Quiz on Thursday on lec 17, 18

CSE525 Lec18
Reduction

...

Debajyoti Bera (M21)

SAT: Satisfiability
CNF: Conjunctive normal form

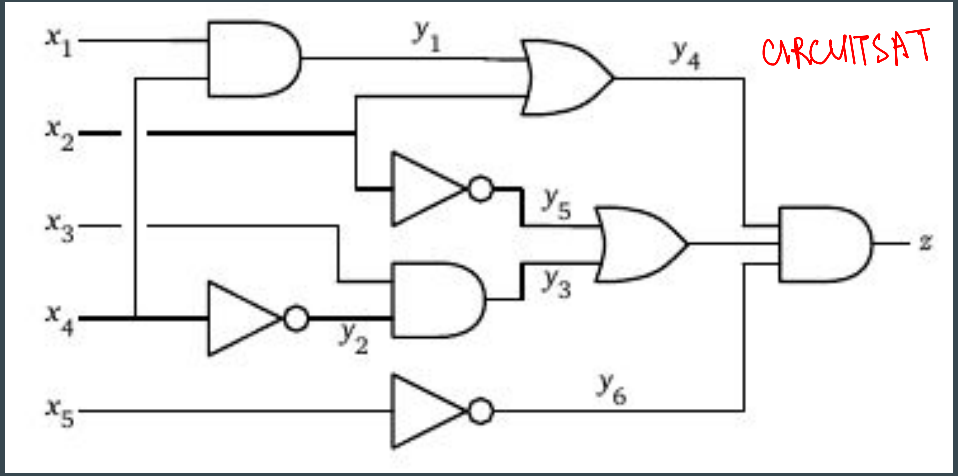
3CNF: CNF with 3 clauses → Every clause has 3 literals

SAT(Boolean expr. E) = Yes|True if E is satisfiable
(a=F, b=F, c=F, d=T is a satisfying assign.)

SAT and CNFSAT and 3CNFSAT

CNFSAT(E: CNF)
↳ $(\ldots \vee \ldots \vee \ldots) \wedge (\ldots \vee \ldots \vee \ldots \vee \ldots \vee) \wedge (\ldots)$
$\underbrace{(\ldots \vee \ldots \vee \ldots)}_{\text{clause}}$ literals
literal: $x, \overline{x}$

$$(a \vee b \vee c \vee \bar{d}) \Leftrightarrow ((b \wedge \bar{c}) \vee \overline{(\bar{a} \Rightarrow d)} \vee (c \neq a \wedge b))$$
F F F T



$(y_1 \vee \overline{x_1} \vee \overline{x_4}) \wedge (\overline{y_1} \vee x_1 \vee z_1) \wedge (\overline{y_1} \vee x_1 \vee \overline{z}$
$\wedge (y_2 \vee x_4 \vee z_3) \wedge (y_2 \vee x_4 \vee \overline{z_3}) \wedge (\overline{y_2} \vee \overline{x_4} \vee z_4$
$\wedge (y_3 \vee \overline{x_3} \vee \overline{y_2}) \wedge (\overline{y_3} \vee x_3 \vee z_5) \wedge (\overline{y_3} \vee x_3 \vee \overline{z_5}$
$\wedge (\overline{y_4} \vee y_1 \vee x_2) \wedge (y_4 \vee \overline{x_2} \vee z_7) \wedge (y_4 \vee \overline{x_2} \vee \overline{z_7}$
$\wedge (y_5 \vee x_2 \vee z_9) \wedge (y_5 \vee x_2 \vee \overline{z_9}) \wedge (\overline{y_5} \vee \overline{x_2} \vee z_1$
$\wedge (y_6 \vee x_5 \vee z_{11}) \wedge (y_6 \vee x_5 \vee \overline{z_{11}}) \wedge (\overline{y_6} \vee \overline{x_5} \vee$

n: # variable in an instance/formula

Assume that no clause contains T/F or literals that can be trivially simplified.

- (x ∨ y ∨ F)
- (x ∨ not(x) ∨ y)
- (x ∨ x ∨ x)

DNF: OR of ANDs

CNF: Pick atleast one literal from every clause
[these literals will be assigned True] s.t. no literal & its compl. are chosen

2CNFSAT to 3CNFSAT

$F' =$ empty
for every clause $(a \vee b)$ in F: // a,b's are literals
$F' = F' \wedge (a \vee b \vee z_i)$ where z_i is a new variable
return F'

$(\cdot \vee \cdot) \wedge (\cdot \vee \cdot) \wedge (\cdot \vee \cdot) \wedge (\cdot \cdot \cdot) \cdots$

① If F is satisfiable, then F' is satisfiable. | Let x_i's denote the vars of F. v_i are truth values

Let $x_1 = v_1, x_2 = v_2, \dots, x_n = v_n$ is a satisfiable assignment of F.

A satisfying assignment of F' is: $x_1 = v_1, \dots x_n = v_n$, & $z_i = F$ for all new z_i's.

Q: Construct poly-time many-one "reduction" from 2CNFSAT to 3CNFSAT.

② If F is not satisfiable, F' is not satisfiable.

For example, $F = (a \vee b) \wedge (a \vee \bar{b}) \wedge (\bar{a} \vee b) \wedge (\bar{a} \vee \bar{b})$

F is not satisfiable (evaluates to F for all assignments)

Reduce(F) $\to F' = (a \vee b \vee z_1) \wedge (a \vee \bar{b} \vee z_2) \wedge (\bar{a} \vee b \vee z_3) \wedge (\bar{a} \vee \bar{b} \vee z_4)$

F' is satisfiable if $z_i = T$ for all z_i.

def Reduce(2CNF formula F):

    ... ??? ...

    Return a 3CNF formula F'

Lemma: F is satisfiable iff F' is satisfiable.

②' If F' is satisfiable then F is satisfiable

⇕

② If F is not sat. then F' is not sat.

Apply Reduce() on `(x ∨ y)`

Apply Reduce() on `(x ∨ y) ∧ (x ∨ NOT(y))`

Apply Reduce() on `(x ∨ y) ∧ (x ∨ NOT(y)) ∧ (NOT(x) ∨ NOT(y)) ∧ (NOT(x) ∨ y)` def Reduce(F): // F has c clauses   Reduction is poly time.

F' = empty

for every clause $(a \vee b)$ in F:

$F' = F' \wedge (a \vee b \vee z_i) \wedge (a \vee b \vee \bar{z}_i)$ where z_i is a new variable

return F'

---

① ⇒ Suppose $x_1 = v_1 \cdots x_n = v_n$ is a satisfying assignment of F.

Construct the following assignment to F':

$x_1 = v_1 \cdots x_n = v_n$, for all z_i, $z_i =$ True

Claim: This assignment satisfies F'.

All $(a \vee b \vee z_i)$ clauses evaluate to true. [$\because z_i =$ True]

All $(a \vee b \vee \bar{z}_i)$ clauses " " " [$\because (a \vee b)$ was already satisfied]

② ⇐ If F' is satisfiable then F is satisfiable.

Let $x_1 = v_1 \cdots x_n = v_n, z_1 = w_1 \cdots z_c = w_c$ be a satisfying assignment of F'.

Construct this assignment: $x_1 = v_1 \cdots x_n = v_n$

$\therefore (a \vee b)$ in F would be True ⟸

Claim: This assignment sat. F.

$(a \vee b \vee z_i)$ & $(a \vee b \vee \bar{z}_i)$.

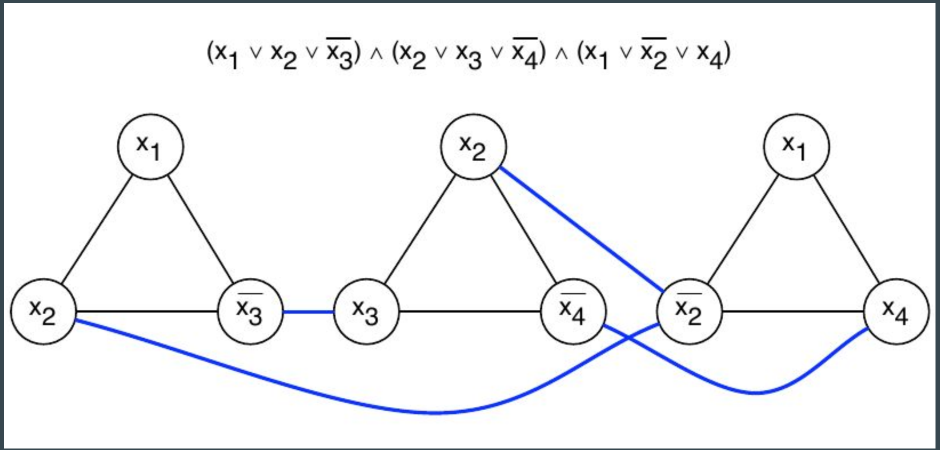
~~both~~ a & b can't be false in these.

IS(G,k) → Yes if there are ≥k vertices which do not have edges between them

IS(G,3) = No

3SAT <= Independent Set (I.S)

IS(G,2) = Yes



Given ANY 3SAT instance F,
reduce F to Independent-Set instance (G,k)

- Reduction takes polynomial time
- If F is satisfiable, then G has an independent-set of size k or more
- If F is not satisfiable, then G has no independent-set size of size k or more

3SAT (alternate view) : is it possible to pick one literal from every clause such that no literal and its negation is selected?

IS : is it possible to pick k vertices such that no vertex and its neighbor is selected?

3SAT ≡ 3CNFSAT

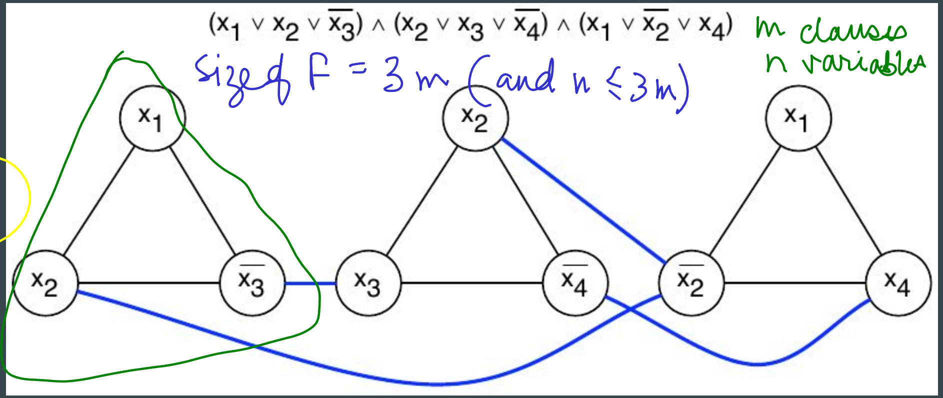
3SAT <= Independent Set

Given ANY 3SAT instance F,
reduce F to Independent-Set instance (G,k)

Complexity:
vertices = 3m
edges $\leq 9m^2$
poly in m, n

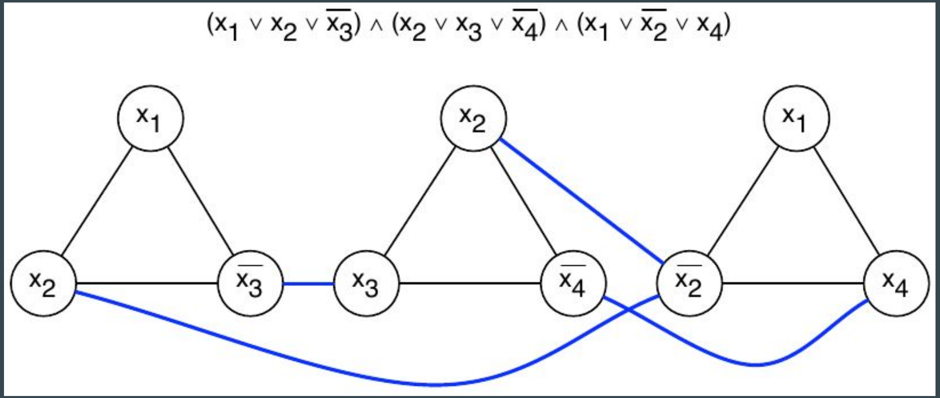
```
def Reduce(F):
    G = empty graph
    For every clause C_i:
        var a,b,c = literals in C_i
        Add to G a triangle over a_i,b_i,c_i
    For every pair of nodes u and v in G:
        If u = x_i and v = y_j:
            If x = not(y):
                Add edge between u and v
    Return G, K = # clauses in F
```

⇒ Suppose F is satisfiable. To show that G has an IS of size m.
F is satisfiable ⇒ F' = set of literals chosen from each clause s.t. .... Construct V' as the corresponding set of vertices. $|V'| = m$.



⇐ Suppose G has an IS V' with m vertices.
Claim: F is satisfiable.

3SAT <= Independent Set

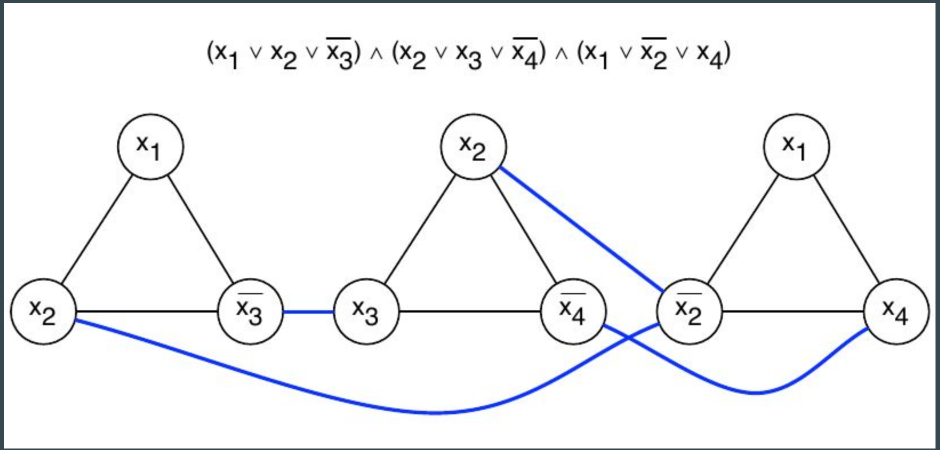


Given ANY 3SAT instance F,
reduce F to Independent-Set instance (G,k)

If F is not satisfiable, then G has no independent-set size of size k or more: If the graph has an independent set S of size k, we know that it has one node from each "clause triangle." Set those terms to true. This is possible because no 2 are negations of each other.

If F is satisfiable, then G has an independent-set of size k or more: If the formula is satisfiable, there is at least one true literal in each clause. Let S be a set of one such true literal from each clause. |S| = k and no two nodes in S are connected by an edge.

3SAT <= Independent Set



Given ANY 3SAT instance F,
reduce F to Independent-Set instance (G,k)

Try 1. Create nodes x1, x1’, x2, x2’, x3, x3’, ... xn, xn’

Try 2: Join x1-x1’, x2-x2’, ...

Try 3: Ensure no more and no less that one literal present in every clause is picked by creating triangles between literals in a clause.

Try 4: Create separate triangles for each clause (otherwise, literals sharing a clause will not be double counted).

3SAT (alternate view) : is it possible to pick one literal from every clause such that no literal and its negation is selected?

IS : is it possible to pick k vertices such that no vertex and its neighbor is selected?

Tut:- UDHS ≤ DHP , 3SAT ≤ 4SAT

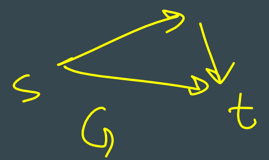
DIRHAMPATH $\leq$ UDIRHAMPATH (undir. G, a, b) $\rightarrow$ if G has a HamPath from a to b

$\underbrace{\text{(digraph } G, s, t)}$ $\rightarrow$ Yes if there is a HamPath in G from s to t

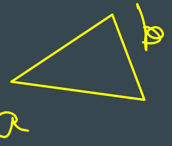
<u>Lemma</u>: G has a HamPath from s to t iff H has a HamPath from a to b

def Reduce(digraph G, s, t): // output (undir H, a, b)

Construct H as a copy of G by removing directions



H:



a = copy of s

b = copy of t.

$\Rightarrow$ Suppose $s \rightarrow v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_{n-2} \rightarrow t$ is a HamPath in G.

Then $s - v_1 - v_2 - \cdots v_{n-2} - t$ " " " " H.

$\Leftarrow$ $G = u_1 \rightarrow u_2 \leftarrow u_3$ $\quad s = u_1, t = u_3$ $\quad$ Reduce(G, s, t) = H: $u_1 - u_2 - u_3$

$a = u_1$

$b = u_3$.