

Efficient Certifiers and NP.

An efficient certifier/verifier for a decision problem X is a polynomial time algorithm B that takes two inputs s and t , and returns yes/no. It should have the property that for some polynomial $f_n \in P$,

(1) If s is a yes instance of X , there must exist a t , with $|t| \leq f_n(|s|)$ for which $B(s, t)$ outputs yes.

(2) If s is a no instance of X , $B(s, t)$ outputs no for every t .

End of Definition.

Think of B in the following way:

If $B(s, t)$ outputs yes, then B is saying "Yes, I accept t as a witness that s is a yes-instance of X ."

If $B(s, t)$ outputs no, then B is saying "No, I do not accept t as a witness that s is a yes-instance of X ."

1. 3CNF-SAT has an efficient certifier.

s is the input 3CNF-SAT formula.

~~t expect~~ B expects t to be a satisfying assignment for s .

B simply checks if t is a satisfying assignment for s , and outputs yes if it is and no if it is not.

Clearly B runs in time polynomial in $(|S| + |t|)$.

Now suppose s is a yes-instance of 3CNF-SAT. This means s has a satisfying assignment. If we let t

be this satisfying assignment, then
(a) $|t| \leq |S|$. (^{So P can be defined as $p(n)=n$.})

(b) $B(s, t)$ outputs yes.

Now Suppose s is a no-instance of 3CNF-SAT.

Then no assignment that satisfies s . Clearly, $B(s, t)$ outputs no

on every s .

We'll describe efficient certifiers for several other problems. We will describe the action of certifier B . The other ~~det~~ properties are easily checked.

IND-SET.

s is a pair $\langle G, k \rangle$: the question is, does G have an independent set of size $\geq k$. B checks if t ~~is~~ is an independent set of size at least k , and outputs "yes" if and only if it is.

VERTEX-COVER.

s is a pair $\langle G, k \rangle$: the question is whether G has a vertex cover of size at most k . B checks if t is a vertex cover of size at most k , and outputs "yes" if and only if it is.

SET-COVER.

s is of the form $\langle U, \{S_1, S_2, \dots, S_m\}, k \rangle$ where U is a ground set, and S_1, \dots, S_m are subsets of U . The question is

whether there exist at most K ~~satisfies~~
subsets whose union is U .

B_{spec} checks if t ^{specifies} encodes at most
 K subsets from $\{S_1, S_2, \dots, S_m\}$
whose union is U , and ~~accepts~~
outputs "yes" iff the check passes.

COMPOSITES.

An instance of this problem is a
binary encoding x of a natural number.
We want to know if x is composite,
that is, whether there is a natural number
 y different from 1 and x that
properly divides x .

B checks if t is a number
different from s and 1 that divides
 s , and outputs "yes" if and only
if it is.

Here is a problem that is not known to have an efficient verifier.

3-CNF-UNSAT: Given a 3CNF-formula ϕ , output yes iff ϕ does not have a satisfying assignment.

As a decision problem, this is the same as 3CNF-SAT, except that yes and no instances are interchanged. But note that the defn. of an efficient verifier is asymmetric with respect to yes and no instances.

It is not clear that 3CNF-UNSAT has an efficient verifier. What would be a short witness that a formula ϕ is not satisfiable?

Another natural problem not known to have an efficient verifier is whether the first player has a winning strategy in a generalized geography game. This game will be described in class.

Now we define the class of problems NP:

NP is the set of all decision problems that have an efficient verifier.

Thus, 3CNF-SAT, IND-SET, VERTEX-COVER, SET-COVER, COMPOSITES are all in NP.

Fact. If a decision problem X is poly-time solvable, then $X \in NP$.
Why? B simply ignores its second input t and checks in poly-time if s is a yes-instance of X , and outputs yes iff it is.

(a) If s is a yes-instance, B outputs yes for every t .

(b) If s is a no-instance, B outputs no for every t .

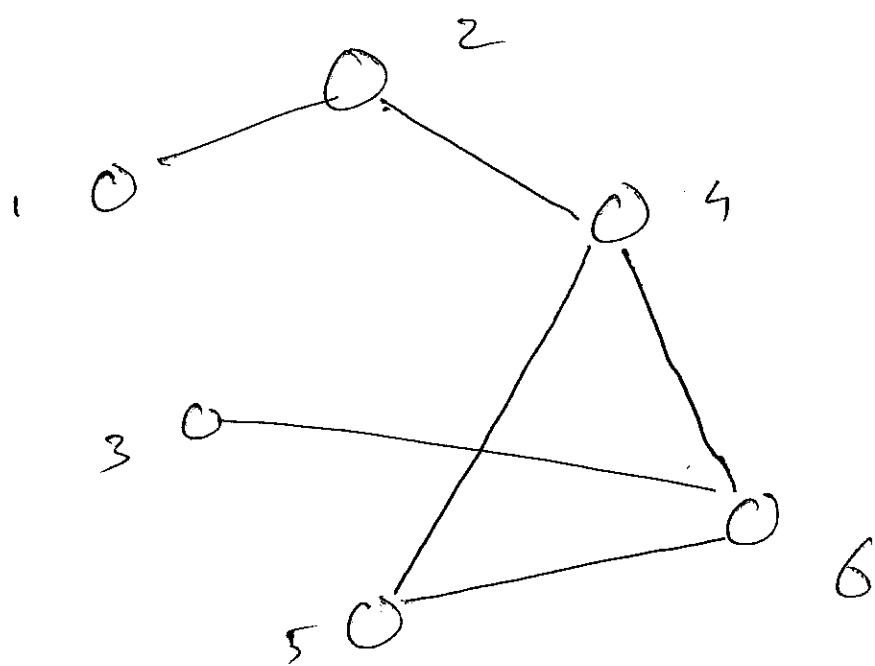
So if we define P to be the set of all decision problems that are poly-time solvable, then $P \subseteq NP$.

The question is, is $P = NP$ or not? If a decision problem has an

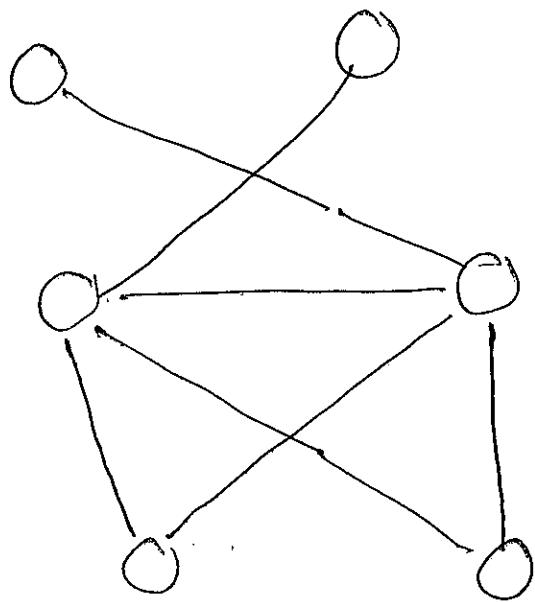
efficient viafa, is it also poly-time
solvable?

It seems unlikely. But here is a
good example to ^{help} appreciate this question
better.

In a graph $G = (V, E)$, a perfect
matching is a subset $M \subseteq E$ so that
each vertex $u \in V$ is incident on exactly
one edge from M .



In this graph, $M = \{(1,2), (3,6), (4,5)\}$
is a perfect matching.



This graph does not have a perfect matching.

In the ~~PERFECT-MATCHING~~ problem, we are given a graph G and asked if it has a perfect matching.

If it is easy to see that ~~PERFECT-MATCHING~~ is in NP. (What is an efficient certifier for it?)

Is ~~PERFECT-MATCHING~~ also in P?

It seems unlikely on the surface,
but the answer is actually yes.

There are several other such examples.
Take ~~any~~ the decision version of any
of the problems we solved by
a greedy algo or by dynamic
programming.

It is easy to see that these problems
have efficient verifiers and are thus
in NP. We also know that
all these problems are in P -
but for nontrivial reasons - either
~~the~~^a greedy algo worked, or ^{the}
problem was amenable to dynamic
programming.