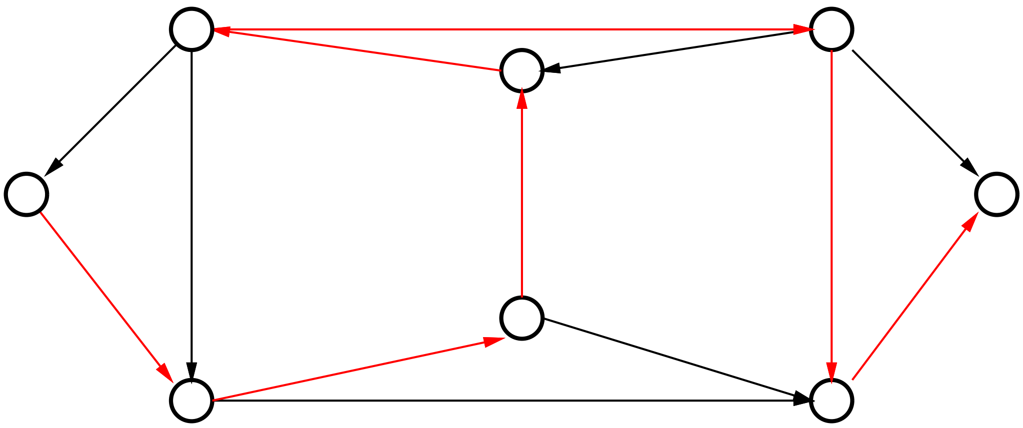
Hamiltonian Path



A Hamiltonian path in a directed G visits each note once.

Hamiltonian Path

$$\text{HAMPATH} = \{\langle G, s, t\rangle | G \text{ has Hamiltonian path from } s \text{ to } t\}$$

Question: How hard is it to decide this language?

Hamiltonian Path

$$\text{HAMPATH} = \{\langle G, s, t\rangle | G \text{ has Hamiltonian path from } s \text{ to } t\}$$

Easy to obtain exponential algorithm

- generate each potential path
- check whether it is Hamiltonian

Hamiltonian Path

This problem has one very interesting feature:
polynomial verifiability.

- we don't know a fast way to find a Hamiltonian path
- but we can check whether a path is Hamiltonian in polynomial time

In other words,

- verifying a path is much easier than
- determining whether it exists

Composite Numbers

A natural number is *composite* if it is the product of two integers greater than one.

$$\text{COMPOSITES} = \{x | x = pq \text{ for integers } p, q > 1\}$$

- we don't know a polynomial-time algorithm for deciding this problem
- we can easily verify that a nubmer is composite
  How?

Verifiability

Some problems may not be polynomially verifiable.

There is no known way to verify $\overline{\text{HAMPATH}}$ in polynomial time.

Verifiability

A **verifier** for a language A is an algorithm V where

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

- measure verifier time only by length of w
- a *polynomial verifier* runs in polynomial time in $|w|$.
- a language A is *polynomially verifiable* if it has a polynomial verifier.

Verifiability

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

The verifier uses the additional information c to verify $w \in A$.

String c is called

- a certificate, or
- a proof

For polynomial verifiers, c must have length polynomial in $|w|$. Why?

Examples

For HAMPATH, a certificate for

$$\langle G, s, t \rangle \in \text{HAMPATH}$$

is simplly the Hamiltonian path from s to t.

Can verify in time polynomial in $|\langle G \rangle|$ whether path is Hamiltonian.

Examples

For COMPOSITES, a certificate for

$$x \in \text{COMPOSITES}$$

is simply one of its divisors.

Can verify in time polynomial that divisor divides x.

Definition: NP is the class of languages that have polynomial time verifiers.

NP is important because it includes may problems of practical interest.

- hamiltonian path
- travelling salesman
- composites (cryptography)

Name comes from *nondeterministic polynomial time*.

NP

Here is a NTM that decides HAMPATH in poly time.

On input $\langle G, s, t \rangle$,

1. Write a list of numbers $p_1, \ldots, p_m$, where m is number of nodes in G, and $1 \leq p_i \leq m$.

2. Check for repetitions. If any found, *reject*.

3. Check whether $p_1 = s$ and $p_m = t$. If either fails, *reject*.

4. for $i, 1 \leq i \leq m-1$, check whether (p_i, p_{i+1}) is an iedge in G. If any is not, *reject*. Otherwise *accept*.

NP

On input $\langle G, s, t \rangle$,

1. Write a list of numbers $p_1, \ldots, p_m$ ...

2. Check for repetitions ...

3. Check whether $p_1 = s$ and $p_m = t$ ...

4. check whether (p_i, p_{i+1}) is an iedge in G ...

- Stage 1 polynomial time
- Stages 2 and 3 simple polynomial-time check.
- state 4 insignificant.

NP

Question: Coincidence? I think not!

Theorem: A language is in NP if and only if it is decided by some nondeterministic polynomial-time TM.

NP

Basic Idea:

- NTM simulates verifier by guessing the certificate.
- Verifier simulates NTM by using accepting branch as certificate.

NP

Claim: If $A \in NP$, then is decided by some polynomial-time NTM.

Let V be poly-time verifier for A.

- single-tape TM
- runs in time n^k

N: on input w of length n

- Nondeterministically select *string* c of length n^k.
- Run V on $\langle w, c \rangle$
- If V accepts, *accept*; otherwise *reject*.

NP

Claim: If A is decided by a polynomial-time NTM N, then $A \in NP$.

Construct polynomial-time verifier V as follows.

V: on input w of length n

- Simulate N on input w, treating each symbol of c as a description of each step's non-deterministic choice.
- If this branch accepts, *accept*, otherwise *reject*.

NTIME$(t(n))$

Definition:

$$\text{NTIME}(t(n)) = \{L \mid L \text{ is a language decided by an } O(t(n))\text{-time NTM}\}$$

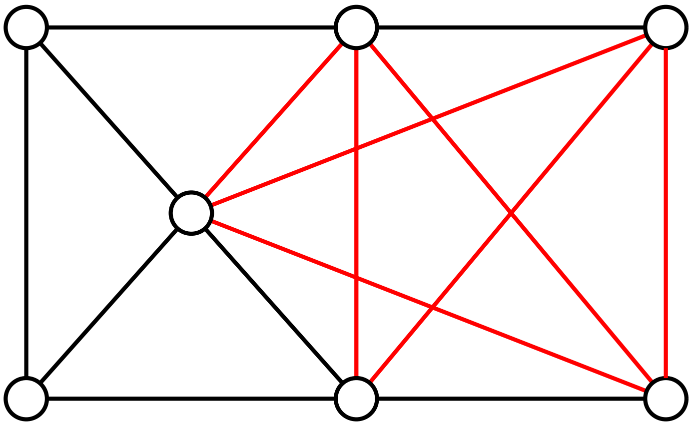
NP

Corollary:

$$NP = \bigcup_k \mathrm{NTIME}(n^k)$$

NP is insensitive to choice of reasonable non-deterministic computational model.

Examples: Clique



A *clique* in a graph is a subgraph where every two nodes are connected by an edge.

A k*-clique* is a clique of size k.

Examples: Clique

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

Examples: Clique

Theorem:

$$\text{CLIQUE} \in NP$$

The clique is the certificate.

Here is a verifier:

V: on input $(\langle G, k\rangle, c)$

- if c is not a k-clique, *reject*
- if G does not contain all vertexes of c, *reject*
- *accept*

Examples: SUBSET-SUM

We have

- a collection of numbers $x_1, \ldots, x_k$
- target number t
- does some subcollection add up to t?

$$\text{SUBSET-SUM} = \{\langle S, t\rangle | S = \{x_1, \ldots, x_k\} \text{ and for some } \{y_1, \ldots, y_\ell\}\, subseteq\, \{x_1, \ldots, x_k\} \sum_{y_j} = t\}$$

Examples: SUBSET-SUM

We have

$$(\{4, 11, 16, 21, 27\}, 24) \in \text{SUBSET-SUM}$$

because $4 + 21 = 25$.

Collections are multisets: repetitions allowed.

Examples: SUBSET-SUM

Theorem:

$$\text{SUBSET-SUM} \in NP$$

The subset is the certificate.

Here is a verifier:

V: on input $(\langle S, t\rangle, c)$

- test whether c is a collection of numbers summing to t.
- test whether S contains c
- if either fail, *reject*, otherwise *accept*.

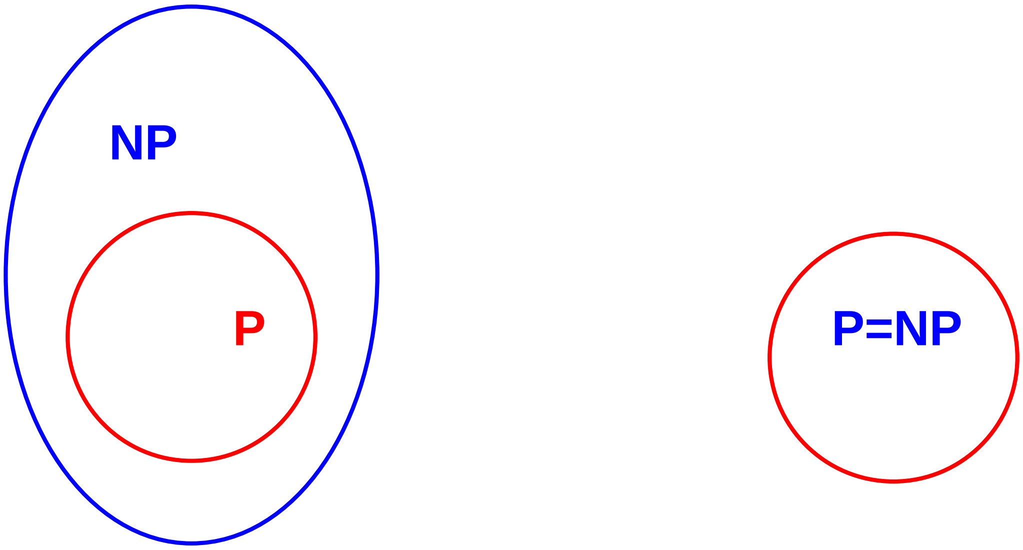
Complementary Problems

$\overline{\text{CLIQUE}}$ and $\overline{\text{SUBSET-SUM}}$ are not obviously members of NP.

It seems harder to verify that something does *not* exist.

Definition: The class coNP is $\overline{NP}$.

No one knows if coNP is distinct from NP.



Whether $P = NP$ is one of the great unsolved mysteries in contemporary mathematics.

- most computer scientists believe they are not equal
- most bogus proofs show them equal

Remarks

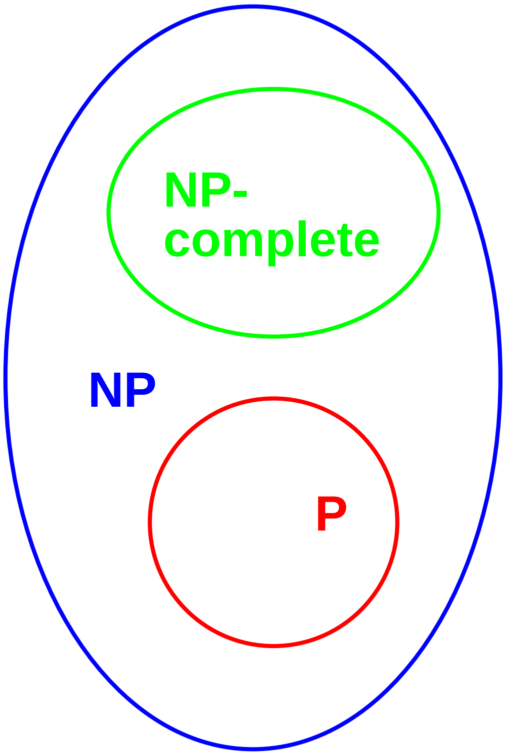
If P differs from NP, then the distinction between P and $NP - P$ meaningful and important.

- languages in P tractable
- languages in $NP - P$ intractable

Until we can prove that $P \neq NP$, there is no hope of proving that a language lies in $NP - P$.

Nevertheless, we *can* prove statements of the form "If $P \neq NP$ then $A \in NP - P$."

Remarks



The class of *NP-complete* languages are

- hardest" languages in NP
- "least likely" to be in P
- If any NP-complete $A \in P$, then $NP = P$.