Complexity Class NP

Poly-time verifiability

Π is a decision problem

Accept(Π) = the set of all instances for Π for which the decision is Yes

Reject(Π) = the set of all instances for Π for which the decision is No

Every $I \in$ Accept(Π) has a certificate C.

No $I \in$ Reject(Π) can have a certificate.

A certificate is a string used to confirm the membership of I in Accept(Π)

Π — To decide whether a person is a swimmer

All i/p instances : all humans in the world

Accept (Π) — the swimmers

Reject (Π) — the non-swimmers

$h \in$ Accept (Π)

Set up a swimming test for h.

A swimmer "certificate" issued by a competent authority compositeness certificate

- a non-trivial divisor d of n

Primality certificates are different

- Pratt certificates (much involved)

HAM-CYCLE / HAM-PATH

- Provide the cycle / path

Eulerian - tour

- supply a tour

Replacing guesses by a certificate.

A certificate must be efficiently verifiable.

$|I| = n \quad (I \in \mathrm{Accept}(\Pi))$

C – a certificate for I.

Any verification algo must read C.

$\downarrow$

efficient (poly-time)

$|C|$ must be bounded by a polynomial in n.

Such certificates are called short/succinct.

Theorem: Π has a non-deterministic poly-time algorithm if and only if each $\underline{I} \in \text{Accept}(\Pi)$ has a succinct certificate.

Proof [if] Each $\underline{I} \in \text{Accept}(\Pi)$ has a succinct certificate C.

$|\underline{I}| = n$

$|C| \leq n^k$

```
Non-deterministically generate a candidate C (a string of length ≤ n^k);
Verify whether C is a certificate for I;
if so, output "Yes"; else output "No";
```

If $I \in Accept(\Pi)$, then I has a certificate.

If $I \in Reject(\Pi)$, then I has no certificates.

[only if] Π has a non-deterministic poly-time algo A.

Upon input I, A makes some guesses $g_1, g_2, g_3, \ldots, g_m$.

If $I \in Accept(\Pi)$, at least one set of guesses succeeds in outputting YES

$\langle g_1, g_2, g_3, \ldots, g_m \rangle$ — a certificate

$m \leq runtime(A) \leq n^k$ ← succinct

NP = the class of problems that are verifiable in polynomial time

P = the class of problems that can be solved in polynomial time

P – easy solvability

NP – easy verifiability

$$P \subseteq NP$$

$\Pi \in P \Rightarrow \Pi$ has a poly-time algorithm

$|C| = 0$ (empty certificate)

1971 Stephen Cook
Leonid Levin

Cook-Levin theorem:

Identifies a subclass of NP consisting of the most difficult problems of NP.

Reduction of one problem to another

Π, Π' two problems

A reduction from Π to Π' is an algorithm that converts an instance $\underline{I}$ of Π to an instance $\underline{I}'$ of Π' such that $\underline{I} \in Accept(\Pi)$ if and only if $\underline{I}' \in Accept(\Pi')$.

$\Pi \leq \Pi'$

The reduction algorithm must run in time polynomially bounded by the input size $|I|$.

If Π' has a poly-time deterministic algo, then Π also has a poly-time deterministic algo.

$$\Pi' \longrightarrow A' \quad — \quad O(n^k)$$

I for Π $\quad |I| = n \quad I \mapsto I' \quad |I'| \leq n^d$

$O(n^{dk})$ time algo for Π.

Examples of reduction

(1) HAM-PATH $\leq$ HAM-CYCLE

$$(G, s, t) \mapsto G'$$

G' has a Hamiltonian cycle

$\Leftrightarrow$ G has an s,t Hamiltonian path

$$G = (V, E) \qquad G' = (V', E')$$

$$V' = V \cup \{z\} \quad \text{(a new vertex)}$$

$$E' = E \cup \{(z,s), (t,z)\}$$

(2) HAM-CYCLE $\leq$ HAM-PATH

$$G \longmapsto (G', s', t')$$

G' has an s, t Hamiltonian path

$\Leftrightarrow$ G contains a Hamiltonian cycle

Pick an arbitrary $u \in V(G)$

Break u into two vertices u_1 and u_2.

$$V(G') = \left(V(G) \setminus \{u\}\right) \cup \{u_1, u_2\}$$

(u, v) Add (u_1, v) and (u_2, v) to $E(G')$

(v, w), $v \neq u$, $w \neq u$ Add (v, w) to $E(G')$

$s' = u_1$

$t' = u_2$