NP-completeness

What to do when you can't find an efficient algorithm for a given problem.

efficient means $O(n^c)$ time where n = input size, c = constant

Shortest path — $O(nm)$ time algorithm

vs.

longest simple path (no cycles) — no efficient alg. known [Is there one?]

Even the decision problem: (solution is either Yes or No.)

Does the longest simple path from vertex s to vertex t in G have length $\geq k$?

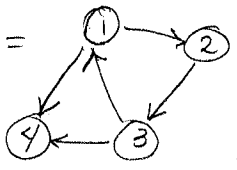
has no known efficient algorithm. (even if edge weights are 1)

Input: $\langle G, s, t, k \rangle$

Output: Yes if there is a simple path from s to t in G that has $\geq k$ edges

No otherwise.

The angle brackets indicate a binary encoding of G, s, t, and k. For example, if

$G =$ 

$s = 1$ $t = 4$ $k = 3$

$\langle G, s, t, k \rangle$ might be

G	s	t	k
0101 0010 1001 0000	001	100	011

0101001010010000 001 100 011

which has size 25

We will restrict our attention to decision problems.

A Decision Problem defines a language ← a set of strings of 0's and 1's.

For example:

$$\text{LONGEST\_PATH} = \{\langle G, s, t, k\rangle \mid \text{there is a simple path in } G \text{ from } s \text{ to } t \text{ that has} \geq k \text{ edges}\}$$

$$= \{\ \ldots,\ 01010010100100000011100011,\ \ldots\ \}$$

A language L is decided (in polynomial time) by an algorithm A if on input x, A outputs "Yes" if $x \in L$ and "No" if $x \notin L$ (and runs in time $O(|x|^c)$ for some constant c).

Complexity Class

$$P = \{ L \mid \text{there is an algorithm that decides } L \text{ in polynomial time} \}$$

For example:

$$\text{PATH} = \{\langle G, s, t, k\rangle \mid \text{there is a path in } G \text{ from } s \text{ to } t \text{ that has} \leq k \text{ edges}\}$$

PATH $\in$ P because for $x = \langle G, s, t, k\rangle$, we could run Breadth First Search (or Dijkstra's or Bellman Ford) to find a shortest path from s to t and output "Yes" if it has $\leq k$ edges. and "No" otherwise
This takes $O(m)$ (or $O(m \lg n)$ or $O(mn)$)
which is $O(|x|^c)$ for $c = 2$

For some languages L, it seems difficult to determine if $x \in L$ quickly (in polynomial time) but with a little more information an alg. can verify that $x \in L$ quickly.

(called a "witness" or "certificate")

For example if $x = \langle G, s, t, k \rangle$ and we want to know "Is $x \in$ LONGEST_PATH?" then if there is a simple path w from s to t in G with $\geq k$ edges (i.e. if $x \in$ LONGEST_PATH) an alg. can verify that $x \in$ LONGEST_PATH given w quickly. How?

Longest Path Verifier $V(x, w)$

Suppose $w = w_1, w_2, w_3 \ldots w_r$ and $x = \langle G, s, t, k \rangle$

① if $w_1 \neq s$ or $w_r \neq t$ or $r < k$ then return No

② for $i = 1$ to $r-1$
  if $(w_i\ w_{i+1})$ is not an edge of G then return No

③ If $w_i = w_j$ for any $i \neq j$ then return No

④ return Yes

A language L is verified in polynomial time if there exists a verifier V and constant c such that

$$L = \{ x \mid \exists\, w \text{ such that } V(x, w) = \text{Yes} \text{ and } V \text{ runs in } O(|x|^c) \text{ time} \}$$

$$NP = \{ L \mid L \text{ is verified in polynomial time} \}$$

Why? Many problems are in NP but not all problems in NP are known to be in P.

Is $NP = P$?

Thm $P \subseteq NP$

Proof If $L \in P$ then there is an algorithm A that decides L in poly. time. From A we can build a verifier V that on input (x, w) runs A on input x and ignores w. V runs in poly. time since A does. Thus $L \in NP$.

Examples of languages in NP

$SAT = \{ \phi \mid \phi$ is a boolean formula in conjunctive normal form that has a satisfying truth assignment $\}$

$\phi = (x \vee y \vee z) \wedge (x \vee \bar{y}) \wedge (y \vee \bar{z}) \wedge (z \vee \bar{x})$

Is $\phi \in SAT$? Yes

AND together a bunch or clauses
a clause is OR of a bunch of literals
a literal is a variable or its negation

Is $SAT \in NP$?

Yes If $\phi \in SAT$ then a string w that specifies a satisfying truth assignment is a good witness for ϕ. A verifier need only check that w actually satisfies ϕ (i.e., for each clause at least one literal in the clause is TRUE)

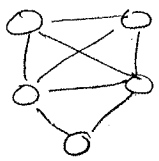
Is $\overline{SAT} \in NP$?

$\overline{SAT}$ ← all formulas that have no satisfying truth assignment

No one knows ... What is a good witness

$$CLIQUE = \{ \langle G, k \rangle \mid G \text{ is a graph with a clique of size } k \}$$

k vertices that are all adjacent to each other



has a clique of size 3 and size 4 but not size 5

Thm CLIQUE $\in$ NP

proof witness w for $\langle G, k \rangle \in$ CLIQUE is a set of k vertices in G that forms a clique. A verifier can check in time polynomial in $|\langle G, k \rangle|$ that w has k vertices and every pair of vertices in w is an edge in G.

Def. NP-hard $= \{ L \mid$ if L can be decided in poly time then all languages in NP could be decided in poly time $\}$

$= \{ L \mid$ if $L \in P$ then $NP = P \}$

Thm SAT $\in$ NP-hard (Cook - Levin Theorem) 1971 (Boston Univ; Toronto)

proof We won't prove this here. The proof is intricate.

Def. L is NP-complete if and only if

① $L \in NP$ and ② $L \in$ NP-hard.