P: problem with a polynomial time algorithm

NP: polynomial time verification

→ given a solution, verify that it is correct in polynomial time

class of decision problems $\begin{cases} \text{yes} \\ \text{no} \end{cases}$ ≡ does there exist an optimal solution $\leq v$?

opt − $\underline{\underline{X}}$ ⟷ $\boxed{\text{d} - \underline{\underline{X}}}$

TSP TSP

NP − Complete (NPC): subset of NP

A problem L is NP − complete iff

1) $L \in NP$

2) all other problems in NP, say L', can be reduced to L

$$\boxed{L' \longrightarrow \underline{\underline{L}} \quad \forall \; \underline{\underline{L'}} \in NP}$$

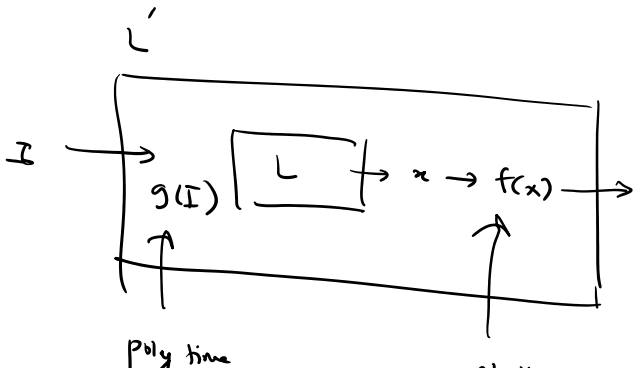
(if L can be solved in P time then P = NP)

Q: how to show that a problem L is NPC / NP − complete?

<u>Approach 1</u>

look at <u>all</u> $L' \in NP$

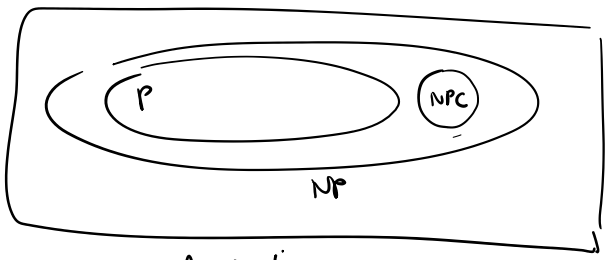
and then show that $\underbrace{L' \longrightarrow L}$



∃ a function f, computable in polynomial time such that f(x) is a solution to L' if x is a solution to L

on an instance I that can also be instance

on an instance I that can also be converted to instance $g(I)$ for L in poly time



most likely relationship

$P \subseteq NP$ ✓

$P \cap NPC \overset{?}{=} \phi$

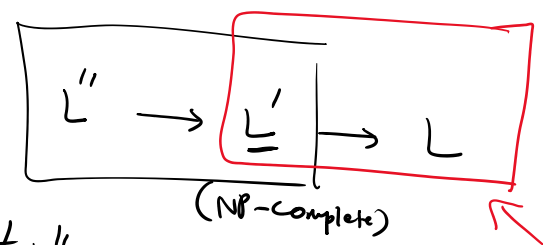
$P \overset{?}{=} NP$

do not know

Approach 2 : Is L NP-Complete?

Start from some known NP-Complete problem, say L'

just show that $L' \rightarrow L$ and L is in NP



just show this

$\forall L'' \in NP$

from the definition of NP-Complete

1st NP-Complete problem

Circuit SAT : circuit satisfiability

SAT problem: satisfiability problem

Input: a Boolean expression comprising of three logical operations (AND, OR, NOT)

Input: a <u>Boolean expression</u> comprising of three logical operations

(AND , OR , NOT)

$\wedge$ $\vee$ $\bar{\ }$

<u>Task</u>: check whether there exists a <u>satisfying assignment</u> or not

What is a Boolean formula / expression

1) Variables : x, y, z, etc

can be either 0 or 1

2) <u>literal</u> : either a variable x

or its negation $\bar{x}$

3) clause : OR of literals (disjunction)

$$(\underbrace{x} \vee \underbrace{\bar{y}} \vee \underbrace{z}) \equiv x \text{ OR } (\text{NOT } y) \text{ OR } z$$

4) expression : AND of clauses (conjunction)

$$(x \vee \bar{y} \vee z) \wedge (y \vee w) \Longleftarrow \text{CNF}$$

Conjunctive Normal Form

2 clauses

length : sum of all the clause lengths

: $3+2 = 5$

DNF:
disjunctive Normal Form

$$\underbrace{(\underbrace{x \wedge \bar{y} \wedge z}_{\text{AND}}) \vee (x \wedge \bar{w})}_{\text{OR}}$$

<u>Satisfying assignment</u>

an assignment of a 0 or 1 to each variable so that the boolean expression is <u>true</u>

$$I : \underbrace{\overset{C_1}{(x \vee y \vee \bar{z})} \wedge \overset{C_2}{(\bar{x} \vee \bar{y})} \wedge \overset{C_3}{(y \vee z)}}_{\text{AND}} \quad \text{CNF}$$

AND ⇒ every clause has to be true

OR

at least one literal has to be true

Algo 1 for SAT (try all possible assignments)

x	y	z	$\bar{x}$	$\bar{y}$	$\bar{z}$	$x \vee y \vee \bar{z}$	$\bar{x} \vee \bar{y}$	$y \vee z$	full expression
0	0	0	1	1	1	1	1	0	0
0	0	1	1	1	0	0			0
0	1	0							

2^N where N is # of variables

L: length of expression

checking a solution take $O(L)$ time

$O(L \cdot 2^N)$ Complexity of naive / brute-force method

exponential

Circuit SAT

↳ hardware version of SAT

Any problem in NP has a $\boxed{\text{digital circuit}}$ that encode it

such that ∃ a solution to the NP problem if circuit SAT has a solution

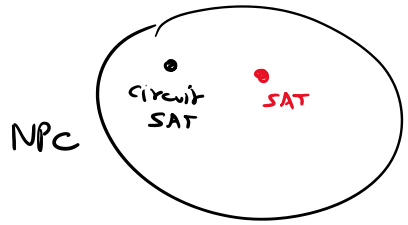
digital circuit must have a size polynomial in the input I

digital circuit must have a <u>size polynomial</u> in the input I

Circuit SAT is NP complete

1) Circuit SAT $\in$ NP $\leftarrow$ poly time verifiable

2) $\forall L' \in NP \quad L' \rightarrow L$



$\underbrace{\text{circuit SAT} \rightarrow \text{SAT}}$ ✓

Hamilton Path (Rudrata): given <u>a graph</u>, does there exist a path that visits each vertex only once starting from <u>s</u> and ending at <u>t</u>

Hamilton Cycle / Rudrata cycle: Given a graph, does there exist a path starting from <u>s</u> that visits all other vertices only one and ends at <u>s</u>

It is known that Rudrata path $\in$ NPC.

Show that Rudrata cycle is also NPC

show that $\boxed{\text{Rudrata Path} \rightarrow \text{Rudrata cycle}}$

1) Rudrata cycle $\in$ NP

$O(|V|)$ time to verify that a solution is correct.

a) length has to be n, $|V| = n$

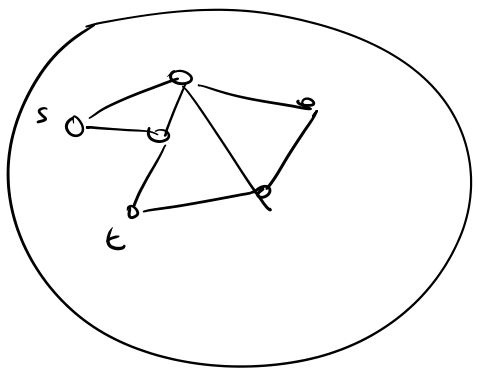
<u>given a solution</u>

b) count of the vertices $\Big\}$ $O(n)$ time

2) a solution to path if solution to cycle

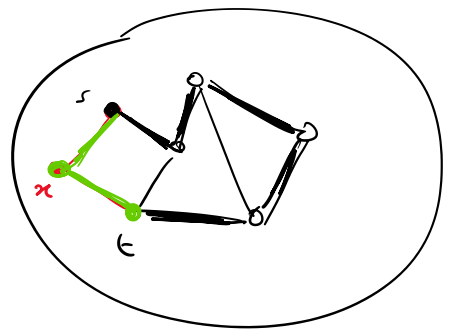
Path

G



Cycle

G'

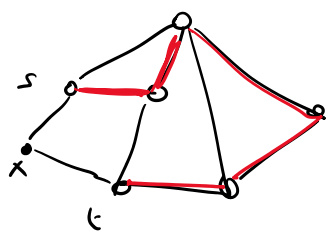


1) if $\exists$ a s-t path that visits all vertices exactly once $\Longrightarrow$ show $\exists$ a cycle that visits each vertex only once

if path then cycle

2) show there is a path $\Longleftarrow$ $\exists$ a cycle

delete x then the remaining portion of the cycle is a path in G $\longleftarrow$



G'

Show that TSP is NP-Complete

1) TSP $\in$ NP ($\underline{\underline{d}}$-TSP)

given a solution s, and a bound b,

$O(|V|)$ to verify the cost of the tour, then check if it is $\leq b$.

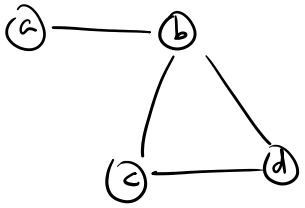
yes no

[G: Complete graph, undirected, weighted
d_{ij} is the weight on edge (i,j)]

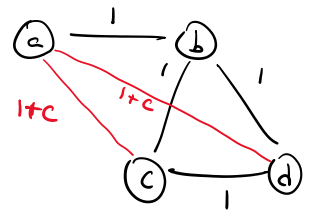
d_{ij} is the weight on edge (i,j)] Y/s no

2) $\underbrace{\text{Hamilton / Rudrata cycle}}_{\in NPC} \longrightarrow TSP$

Rudrata cycle (G)



$TSP(G')$ d_{ij}



original edges $d_{ij} = 1$

new edges $d_{ij} = 1+c$

$c \geq 1$

a) transformation is $O(n^2)$ in the worst case

a rudrata cycle in G $\underset{\text{if}}{\Longleftrightarrow}$ a TSP tour in G' with cost $\leq \underbrace{b = |V| = n}$

1) rudrata cycle $\rightarrow$ TSP with cost $\leq n$

$s \xrightarrow{1} v_1 \xrightarrow{1} v_2 \xrightarrow{1} \cdots \rightarrow v_{n-1} \rightarrow s$

rudrata cycle in G

$\Rightarrow$ tour in G', cost is exactly n

2) If there is no rudrata cycle in G, then $\nexists$ a tour in G' with cost $\leq n$

$\Downarrow$

we cannot visit all the nodes in G exactly once,

however $\exists$ tour in G', but the tour will have to use at least one of the new edge

$\Rightarrow$ cost of the tour is at least

one of the new edge

$\Rightarrow$ cost of the tour is at least

$$\text{cost} \geq \underbrace{(n-1)}_{\text{Other edges in the tour}} + \underbrace{(1+c)}_{\text{one new edge}}$$

$$\text{cost} \geq n+c$$

Any TSP tour has $\underbrace{\text{cost} \leq n \quad \text{or} \quad \text{cost} \geq n+c}_{\text{gap} = c}$

we can choose c to be as large as we want

Show that TSP has no polynomial time approximation

---

opt-TSP: minimization problem

least cost or min cost tour

given G, $\text{opt}(G)$ is the value of the optimal cost on G

Given A, approximation algo for TSP (polytime: greedy algo $\rightarrow$ always pick the closest unvisited city)

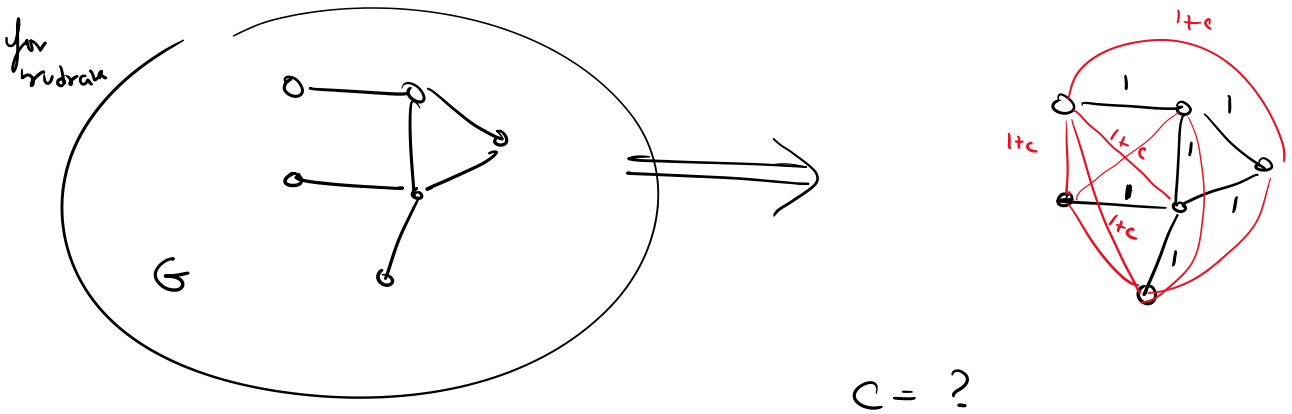
$A(G)$ is the cost of the solution using A on G

$$\alpha_A = \max_G \frac{A(G)}{\text{opt}(G)} \qquad \frac{100}{10} = 10$$

approx ratio for A

Connect rudrata cycle problem to the gap c

Connect hudrata cycle problem to the gap $\underline{\underline{c}}$



$c = ?$

Assume that $\exists$ an Approx algo with approx ratio α

$$\alpha = \max_G \frac{A(G)}{opt(G)}$$

Choose $\underline{c} = n \cdot \alpha$

1) then $\exists$ a hudrata path, if TSP tour of cost n

$$\Rightarrow \alpha = \frac{A(G)}{\underbrace{opt(G)}_{n}} \Rightarrow A(G) = n\alpha$$

2) then $\nexists$ hudrata path if $TSP \geq n + c = n + n\alpha$

$$= n(1+\alpha)$$

$$\Rightarrow \alpha = \frac{A(G)}{n(1+\alpha)} = A(G) = n\alpha(1+\alpha) > n\alpha$$

$A(G)$ run in poly time

Contradiction unless $\underline{\underline{P = NP}}$

$\Rightarrow$ poly time approx algo cannot exist