NP-completeness

P – polynomial-time

NP – nondeterministic polynomial time

million-dollar question

$$P \stackrel{?}{=} NP$$

P = the class of all computational problems that can be solved in polynomial time $\rightarrow$ polynomial in the input size

Decision problems

Functional / optimization problems

Π computational problem

I an instance for Π (input)

$|I|$ = the no. of bits needed to represent I

O/P — Yes / No, Accept / Reject

P consists of problems that are efficiently soluable.

n – i/p size

$O(n^{100})$ time

$O(n \log n)$, $O(n^3)$, ...

August 2002
AKS

P is infinite

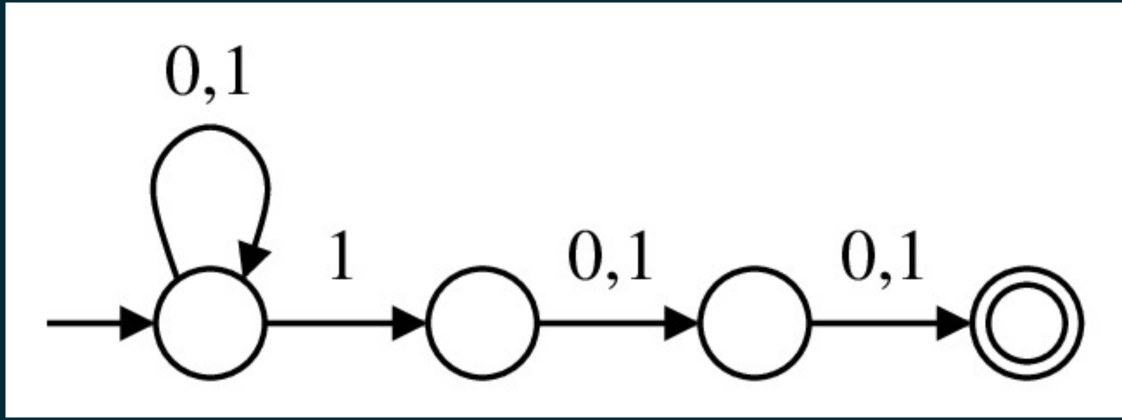
$k \in \mathbb{N}$ k–MUL

I/p: (a, b) $O/p = \begin{cases} 1 & \text{if } b = ak \\ 0 & \text{if not} \end{cases}$

The boundary of P is hazy.

NP – nondeterministic algo
– certificates

FA, PDA



COMPOSITE

Input : A positive integer n

Output : Yes if n is composite

No if n is not composite

A nondeterministic algorithm can guess

Any guess can be modeled by a sequence of bit guesses

Guessing each bit takes constant time.

```
Let l be the bit-length of n;
for (i=0; i<l; ++i) guess a bit d_i from the set {0,1};
Let d = (d_{l-1} d_{l-2} ... d_1 d_0)_2;
if ((2 <= d <= n-1) && (d divides n)) output "Yes"; else output "No";
```

If n is composite, some sequence of guesses succeeds.

If n is not composite, no sequence of guesses succeed.

Step 1: Compute l $O(l)$ time

Step 2: $O(l)$ time

Step 3: $O(l)$ time

Step 4: $O(l^2)$ time (division)

Total $O(l^2)$

- Hints
- Infinitely many processors (each processor handles one d)

$G = (V, E)$

$|V| = n$

A Hamiltonian cycle in G is a permutation $v_1, v_2, \ldots, v_n$ of the vertices in G such that

$$(v_1, v_2), (v_2, v_3), \ldots, (v_{n-1}, v_n), (v_n, v_1) \in E$$

HAM-CYCLE

DHAM-CYCLE

Given G, determine whether G contains a Hamiltonian cycle

```
for (i=0; i<n; ++i) guess the vertex v_i;
Check whether v_0, v_1, ..., v_{n-1}, v_0 is a Hamiltonian cycle in G;
if so, output "Yes"; else output "No";
```

$|G| = |V| + |E|$ i/p size

$|V| = n$ $V = \{0, 1, 2, \ldots, n-1\}$

Step 1 : $O(n \log n)$ time

Step 2 : visited$[v_0] = 1$

$v_0, v_1, v_2, v_3, \ldots, v_{i-1}$

poly$(|G|)$ time

HAM-PATH

Input : (G, s, t)

$s = v_0, v_1, v_2, \dots, v_{n-1} = t$

permutation of the vertices

$(v_i, v_{i+1}) \in E \quad \forall\, i = 0, \_, n-2.$

DHAM-PATH

EULERIAN-TOUR

$G = (V, E)$ undirected graphs

$|E| = m$

$e_0, e_1, e_2, \ldots, e_{m-1}$

a permutation of the edges

$(u_0, v_0), (u_1, v_1), (u_2, v_2), \ldots, (u_{m-1}, v_{m-1})$

$v_0 = u_1 \quad v_1 = u_2 \quad v_{m-1} = u_0$

```
for (i=0; i<m; ++i) guess the edge e_i;
Check whether e_0,e_1,...,e_{m-1} is an Eulerian tour in G;
if so, output "Yes"; else output "No";
```

step 1 : $e_i = (u_i, v_i)$ $2\log n$ bits

$O(m \log n)$

step 2 : poly$(|G|)$ time

NP = the class of problems that have nondeterministic polynomial-time algorithms

A DFA is also an NFA.

A deterministic algorithm is also non-deterministic

$P \subseteq NP$ ✓

$NP \overset{?}{\subseteq} P$ (Not known)

Let $\Pi \in NP$.

Π have a non-deterministic poly-time algo A.

The running time is $f(n)$ where n is the i/p size.

A makes g bit guesses. $f(n) = n^d$

$g \leq f(n)$ $O\left(2^{n^d + d \log_2 n}\right)$

2^g possibilities

simulate A by a deterministic algo,

running time $O\left(2^g f(n)\right) = O\left(2^{f(n)} f(n)\right)$