Cpts 515. 11-13-2020

---

nondet. poly-time alg.

Def. of NP: the class of problems that can be solved by nondet. Algs in poly-time.

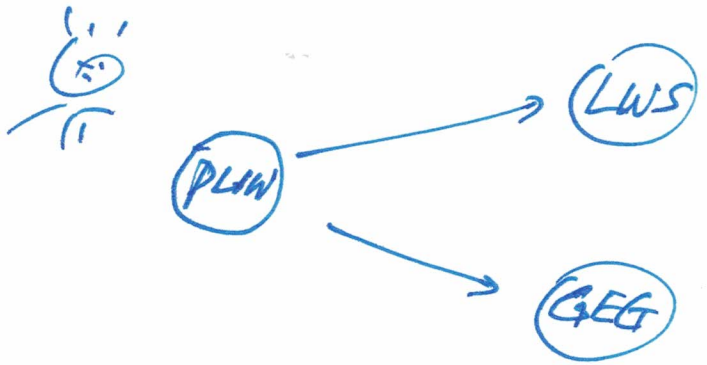
Two way to define those algs.:

① Last time: Through TMs.

② using template.

---

(1). Understanding nondet.



Key: ① guess.

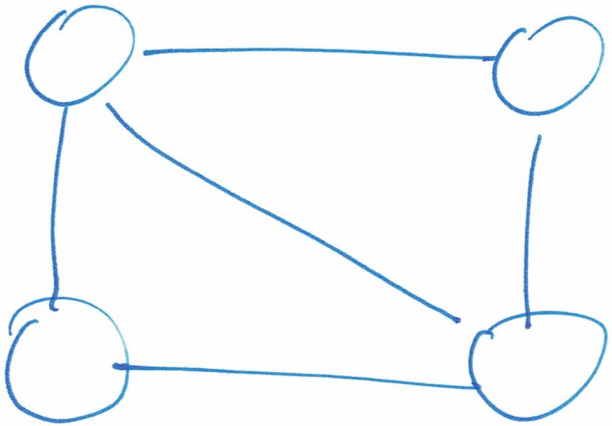
② check or verify

③ nondet. Alg or program runs once.

HC (Hamiltonian Circuit)

Given: a graph G

Question: Is there a walk on G s.t. The walk covers every node in G exactly once?



yes.

(2). template for nondet. poly-time alg.

input x

Guess y with $|y| \le p(|x|)$

if Check(x, y) == okay

return yes

crash

↑ the guessed y is not too long.

// the check(·,·) is det.
// poly-time alg — our every
// day alg, r_{ij} is time $f(|x|+|y|)$.

// you don't return NO here.

// Nondet. Alg never says
// NO.

Show: HC $\in$ NP.

Proof. We need only to design a nondet. poly-time alg. to solve HC:

input G

Guess a walk w s.t. $|w| = n$ where n is # of nodes in G;

Check: w covers every node in G exactly once.

if the check is okay,

ret yes.

Crash.

takes poly-time

// What if the guess is wrong? if there is a correct guess,
// the alg will make that guess.
//
//
//

Remark: ① nondet. poly-time alg has det. output!

② it will output yes or stay quiet. (or crash).

it never says NO. However, since it runs in poly-time, so, after this poly time you still couldn't see an answer, it means "NO".

We have many problems that are in NP, including

$$P \subseteq NP.$$

---

cracking-RSA = large number factorization.

(FACT) Given: n

Question: $\exists\, p, q$ s.t. $n = p \cdot q$ and p, q are primes?

Show: FACT $\in$ NP.

Proof.

We need only design a nondet. poly-time alg for FACT:

input n

Guess p, q with $|p|, |q| \leq |n|$

// $|n|$ is size of n.

// $= \log_2 n$

Check:

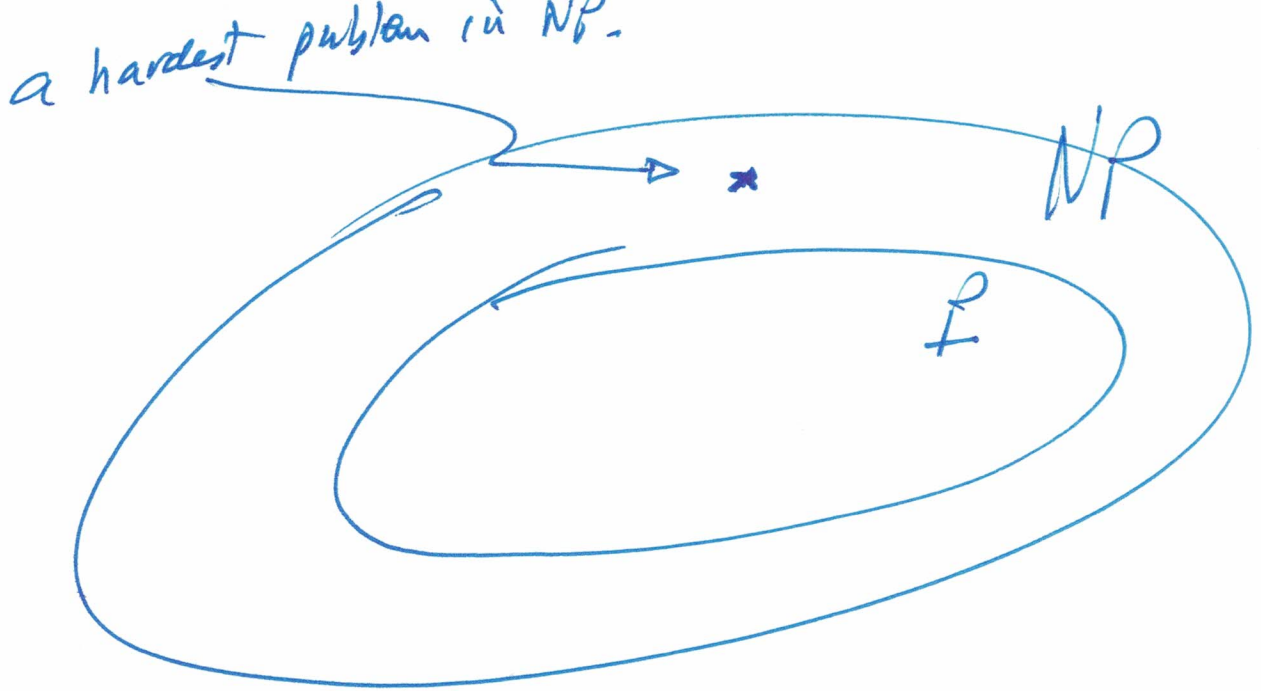
① $n = p \cdot q$ ← det. poly-time.

② p, q are primes. ← det. poly-time. (2006)

if checks is okay,

then yes.

crash.



$P \neq NP$? ⟵ No.1 open problem in CS.

To do this (structural complexity approach to $P \neq NP$):

(1). a math. def of "hardness".

(2). a math. def. of "hardest".

I have two problems

A ≤ B

possible ways to define "≤":

① ~~A~~ takes less time to solve than B.

looking resources used. Exple: your Exam — problem 1 often is easiest.

② Suppose that we have only two problems in an Exam: Prob 1, Prob 2. You know Prob 1 is easier than Prob 2. why?

(a). you can solve Prob 1, but you can't solve Prob 2.

(b). you can't solve Prob 1. you can't solve Prob 2.

go to the Moon

go to the Mars.

©. (Steven. Cook).

"Once you can solve prob2, then, you can solve prob1."

↳ prob1 ≤ prob2.

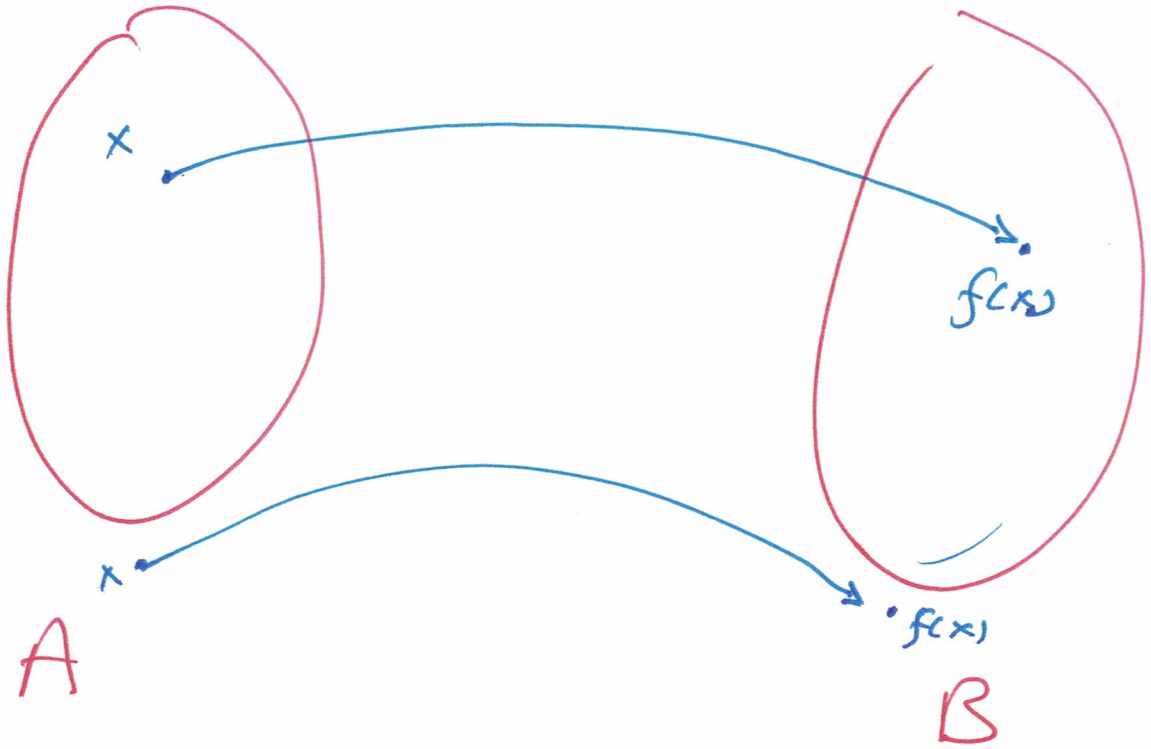
---

5. Cook's reduction (many-to-one reducibility).

(1). problem is a set of strings.

(2). I want to define $A \leq_m B$.

↳ $\leq_m$ = many-to-one



$A \leq_m B$ if there is a mapping f s.t.

(1). f can be computed in poly-time

(2). $\forall x$, $x \in A$ iff $f(x) \in B$.

// hardest problem in NP.

A is NP-complete if

(1). $A \in NP$

(2). $\forall B \in NP$, $B \leq_m A$.

Summary:

(1). How to design a nodet. poly-time alg.
or, to show a problem is in NP.

(2). $P \subseteq NP$.

(3). $\leq_m$.

(4). NP-completeness.