

Question: Does the graph have a directed path from s to t?

$PATH$: the set of all positive instances $\langle G, s, t\rangle$ of 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 $PATH$

Let $G = (V, E)$ be a DG, $n = |V|$, and A the adjacency matrix of G.

For each $k \geq 1$, let $A^{(k)}$ be the kth power of A, where $\vee$ **and** $\wedge$ **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.**

Thus the following will do for deciding whether $\langle G, s, t\rangle \in PATH$:

(*) Compute $B = A^{(n)}$; **if** the (s, t)th entry of $B = 1$ **accept** ; **else reject**

Running Time Analysis: $2n$ bits examined per entry, n^2 entries, $n - 1$ sequential multiplications yields $A^{(n)}$ $\cdots$ an $O(n^4)$ step algorithm

Testing Relative Primality of Two Numbers

The Relative Primality Problem
Input: Integers $x, y \geq 1$
Question: Are x and y, relatively prime, i.e., $\gcd(x, y) = 1$?

$RELPRIME$: 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

We will use *Euclid's algorithm for GCD*:

(*) if Euclid's Alg. on $\langle x, y \rangle$ outputs 1 **then accept** ; **else reject**

Running Time Analysis: $O(n^3)$, based on running time of Euclid's algorithm. Thus Relative Primality is in P.

A Cubic Time Algorithm for GCD

Euclid's Algorithm for GCD: On input $\langle x, y \rangle$:

1. **repeat** $x \leftarrow x \bmod y$; swap x and y; **until** $y = 0$
2. **output** x

How quickly does x decrease? (After 1 step, x remains $> y$)
If $y > x/2$, then $x \bmod y \le x - y < x/2$;
If $y \le x/2$, then $x \bmod y \le y - 1 < x/2$.
So, each iteration reduces x by at least half. (1 bit less per iteration)

If $\max\{|x|, |y|\} = n$, then the running time is $O(n^3)$. (Finding remainder is $O(n^2)$, e.g., using binary division.)

Invariant in Euclid's algorithm: x, y are relatively prime throughout iff they are relatively prime initially, since relative primality is preserved by $x \leftarrow x \bmod y$. (Consider 2 cases: $x \le y$, $x > y$.)

Polynomial Time Decidability of Context-Free Languages

Theorem. *Every context-free language is in* P.

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

if $n = 0$ **then accept** if and only if $S \to \epsilon$ is a rule.

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

Idea: 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

Initial: $t(i,i) \leftarrow$ the set of all A such that $A \rightarrow w_i$ is a rule

Loop:

for $\ell = 2$ **to** n {length of substring}
  for $i = 1$ **to** $n - \ell + 1$ {pos$^{\underline{\mathrm{n}}}$ (left index) of substring}
    $j = i + \ell - 1; t(i,j) = \emptyset;$
    for $k = i$ **to** $j - 1$ {dividing point in substring}
      if $\exists A, B \in t(i,k), C \in t(k+1,j)$ such that $A \rightarrow BC$ is a rule
        then add A to $t(i,j)$

Final Test: **accept** if and only if $S \in t(1,n)$ ∎

The Class NP

The Hamiltonian Path Problem
Input: A directed graph $G = (V, E)$ and $s, t \in V$, $s \neq t$
Question: Is there a Hamiltonian Path from s to t in G, i.e., a directed path from s to t that visits all the nodes exactly once?

$HAMPATH$: the set of all positive instances $\langle G, s, t \rangle$ of the Hamiltonian Path Problem

The Compositeness Problem
Input: Integer $x \geq 1$
Question: Is x a composite number, i.e., does it have an integer divisor other than 1 and x?

$COMPOSITES$: the set of all composite numbers x

A Characterization of NP by Verifiers

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

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

Measure the time of V in terms of the length of w. For a fixed V, the string c serving as witness to $w \in A$ is called a **certificate** or a **proof**

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

Key observation: if V accepts $\langle w, c\rangle$ in time $p(|w|)$ for some polynomial p, then the portion of c examined in the computation is at most of length $p(|w|)$.

Equivalence of the Two Definitions of NP

Theorem. *The alternative definition of NP is equivalent to the first.*

Proof (Sketch) Let p be any polynomial.

Suppose L has a $p(n)$ time verifier V. Then for every x, we can consider all certificates of length at most $p(|x|)$. Let N be an NTM that, on input x of length n, (i) nondeterministically guesses a string c, $|c| \leq p(n)$, where $n = |x|$, (ii) simulates V on $\langle x, c\rangle$, and (iii) accepts if and only if V has accepted. Then N decides L and is $O(p(n))$ time.

Suppose L is decided by a $p(n)$ time NTM N. Consider a verifier V that, on input $\langle x, c\rangle$, treats c as a computation path (choice sequence) for N and simulates N on x along c for at most $p(|x|)$ steps, and accepts if and only if N has accepted. Then for every x, V accepts x for some c if and only if N on x accepts for some computation path. Also it's clear that V operates in polynomial time. ∎

Membership of $HAMPATH$ and $COMPOSITES$ in NP

$HAMPATH$: 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, $1 \le i \le n$, $i = v_j$ for some j,

(ii) $s = v_1$,

(iii) $t = v_n$, and

(iv) for every i, $1 \le i \le n-1$, $(v_i, v_{i+1}) \in E$.

A correct certificate can be of **length** $O(n \log n)$ and verification can be done in $O(n^3)$ **steps** using an adjacency matrix.

$COMPOSITES$: 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** $O(n)$

More Problems in NP

The Clique Problem
Input: **A graph** $G = (V, E)$ **and** $k \geq 1$
Question: **Does** G **contain a complete graph of size** $\geq k$**?**

$CLIQUE$: the set of all positive instances $\langle G, k \rangle$ of the Clique Problem

Theorem. *$CLIQUE$ is in NP.*

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:

for every i, j, $1 \leq i < j \leq n$, **if** $c_i = c_j = 1$, **then** $(i, j) \in E$

Then verification can be done in $O(n^3)$ steps. ∎

More Problems in NP (cont'd)

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?

$SUBSET\text{-}SUM$: the set of all positive instances $\langle S, t\rangle$ of the Subset Sum Problem

Theorem. *$SUBSET\text{-}SUM$ is in NP.*

Proof (Sketch) Define a certificate for an instance $\langle S, t\rangle$ with $|S| = n$ in $SUBSET\text{-}SUM$ to be a k-bit sequence $c_1 \cdots c_k$ such that $\sum_{i=1}^{k} c_i x_i = t$ $(k \leq n)$.

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