P versus NP : a millennium problem.

P : the collection of problems that can be solved by a computer in polynomial time.

NP : the collection of problems that can be solved by a computer in non-deterministic polynomial time.

EXP :

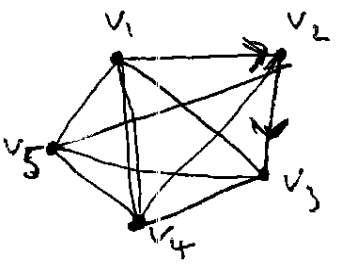
Certainly, $P \subseteq NP \subseteq EXP$ Problem : $P = NP$? $P \neq EXP$

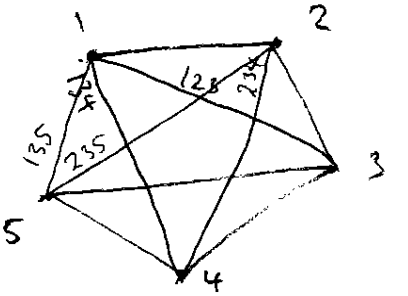
Consequence of $P = NP$: there are polynomial-time algorithms to break every encryption system currently in use.

Consequence of $P \neq NP$: our current computer security systems will last a bit longer.

Example of a problem that can be solved in polynomial time.

REACHABILITY : Given a ~~directed~~ graph G ~~with~~ on n vertices $V = \{v_1, \ldots, v_n\}$, and given two vertices v_i, v_j, is there a path from v_i to v_j?





~~Algorithm~~ Input: represent the graph to the computer by its adjacency matrix $A_G = (a_{ij})$

$a_{ij} = \begin{cases} 1, & \text{if there is an edge from } v_i \text{ to } v_j \\ 0, & \text{otw.} \end{cases}$

$$A_G = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

Algorithm: generate the adjacency matrix for the graph $\tilde{G}$ which satisfies: if there is an edge from v_i to v_j and there is an edge from v_j to v_k, then add an edge from v_i to v_k.

Claim: There is a path from v_r to v_s in G if and only if there is an edge from v_r to v_s in $\tilde{G}$.

"$\Leftarrow$" an edge comes into existence in $\tilde{G}$ from a path in G.

"$\Rightarrow$" if there is a path from v_r to v_s then at every intermediate node v_i, add an edge from v_r to v_i; eventually get an edge to v_s.

Thus: there is an edge from v_r to v_s in $\tilde{G}$ $\Leftrightarrow$ for $A_{\tilde{G}} = (b_{ij})$ $b_{rs} = 1$.

lw

Complexity of the algorithm:

for each v_j (n of them)

look at v_i for $i \neq j$ $(n-1)$

and at v_k for $k \neq j$ and $k \neq i$ $(n-2)$

If $a_{ij}=1$ and $a_{jk}=1$ then set $a_{ik}=1$ 3 operations, say.

if not, do nothing

Thus complexity of algorithm is $C = n(n-1)(n-2)3$

for n large, C is ~~about~~ approximately $3n^2$;

we say C is $O(n^3)$.

Example of a problem that can be solved in ~~polynomial~~ exponential time.

HAMILTONIAN CYCLE: Given a directed graph, is there a cycle that passes through each ~~node~~ vertex exactly once?

Algorithm: choose any vertex to start (n ways), a_1

choose any other vertex $(n-1)$ ways a_2

$\vdots$

last vertex (1) a_n

Check if $a_1 a_2 \ldots a_n$ is a path ie. each a_i is connected by an edge to a_{i+1} $(n-1)$.

Thus complexity $= n(n-1)\ldots(1)(n-1) = n!(n-1)$.

Stirling's Formula: $n! \sim \sqrt{2\pi n}\,\frac{n^n}{e^n} = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$

Thus certainly $> 2^n$, for $n > 2e$, which defines exponential time.

This algorithm involves a search; notice that checking whether the guess is a solution takes time $n-1$ i.e. linear time.

~~This is what characterizes an NP problem:~~

~~A problem is NP if the~~

there is an ~~is~~ non-deterministic polynomial time algorithm to solve this problem:

- guess the correct sequence of vertices $a_1, a_2 \ldots a_n$
- verify this answer is correct - time $n-1$.

How do you guess? - that is what makes the algorithm non-deterministic.

$$P \subseteq NP \subseteq EXP \qquad P \subsetneq EXP$$

Prime Factoring: Given N, find its prime factors.

Alg: search all primes p less than $\sqrt{n}$, check if n/p is an integer

Travelling Salesman: Given n cities, is there a route of length less than B (budget).