CSL 356 lecture 40 Nov 14

Bellman- Ford $|V| = n$

$|E| = m$

$O(n \cdot m)$ time

Dijkstra's $O((n+m)\log n)$

APSP : Run either BF or Dijkstra's from every vertex

$\Rightarrow O(n^2 \cdot m)$ or $((n+m)n \log n)$

An algebraic approach, where the adjacency matrix A is "multiplied" with itself n times A^n

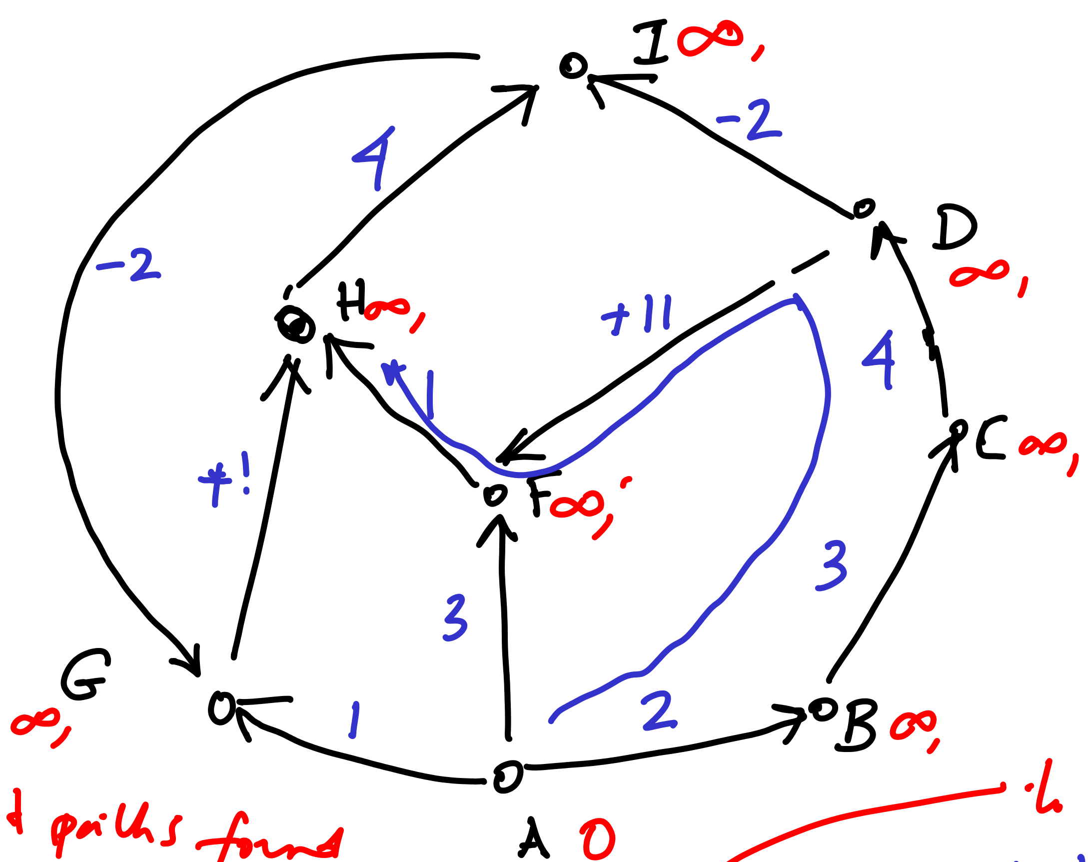
under (addition and min)

$(a_1\ a_2\ a_3 \ldots a_n) \cdot (b_1\ b_2 \ldots b_n)$

$\min(a_1+b_1,\ a_2+b_2,\ a_3+b_3 \ldots a_n+b_n)$

$|V| = n$

$|E| = m$



shortest paths found
a

S : A, G, H

to be found

T :

B	C	D	E	F	G	H	I
∞	∞	∞	∞	∞	∞	∞	∞
2	∞	∞	∞	3	$\boxed{1}$	∞	∞
2	∞	∞	∞	3		0	∞

Correctness of Dijkstra's shortest path algorithm

By induction on the order the vertices get added to S

(It adds the vertices in increasing order of the shortest)

Case 1

x doesn't have the right value of shortest path

Handling -ve cycles in Shortest path problems

Adding a large +ve number to all the weights would affect paths of different (edge) lengths differently.

Is there a path (cycle) connecting all vertices in the graph where every vertex is visited exactly once.

(Euler tour visits every edge exactly once)

→ Hamilton cycle

No polynomial time algorithm is not known for H.C.

→ (as opposed to not possible)

Prover
(Has the secret algorithm)

Verifier
(Must verify if the prover is correct)

Verifying H.C. is easy

" TSP is not known to be easy

Is the given graph 3-colorable?

<u>Opt version</u> : find chromatic no.

NP : ~~not~~ polynomial

<u>non-deterministic</u>

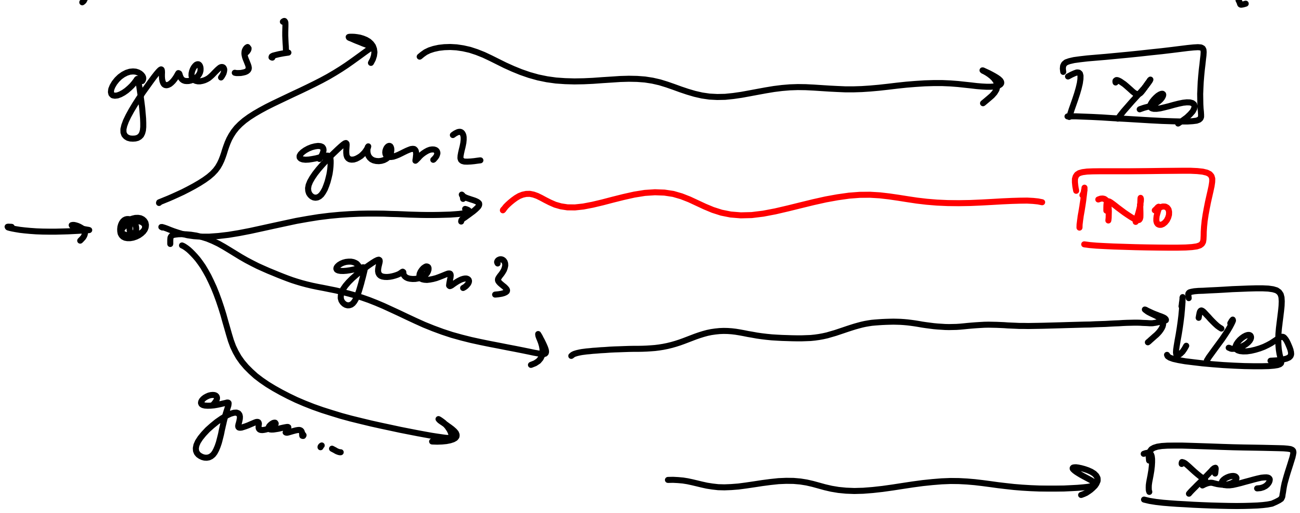
Non det. algorithm

— If the certificate is verified then answer is YES

→ If certificate is not correct can we conclude NO?

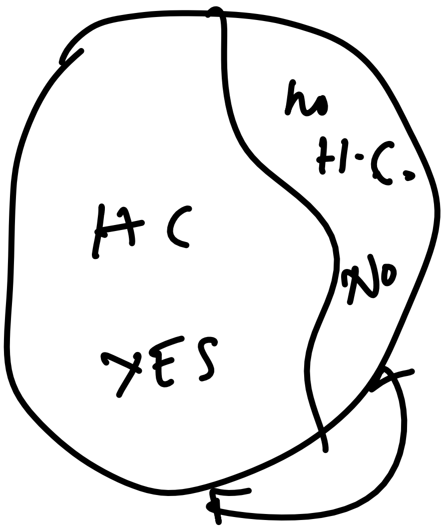
If the given instance has a HC ⟹ there is a short proof / certificate

If there is no HC, then ??



The class of decision problems for which there is a <u>polynomial time verification</u> (when the answer is YES) is called Non-deterministic Polynomial

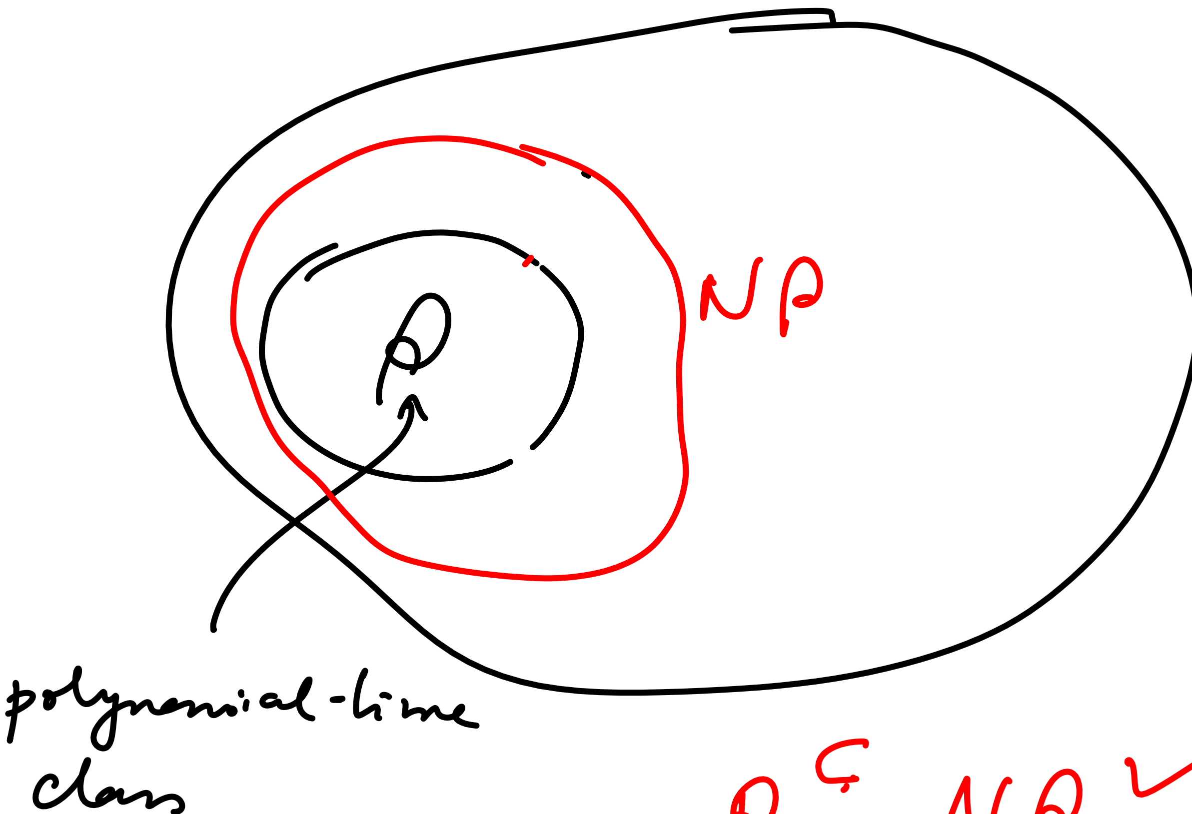
Set of all graphs



Polynomial in size of input

<u>Polynomial Time</u>: Is the class of problems that have an algorithm running in $O(n^i)$ time for some integer i and input size n

Decision Problems



$P \overset{?}{\subseteq} NP$ ✓

$P \overset{?}{=} NP$

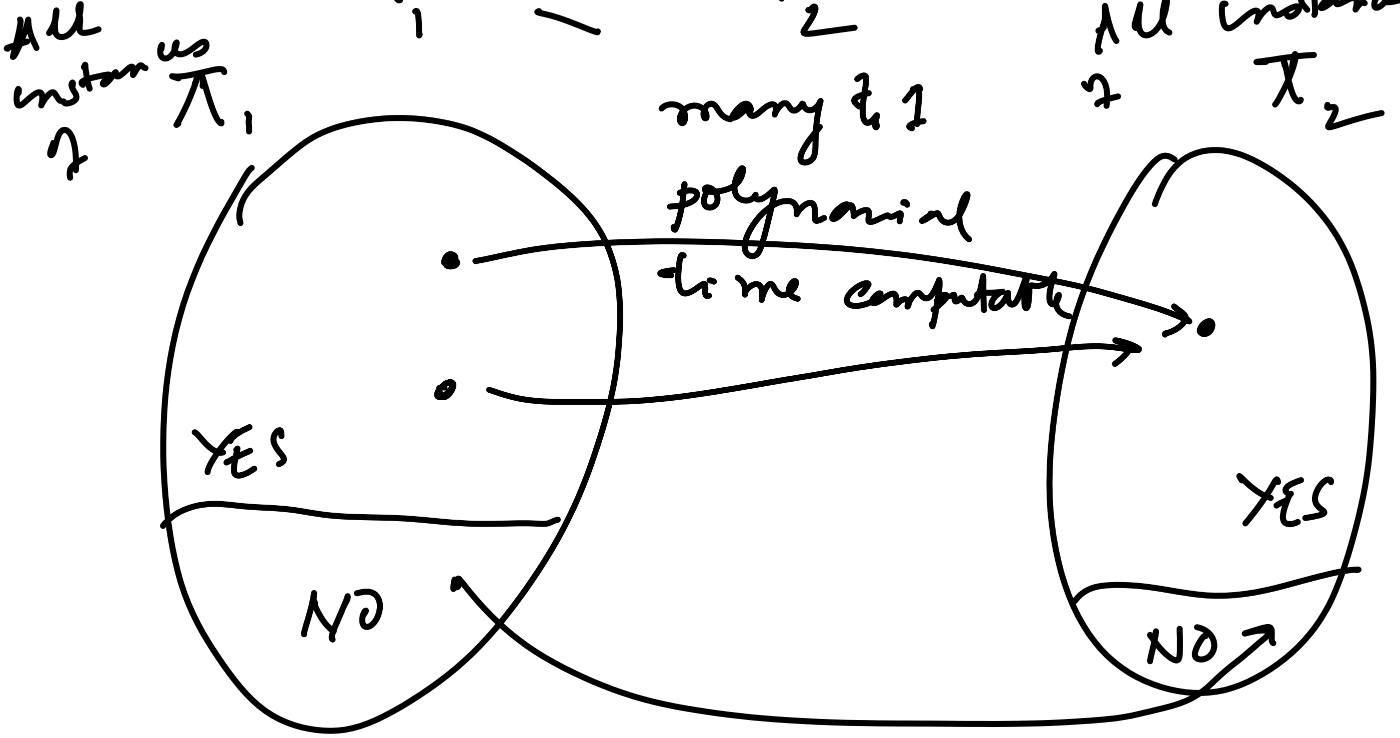
If we can show that H.C. problem cannot (lower bound) have polytime algorithm

$\Rightarrow P \neq NP$ i.e. $\because P \subset NP$

What if there is a polynomial time algorithm for H.C.?

What does it mean for a problem π_1 to be reducible to problem π_2

$$\pi_1 \le \pi_2$$



Given an instance of π_1 of input size n we first run a mapping algorithm that takes, say $g(n)$ time where g is a polynomial function and produces an instance $f(x)$ $|x| = n$

We run the algorithm for π_2 on the instance $f(x)$ that takes time $q(|f(x)|)$: q is some function corresponding to the running time of π_2

So the overall algorithm for π_1 takes $g(n) + q(|f(x)|)$

If q is a polynomial function then this is polynomial time since $|f(x)| \leq g(n)$

Reducibility is a relation (π_1, π_2) $\pi_1 \leq \pi_2$

polynomial time reducibility is a special kind of reducibility and we write $\pi_1 \leq_{poly} \pi_2$

If $\pi_1 \leq_{poly} \pi_2$

and $\pi_2 \leq_{poly} \pi_3$

$\Rightarrow \quad \pi_1 \leq_{poly} \pi_3$ (transitive)

since composition of polytime function is polynomial time.

<u>NP-complete problem</u>:

π is NPC if ① π is in class NP

② All problem $P \in NP$ are reducible to π in polynomial time

$\Rightarrow$ If π is NPC and there is a poly-time algorithm for π then $P = NP$