max independent set - set of vertices in a graph s.t no 2 vertices in the set are adjacent

classmate Date Page

Theory of NP completeness

Can we talk about what are "easy" & "hard" problems? We restrict ourselves to "decision problems" (problems that have a yes/no answer).

Why is this not a restriction?

a) For many problem, if we can solve the decision version, we can solve the original problem.

Ex: Find the size of the max independent set.

Decision version - Is there an independent set of size $\geq K$ for a given K? → yes.

b) If the decision version is "hard", then the original problem also is.

↳ SAT (satisfiability)

Ex: Given a boolean formula, find an assignment of 0/1 to its variables to satisfy it (or say no such assignment is possible).

Decision version - Given a boolean formula, does there exist an assignment of 0/1 to satisfy it?

"easy" problem - problems for which deterministic poly. time algorithms exist.

→ Is n^{100} OK?

Theoretically yes, practically no. But for all problems, either we know an algorithm with low degree polynomial (n^2, n^3) or we don't know any poly time algorithm.

"Hard" problem

Let Π be a problem.

Let I = set of all instances of Π

Accept (TT) : Subset of I with o/p "yes".
Reject (TT) : Subset of I with o/p "no".

"Certificate" : a string (set of an instance) which is an evidence ~~of the "yes" answer~~ in support of a yes answer.

Every instance in Accept (TT) must have a certificate.
No instance in Reject (TT) will have a certificate

Verifier / Certifier - an algorithm that can verify the certificate.

Class NP
The set of problems for which a polynomial time verifier exists (set of problems for which a certificate can be verified in polynomial time).

Ex: Compositeness check.
Instance : an integer n
Certificate : a divisor of n
verifier : n/divisor gives remainder 0.

Ex: Hamiltonian Cycle Problem
Instance: Graph $G = (V, E)$
certificate : a permutation of vertices V
verifier : easy to write.

Hamiltonian path -> a path which visits each vertex exactly once
Hamiltonian Cycle -> a cycle which visits each vertex exactly once

NP

P

An obvious result

$\Rightarrow P \subseteq NP$ [is P = NP? million $ question]

Reduction

Reduction of Pr A to Pr B.

A mapping of every instance of A to some instance of B such that A had a "yes" answer if and only if B has a "yes" answer

(B yes $\Rightarrow$ A yes, B no $\Rightarrow$ A no)

Polynomial time reduction:

The mapping can be done in polynomial time.

Ex HAM PATH $\leq$ HAM cycle

Instance of HAM PATH : $G = (V, E)$ $s, t \in V$ s.t $s \neq t$

Create instance of HAM cycle

$G' : (V \cup \{u\}, E \cup \{(s,u), (u,t)\})$

If path in G from s to t, ∃ a ham cycle

" cycle in G', ∃ a Ham path from

s - u - t must occur successively in the ham cycle remove (s,u) & (u,t)

∃ Ham path in G.

max independent set
Hamiltonian cycle
Hamiltonian path.
Satisfiability (SAT)
} NP complete

classmate
Date
Page

NP - hard

A problem 'A' is said to be NP hard if all problem in NP can be polynomially reduced to A.

NP complete

A problem 'A' is NP complete, if A is NP-hard and A is in NP

Implication

If any NP-hard problem is solved using a poly time algo, all problem in NP can be solved in poly time
⇒ P = NP

→ How to prove a problem is NP-hard?

Start from an existing NP hard problem 2) Reduce B in polynomial time to A.

OR

Prove from scratch

Cook Levin Theorem -

SAT is np-complete ⇒ This was proved from scratch.

3 SAT problems were proved to be NP hard

SAT → Satisfiability

CNF - Conjecture Normal form.

Product of sum

$(x_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2)$

clauses = literals $(x, \bar{x})$ connected by $\vee$

3-CNF = every clause has exactly 3 literals.

eg: $(x_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (x_1 \vee x_2 \vee x_3)$

SAT
CNF-SAT
3-CNF SAT
} all NP complete.

To prove: Max ~~Indep~~ Independent set is NP complete.

Decision Version:

Given a graph G & an integer K, is there an independent set of size = K?

↳ max indep. set

i) MIS is NP. → Given a certificate c, verify if $|c| = K$ & if c is an ind set.

ii) choose 3 CNF SAT as the starting NP hard problem.

Instance of 3CNF SAT

$\phi = \alpha_1 \wedge \alpha_2 \wedge \alpha_3 \ldots \wedge \alpha_d$

each $\alpha_i = (a_1^i \vee a_2^i \vee a_3^i)$

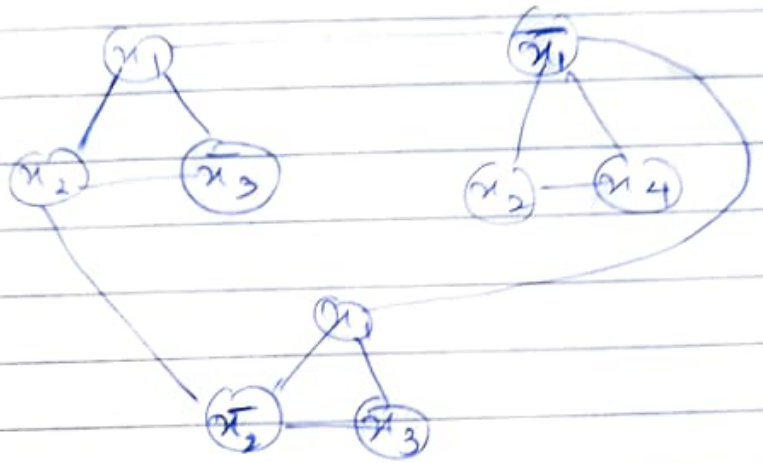
Create an instance of MIS

i) For each clause α_i, add 3 nodes connected to each other, one for each literal

ii) Add an edge b/w 2 nodes in 2 clauses if they are compliments of each other.

Call this Graph G

$\psi = (x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee x_2 \vee x_4) \wedge (x_1 \vee \bar{x}_2 \vee \bar{x}_3)$



To prove:

Max Independent set is NP complete.

To prove:

1) If ψ is satisfiable, $\exists$ an index set in G of size l.

ψ is satisfiable $\Rightarrow$ atleast one literal evaluates to 1 in each clause.

Choose S = nodes in G corresponding to one literal in each clause that evaluates to 1,

$|S| = l$

Now $|S|$ is an independent set.

Ham path $\leq$ Ham cycle

$(G, s, t) \longrightarrow G'$

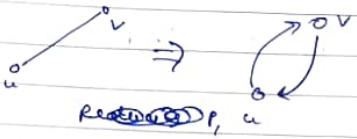
AD

Reducing $P_1 \longrightarrow P_2$.

Undirected Hamiltonian cycles $\leq$ Directed Hamiltonian cycles

$G \longrightarrow G'$

Replace

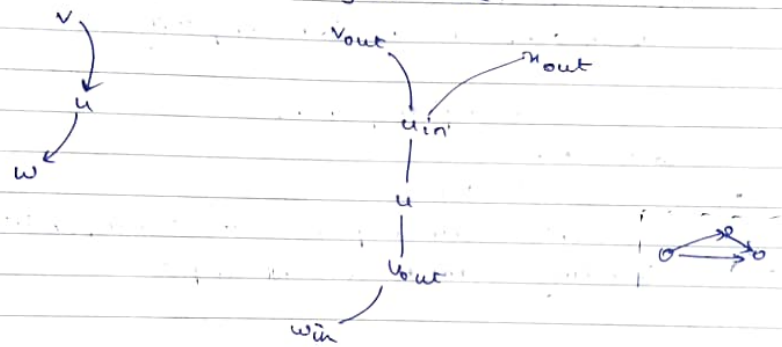


Reducing P_1 u

directed hamiltonian cycle $\leq$ undirected Hamiltonian cycle.

$G \longrightarrow G'$

Reducing $P_1 \longrightarrow P_2$.



A problem Q is called NP hard if $Q' \leq Q$ for all $Q' \in NP$

A problem Q is called NP-complete if

(1) $Q \in NP$

(2) Q is NP-hard.

Theorem: If any NP-complete problem has a poly-time algorithm, then $P = NP$

Proof: $P \subseteq NP$

To prove $NP \subseteq P$

Let Q be a NPC problem having a poly. time algo, take $Q' \in NP$

$\Rightarrow Q' \leq Q$. $\quad I' \longmapsto I$

$Q' \xrightarrow{\text{poly time}} Q$

$I' \longrightarrow I$ in polynomial time

reduction → poly-time } ⇒ poly time.
Q → poly time }

Q' → poly time → $Q' \in P$.

⇒ Reduction

reducing $P_1 \longrightarrow P_2$ if it converts an instance I_1 of P_1 to an instance I_2 of P_2 s.t $I_1 \in$ Accept(P_1) ⟺ $I_2 \in$ Accept(P_2)

iff $P_1 \leq P_2$ (time complexity)

Are there NP complete problems?

~~Cook Levin~~:

⇒ Cook Levin Theorem:

SAT is NP Complete (SAT → satisfiability)

⇒ Corollary: CNFSAT is NP complete

⇒ Satisfiability → $\phi(x_1, x_2 \ldots x_n)$ is a boolean formula in $x_1, x_2 \ldots x_n$. Does there exist a $t_1, t_2 \ldots t_n$ truth assignment of $x_1, x_2 \ldots x_n$ s.t $\phi(t_1, t_2 \ldots t_n) = 1$.

CNF → conjunctive normal form

∴ And of ors ∴

Product of sum

Ex.→ $(x_1 | \bar{x}_2 | x_3)$ & $(x_5 | \bar{x}_7 | \bar{x}_8)$

→ literal → a variable or its complement

→ clause → OR of literals.

→ A formula in CNF is a conjunction (AND) of clauses.

⇒ If each clause has k-literals, then the formula is in k-CNF

ex - $(x_1 \vee x_2' \vee x_4) \wedge (x_2 \vee x_2' \vee x_3') \wedge (x_2 \vee x_2 \vee x_4)$

2 - CNFSAT $\in P$

3 - CNFSAT is NP complete.

Theorem : $Q, Q' \in NP$

$Q \leq Q'$, If Q is NPC, then Q' is NPC

$\Rightarrow$ to prove a new problem Q' is NPC, start with an NPC problem Q, and prove that $Q \leq Q'$

we know 3-CNFSAT is NP complete

1) 3 CNFSAT $\in$ NP

certificate : a satisfying truth assignment.

2) CNFSAT $\leq$ 3-CNFSAT

$\phi \longrightarrow \psi$

Pf:

Take a clause $l_1 \vee l_2 \vee l_3 \dots \vee l_k$ in ϕ

$k = 1$ $\quad l_1 \vee l_1 \vee l_1$

$k = 2$ $\quad l_1 \vee l_2 \vee l_2$

$k = 3$ $\quad l_1 \vee l_2 \vee l_3$

$k > 4$ Introduce $k-3$ new variables

$y_1, y_2 \dots y_{k-3}$

$l_1 \vee l_2 \vee l_3 \dots \vee l_k$ is satisfiable $\Leftrightarrow$ (S1)

$(l_1 \vee l_2 \vee y_1) \wedge (\bar{y}_1 \wedge l_3 \wedge y_2) \wedge (\bar{y}_2 \wedge l_4 \wedge y_3) \dots$

$\wedge (\bar{y}_{k-3} \wedge l_{k-1} \wedge l_k)$

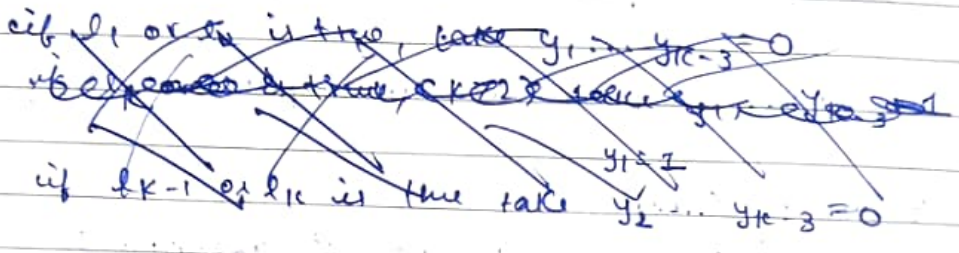
$(l_1 \vee l_2 \vee y_1) \wedge (\bar{y}_1 \vee l_3 \vee y_2) \wedge (\bar{y}_2 \vee l_4 \vee y_3) \wedge \dots$

$\dots \wedge (\bar{y}_{k-3} \vee l_{k-1} \vee l_k)$ (S2)

is satisfiable.

- we need to prove both ways
  that if S1 true → S2 true
  and if S1 false → S2 false

S1 true → S2 true is trivial
if S1 is false, S2 simplifies to →
$y_1 \wedge (\bar{y}_1 \vee y_2) \wedge (\bar{y}_2 \vee y_3) \ldots \wedge (\bar{y}_{k-3})$
which can never be true.

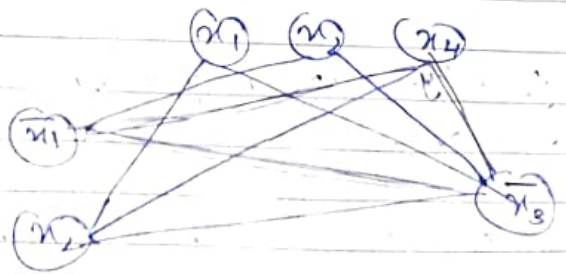


if any general, then choose s.t only l_i is true in l_i clause, then continue on both sides.

clique → size of a set of vertices s.t edges with endpoints at these vertices form a complete graph

converting $\phi = (\bar{x}_1 \vee x_2) \wedge (x_1 \vee \bar{x}_2 \vee x_4) \wedge (\bar{x}_3)$ to an undirected graph for proving NP completeness of the clique problem

eg:



Connect all literals which are.
i) Not belonging to same clause
ii) not complimentary to each other
(contradiction)

if ϕ has k clauses & ϕ is satisfiable

$\Leftrightarrow$

The connected graph has a k-clique

Q To prove DoubleSAT (atleast 2 satisfying assignments) is NP complete.

Ans) DOUBLESAT $\in$ NP

SAT $\leq$ DBLSAT

$\phi(x_1, x_2 \ldots x_n) \longrightarrow \phi(x_1, \ldots x_n) \wedge (y_1 \vee y_2)$

$= \psi(x_1, x_2 \ldots y_n, y_1, y_2)$

ϕ is satisfiable

$\Updownarrow$

ψ is double satisfiable.

Q not all equal satisfiable $\Rightarrow$ atleast one literal true & atleast one literal false.

NAESAT : decide whether not all equal satisfiable exists for CNF. To prove: NAESAT is NP complete.

NAESAT is NP (before checking if satisfied check all not 1 or all not 0)

CNFSAT $\leq$ NAESAT

$\phi(x_1, x_2 \ldots x_n) \longrightarrow \psi(x_1, x_2 \ldots x_n, y)$

for all clause $l_1 \vee l_2 \ldots \vee l_k \longrightarrow l_1 \vee l_2 \ldots \vee l_k \vee y$

ϕ is satisfiable $\Leftrightarrow$ ψ is satisfiable

$\Rightarrow$ Take a satisfying truth assignment of ϕ and $y = 0$ (not all l_i can be 0).

Converse

$\Leftarrow$ $(t_1, t_2 \ldots t_n, 0)$ $\quad 0 = 0\ (t_1, t_2 \ldots t_n)$ satisfies ϕ

$0 = 1\ (\bar{t}_1, \bar{t}_2 \ldots \bar{t}_n)$ satisfies ϕ.

remember, in any clause at least one I & at least one 0 or complement this property satisfied.

Now if for some clause $y = 1$ $\Rightarrow$ at least one t_i in the clause is 0

$\Rightarrow$ $\bar{t}_i = 1$ at least one in each clause.

1. Given 2 graphs G and H decide whether a injective func^n exists $f : V(H) \longrightarrow V(G)$ such that

$(u, v) \in E(H)$ iff $(f(u), f(v)) \in E(G)$

Prove subgraph isomorphism is NP complete.

L - sweeping line

⇒ Problem → Given n line segment, find all intersection points of these lines.

⇒ Naive sol": for every pair of line segments, check intersection. Time complexity $O(n^2) \rightarrow {}^nC_2$ pairs

⇒ Better sol" we propose a sol" of time complexity $O((n+h)\log n))$ where $h \rightarrow$ number of points of intersection of these n lines better if $h < \frac{n^2}{\log n}$

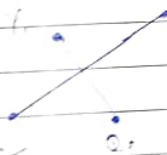
for this all line segments should be in general position -

i) No two endpoints or intersection points of line segment have same x-coord

ii) No L is vertical

iii) No 2 lines are parallel.

$L_i = (P_i, Q_i)$ $P_i \rightarrow$ point with smaller x-coord $Q_i \rightarrow$ point with larger x-coord



sweeping line → goes from $x = -\infty$ to $x = +\infty$

Initial sol":
- Starts from $(x(P_1), x(P_2) \ldots x(P_n))$
- Ends at $(x(P_1), x(P_2) \ldots x(P_n))$

3 types — i) Endpt of segment
ii) Line segment
iii) Intersection

Number of events → $2n+h$

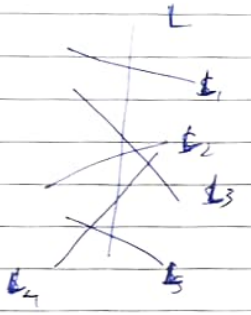
Each event takes $O(\log n)$

Maintain 2 Data structures

⇒ i) BS sweep line information
the set of all active lines sorted from top to bottom

S will not store the y-co-ordinates of intersection of line L with active lines

eg:



S → L_1, L_2, L_3, L_4, L_5

when does S change i) encountered P of a new line
ii) exited from P of a old line
iii) encountered an intersection

ii) A event queue E

i → segment endpoints to the right of current position of L

ii → L_i, L_j are consecutive in S,
$\cap(L_i, L_j)$ → x-coord of intersection point
→ is to the right of L

iii) If i, ii are satisfied $\cap(L_i, L_j)$ is x-coord
is stored in E

E has x-coord of intersection point
belonging to right of sweep line.

Orientation

P_1 to P_1 — P is left
P is right

P_2 to P_1 — P is left
P is right

$P_1 = (x_1, y_1)$
$P_2 = (x_2, y_2)$
$P = (x, y)$

$$side(P_1, P_2, P) = \begin{vmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x \\ y_1 & y_2 & y \end{vmatrix}$$

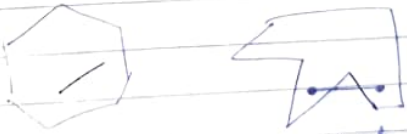
if $side(P_1, P_2, P) > 0$ — P lies left of $\overrightarrow{P_1P_2}$
$= 0$ — lies on P_1P_2
< 0 — P lies to the right of $\overrightarrow{P_1P_2}$

⇒ Simple Polygon → edges intersect only at corners

⇒ not simple →

Convex Polygon

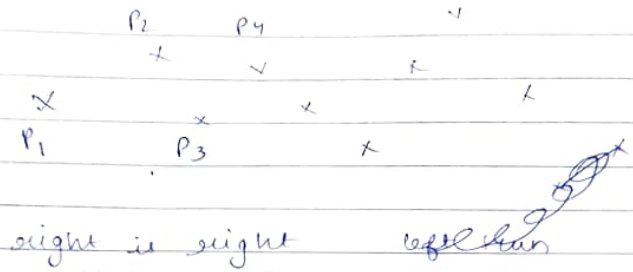
$P \subseteq \mathbb{R}^2$ is called convex if for any P_1 and $P_2 \in R$ the entire line segment P_1P_2 lies in R

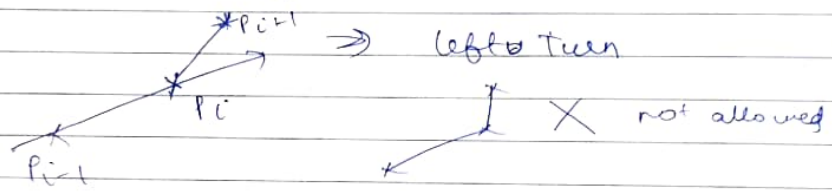


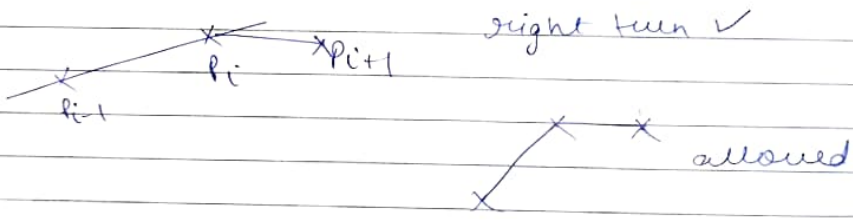
not convex

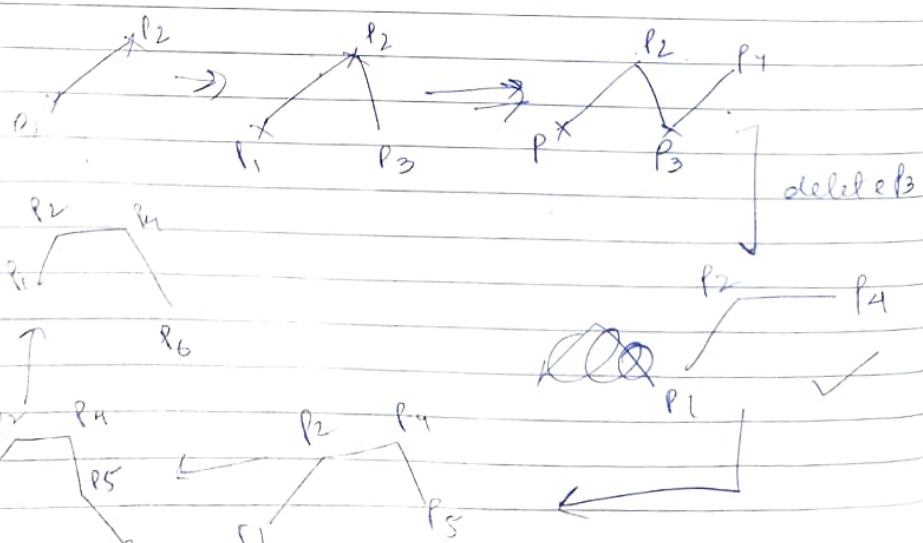
Graham Scan

i) Sort wrt x-co-ord in nlogn time
ii) start with point with least x-cord









Algo T → stack

i) Sort ∞ wrt x-co-ord

ii) push (T, P_1) P_1 → point with least x-co-ord

for i = 1, 2 ... n, repeat: [2nd from top → top

while |T| > 1 and ($P_{prevprev}$, P_{prev}, P_i) is wrong turn

pop (T, P_{prev});

push (T, P_i)]