Ford Fulkerson — $O(mC)$

Scaling Max-flow — $O(m^2 \log C)$

DEK Algo — $O(m^2 n)$

Push-Relabel Algo — $O(n^2 m)$

Goldberg and Tarjan — $O(n^3)$

J. B. Orlin (STOC 2013) — $O(mn)$

Chen et. al. (FOCS 2022) — $O(m^{1+o(1)})$ — small 'o'

Lecture 20 (25/10/24) – Missed

Lecture 21 (26/10/24)

Satisfiability problem (SAT)

- We are given a set X of boolean variables $x_1, x_2 \dots x_n$, each can take a value 0 and 1.
- By term over X, we mean $t_i = x_i$ or $t_i = \bar{x}_i$.
- A clause is a disjunction ($\vee$) of distinct term

$$C = t_1 \vee t_2 \vee \dots \vee t_l$$

where $t_i \in \{x_1, x_2 \dots x_n, \bar{x}_1, \bar{x}_2 \dots \bar{x}_n\}$

- A clause C is of length l if it contains l terms ~~and~~

~~target~~ Truth ~~argument~~ assignment is an ~~argument~~ assigment of 0 or 1 to each x_i.

▷ An assignment satisfy a clause C if it causes C to evaluate to 1 under rules of Boolean logic.

Observation

▷ A boolean formula ϕ is a conjunction ($\wedge$) of clauses

$$\boxed{\phi = C_1 \wedge C_2 \wedge \dots \wedge C_K}$$

▷ An assignment satisfies ϕ if it ~~ekr~~ causes all C_i to evaluate to 1.

$\{\phi$ is in POS form$\}$

Ex.

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

$$x_1 = 1,\ x_2 = 1,\ x_3 = 0$$

SAT Problem: Given a boolean formula $\phi : \{C_1 \wedge C_2 \wedge \dots \wedge C_K\}$ over a set of variables $X = \{x_1, x_2, \dots, x_n\}$ does there exist a satisfying assignment?

3-SAT Problem: SAT problem in which every clause is of length exactly 3

Theorem: 3-SAT $\leq_p$ Independent set

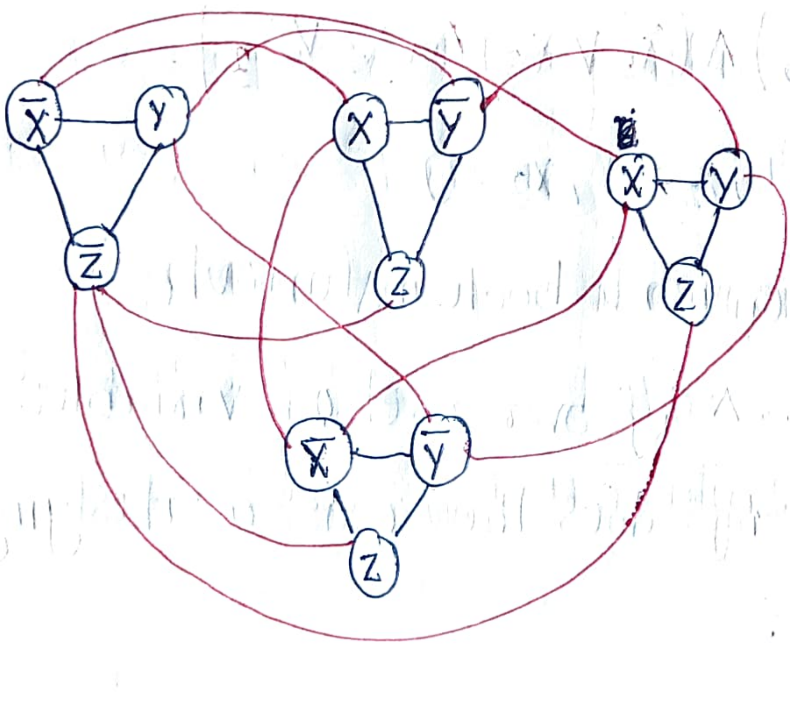
ϕ with k clause each of length 3 $\longrightarrow$ G, K

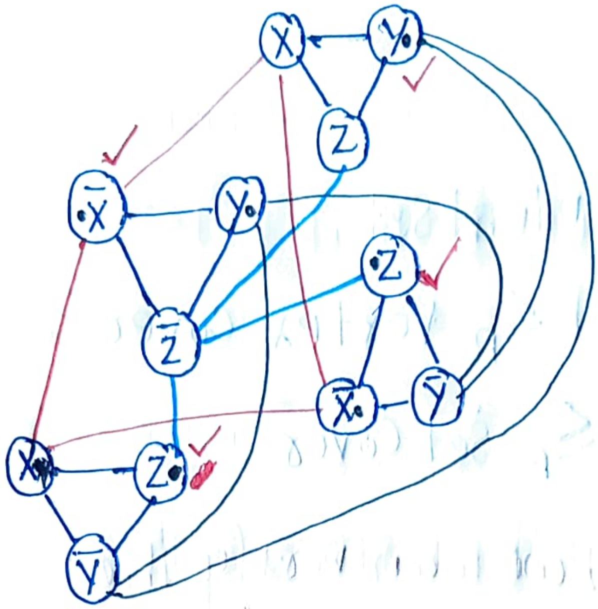
$$\phi = (\bar{x} \vee y \vee \bar{z}) \wedge (x \vee \bar{y} \vee z) \wedge (x \vee y \vee z) \wedge (\bar{x} \vee \bar{y} \vee z)$$

Intuition 1: Select a term from every clauses and set its value to 1.

Intuition 2: No two terms selected conflict

one term is x_i and other term is $\bar{x}_i$.





Claim: G has an independent set of size atleast K if and only if ϕ is satisfiable

Proof: Suppose ϕ is satisfiable.

Consider the satisfying assignment and from each clause select one term that is set to 1 in the assignment.

Let this be set T.

$S = \{v_i \mid t_i \in T\}$, S is an independent set.

Cr. Suppose G has an independent set of size $\boxed{\text{atleast}}$ K

exactly

Theorem: 3-SAT $\leq_p$ Independent Set

Independent set $\leq_p$ Vertex Cover

Vertex cover $\leq_p$ Set Cover

If there is a solution, we can verify the solution (certificate).

Problem X: X is a decision problem and the set problem X consists of input strings for which the answer is YES

Algorithm A: An algorithm A for a decision problem X receives an input string S and returns YES or NO answer.

▷ Let the output of the algorithm A on S be A(S)

Algorithm A solves Problem X:

Algorithm A solves the decision problem X if for all strings S we have.

$$A(S) = \text{YES if and only if } S \in X$$

P = {set of all problems X for which there exists an algorithm A with a polynomial ~~varying~~ running time in the size of the input string that solve X}

Checking algorithm (B): For a checking algorithm B to verify a solution, we need an input string 's' and a 'certificate' 'c' that acts as an evidence that 's' is a YES instance

Algorithm 'B' is an efficient certifier for a problem X if it has following properties:

① B is a polynomial time algorithm taking input s and t

② There is a polynomial function P so that for every string s, we have

$s \in X$ if and only if there exists a string 't' such that $|t| \leq P(|s|)$ and $B(s,t) = YES$

NP: {Set of all problems X for which there exist an efficient ~~condition~~ certifier B}

$\boxed{P \subseteq NP}$

$X \in P \Rightarrow \exists$ A that solves X

B (S, t) { return A(S) }

COOK and LEVN 1971

Q: Does there exists a problem that belongs to NP and not P?

(or)

Does P = NP ?

Q: Among all the problem in NP, what are the $\boxed{\text{hardest}}$ problems?

A: A problem X satisfying the following:

1. $X \in NP$
2. For all $Y \in NP$, $Y \leq_p X$

Such problems are ~~∅~~ called NP complete

[X is called NP-Complete iff X is in NP. (① and ② both)

~~X~~ and X is NP-Hard (satify only ②)]

Theorem:

Suppose X is NP-complete. Then X is in P if and only if P = NP

Proof: suppose P = NP

$X \in NP \rightarrow X \in P$

Suppose $X \in P$

take any $Y \in NP$, we have $Y \leq_p X$, ~~$Y \in P$~~

$\rightarrow Y \in P$

$\Rightarrow NP \subseteq P$ | we know $P \subseteq NP \Rightarrow \boxed{NP = P}$