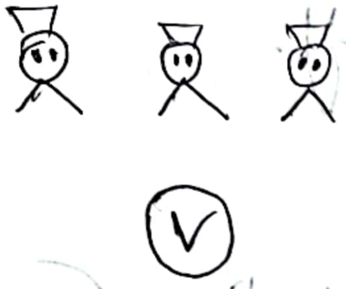
Lecture-01 31.08.24

Algo Engineering

Solve the problem practically.
How to deal with hard problems.



My prob is 'at least' as much as hard as prob V.

↓

polynomial time reduction.

Prob X is 'at least' as hard as prob Y

How?

let, X → Black Box → Output_X

'polynomial' times Black Box use করা যাবে।

Additional computations allowed (bounded by polynomial)

∴ Total complexity

= O(black box complexity) × polynomial + polynomial

If my B.B. solutⁿ complexity ~~is~~ is polynomial, then total =

poly. × poly. + poly.

= polynomial

∴ আমার prob যদি polynomial time এ solve করা যায়, then আসল prob ও poly. time এ solve করা যাবে।

— Reverse is not (or may not be true) hence can't be said.

* Arbitrary Instance: prob instance এর উপর depend করে time complexity.

— Quick Sort এ sorted array input $O(n^2)$

'arbitrary' used as generalization time.

The Qs in Slide

Yes NO

$Y \leq_p X$

prob Y is reducible to prob X
in polynomial time.

Generally, when a prob is hard?

— polynomial time algo পাবার সম্ভাবনা
খুবই ক্ষীণ।

II Low memory algo কই লাগে? Embedded Systems এ

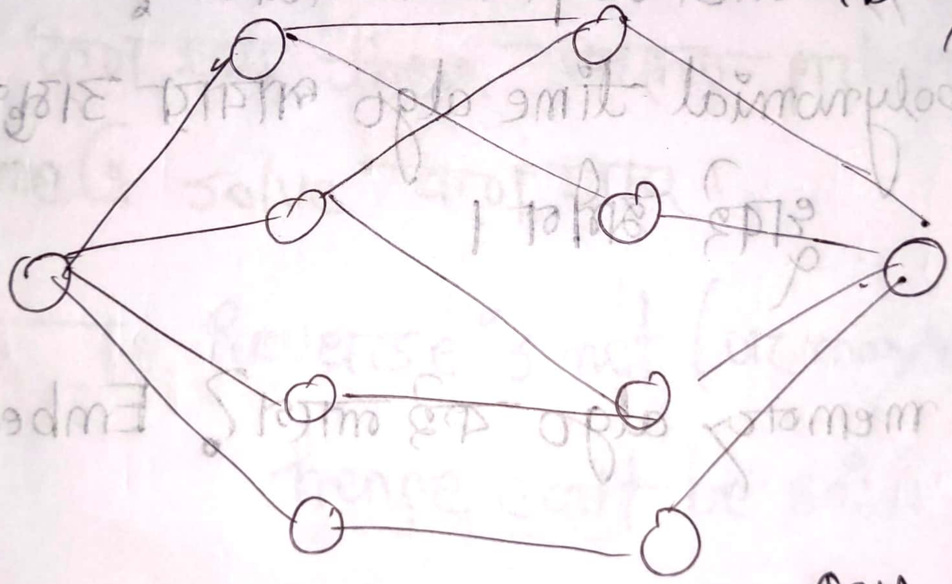
Lecture-02 1/9/24

Why poly. time reduction?
— To compare the hardness of 2 problems.

'satisfiability'

$\boxed{Y \leq_p X}$

* contra-positive statement. $a \Rightarrow b$
$\neg b \Rightarrow \neg a$



$Y \leq_p X$ if X is ~~ab~~, then Y ~~can't~~ can be. . . .

contra-positive: If Y can't be solvet in polynomial time, then X can't be solved . . . .

Lecture - 02 1/9/24

Why poly. time reduction?

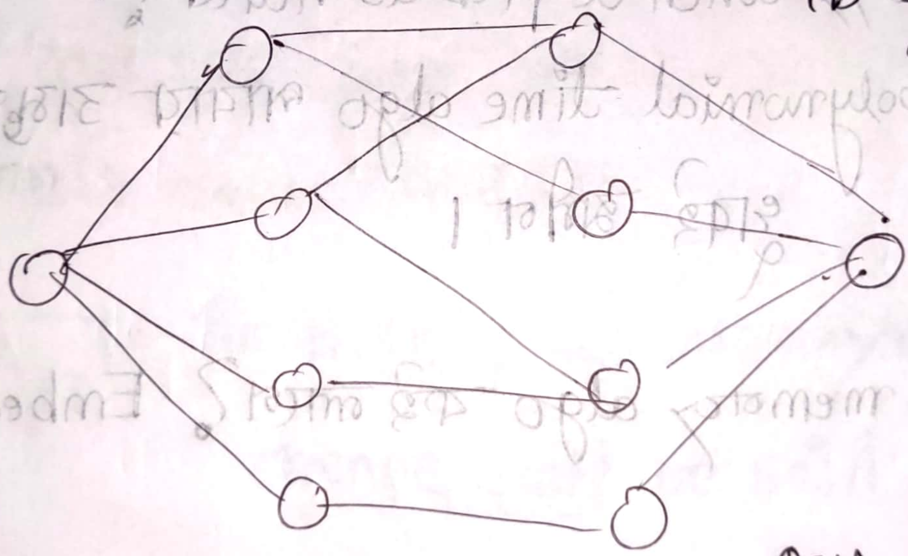
— To compare the hardness of 2 problems.

'satisfiability'

$\boxed{Y \leq_p X}$

* contra-positive statement. $a \Rightarrow b$

$\neg b \Rightarrow \neg a$



$Y \leq_p X$ if X is ~~ab~~, then Y can ~~cannot~~ be . . . .

contra-positive: If Y can't be solved in polynomial time, then X can't be solved . . . . .

Reducing problem Y to X

হতেই হবে Polynomial

Blackbox of X

X এর input → Blackbox of X → X এর output

Y এর input → converter → X এর input

X এর output → converter → Y এর output

∴ Reduction মানে এই 2টি converter design করতে হবে।

for any arbitrary conversion.

① Independent Set Problem

⇕

Vertex Cover Problem.

Indⁿ Set:

— Vertex Set এর Subset

— যেই set এর vertex গুলোর মধ্যে pairwise edge নাই।

Naive Solⁿ: {যে-কোনো 1 টা Vertex}

— We have to $\boxed{\text{maximize}}$ the set.

— The set might not be unique

— set এর size হল unique.

optimization prb
max/min

Decision Version: ... indt set of size of at least K ?

Vertex Cover:

- A subset of Vertex set.
- যে-কোনো edge যদি pick করি, at least 1 টা vertex ∈ Vertex Cover যেন ঐ edge কে cover করে।

Naive Solⁿ: { সবগুলো node }

- We have to minimize this set.

optimization problem।

Equally Hard: $Y \leq_p X$

$X \leq_p Y$.

maxiᶻ → at least K
miniᶻ → at most K

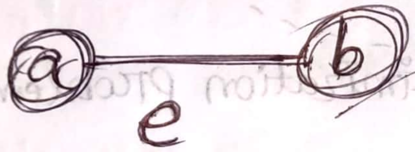
Optimization prob এর → decision Version

~~নিয়ে কাজ করতে হয়।~~

যার Ans yes/no.

Either $Y \leq X$ or $X \leq Y$ যার দেখাই iff prove করতেই হবে। এটা basically বলায় এ input / output, both can be converted.

Step-1: ২ টা prob এর মধ্যে relationship establish করতে হবে।



Necessity: S independent set.

Sufficiency:

V−S vertex cover, then, S ind. set.

proof by contradiction

Contradiction:
V−S vertex cover না
∴ কোনো 1 টা edge আছে যার endpoints V−S এ নাই।
∴ ২ টা endpoints ই S এ আছে।
but তাহলে S independent set ই না।

B.B.

K size at least → [ind set] → at most (n−K) → [V. cover] →

just 1 টা
বিয়োগ হচ্ছে। (polynomial time)

Lecture - 03

Independent Set $\leq_p$ Clique Problem.

Clique Problem: Maximum size এর clique বের করা।

Complete Graph:

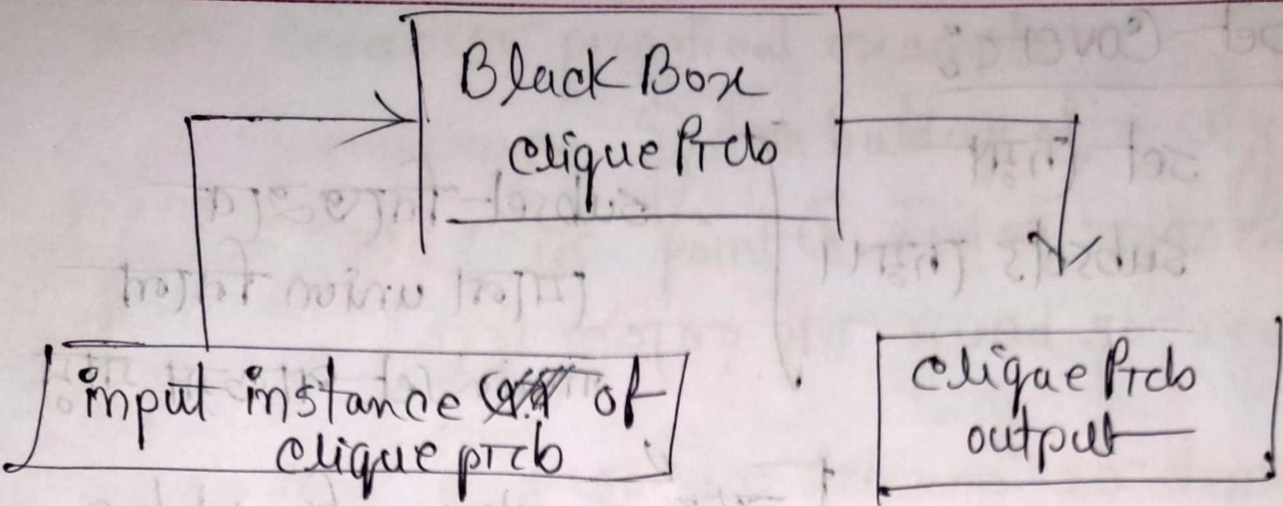
# nodes	1	2	3	4
# name	K_1	K_2	K_3	K_4

Clique: Graph এর Subgraph which is a complete graph.

In a graph,
every vertex is a clique (K_1)
every edge is a clique (K_2)

Decision Version:

input	qs	Ans
Graph G, integer K	at least K size এর clique আছে কি না	Yes NO.



complement of a Graph :

G এর complement $= G'$

$V = V'$

$(E \cup E') =$ complete graph of V.

complement

K_1 K_2 K_3 → cliques

C_1 → • C_2 → • • C_3 → • • • → independent sets.

NULL graph
= edge set empty.

* Time complexity (~~ও~~

→ conversion time (এখানে linear, $O(E)$).

→ Black box কতবার use করতে হচ্ছে।

Set Cover:

set দেয়া
subsets দেয়া। → subset নিতে হবে যেন union নিলে main set পাওয়া যায়।

[This number should be minimized]

Practical Example:

a set of Microsoft Packages;

এই টি package নিলে সব software থাকে পাওয়া।

[S1, S2, S3] → 100$

[S2, S4] → 200$

[S2, S3] → 500$

Indepent set Prob এর practical on:

Register allocation to Variables.

Vertex Cover এর practical example:

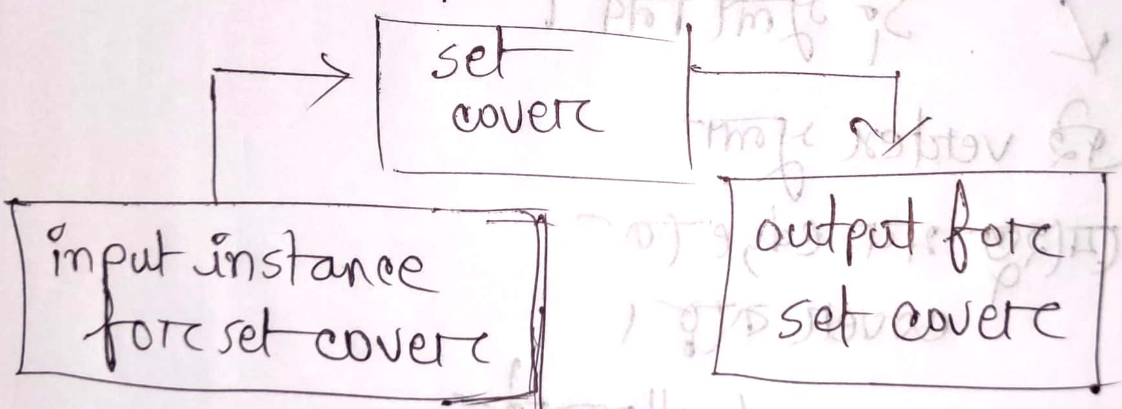
Surveillance System buildup of a city.

at least [কোন কোন point এ device বসাইলেই পুরো শহরের সব road এর security done.

II Clique problem: 9-11 এর পর telecom এর dense subgraph/ complete subgraph গুলো Analysis করা হয়।

criminal রা সবাই সবার সাথে কথা বলায়।

Vertex Cover $\leq_p$ set cover



slide এর graph

$U = \{e_1, e_2, e_3, e_4, \ldots \ldots e_{10}\}$

এরগুলো edge নিয়ে universal set, U.

যতগুলো Vertex আছে ততগুলো subset নিব।

S_1 = vertex 1 এর সাথে যতগুলো edge conn^d

$= \{e_1, e_5\}$

II ~~Necessity~~: Sufficiency

If G has a vertex cover of size at most K, then U can be covered with at most K of subsets $S_1, S_2, \ldots S_m$.

→ Vertex cover নিলাম K size এর।

∴ প্রতিটা edge covered.

সেই vertex গুলা নিচ্ছি তাদের corresponding S_i গুলা নিব।

ঐ vertex গুলা

যেহেতু সব edge কে cover করে।

∴ S_i গুলা build করার যে সব edge touched with vertex V_i is in S_i

∴ ঐ S_i গুলা নিলে আমি সব edge পাবোই। → U.

$Y \leq_p X \longrightarrow$ এটার particular instance.

এটার যে-কোনো instance