

Reductions

Zeeshan Ahmed



3SAT \in NP

x_1, \dots, x_n

$x_i = \text{True or False}$ [guess in constant time] $\wedge x_i$

Non deterministic machines \rightarrow biased towards True, so even if 1 setting where x_i is true satisfies the overall condition, it will return true.

\rightarrow All the x_i are found in such order & then at the end we can check by substituting them into the equation if the overall expression is satisfied.

Polynomial time solution (in Non deterministic machine)

Guesses \Rightarrow Witness / certificate

NP hard \Rightarrow Problems as hard as any problem in NP

NP complete \Rightarrow NP hard \cap NP

You can reduce all the problems of NP to a problem in NP hard

↳ misleading name

Reduction \Rightarrow You reduce problem A to problem B if you can convert the input of A to equivalent input to problem B.

- If $A \rightarrow B$ then B is as hard as A.

$X \in NP$ complete

$\rightarrow X \in NP$

$\rightarrow X \in NP$ hard \Rightarrow by reducing all the problems in NP to X

[OR you can reduce already known NP complete to X]

A \in NP B \in NP complete $X \Rightarrow$ given problem
If $B \rightarrow X$ then $\forall A \rightarrow X$ since $\forall A \rightarrow B$

Super Mario Brothers reduction

To prove that the game is NP hard, we need to reduce an NP complete problem to Mario. So we choose 3SAT.

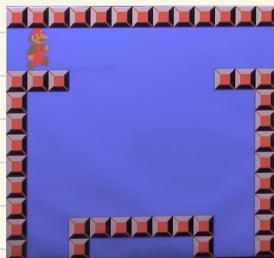
Generalizations:

\rightarrow No time limit

\rightarrow The whole game is in 1 screen

Permissive construction

Variable:



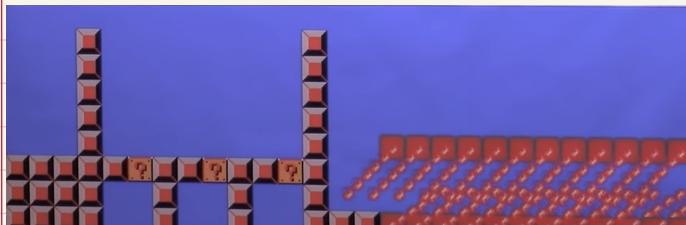
Entry

• n clauses for n variables

True exit leads to the clauses with X in them & False exit leads to \bar{X} clauses

• After visiting all the clauses you get back to the next variable entry.

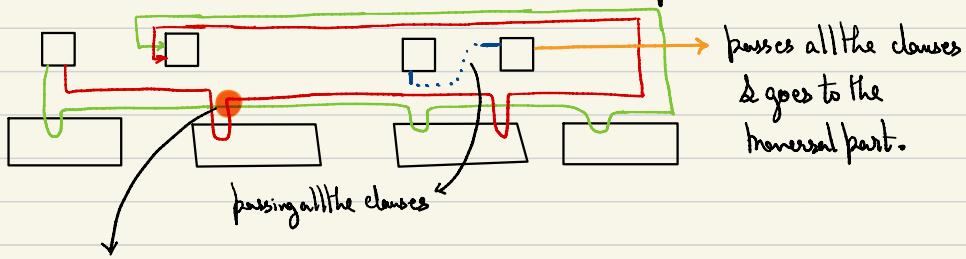
Clauses : • Needs to activate At least one of the stars (OR) & having more than 1 star gives nothing.



$$x_1 \ x_2 \ x_3 \Rightarrow (x_1 \vee x_2 \vee x_3)$$

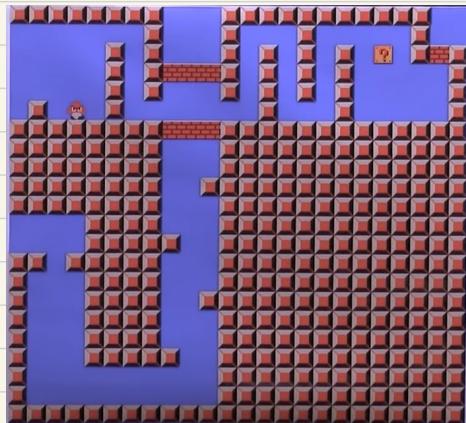
Traversal : After marking all the clauses it goes through all of them from the top.

The last variable Circuit is connected to this traversal part.



To make a realistic level you need to deal with the cuts, for this we use another setup called Cross

GRESS: Does not allow the crossing path to go through in other directions.



Unidirectional paths to prevent 1 path to lead to multiple openings.

Allowed Reductions

$$A \xrightarrow{f(x)} B \xrightarrow{g(x)} C$$

$T(n)$ $T'(m)$

size of $f(x)$ can be at most $O(T(n))$
now

so the complexity of overall conversion will be $T'(T(n))$

3SAT is NP complete \Rightarrow Cook Levin Theorem

Independent set

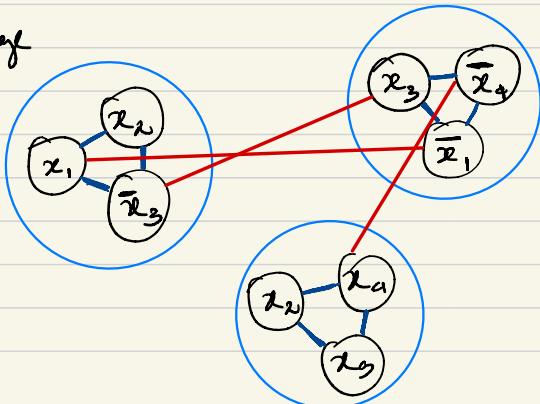
- Given a graph, we pick k vertices such that no 2 vertices have an edge b/w them.
- To prove that is NP hard we reduce 3SAT to k clique.

Given an expression: $(x_1 \vee x_2 \vee \bar{x}_3) \wedge (x_3 \vee \bar{x}_2 \vee \bar{x}_1) \wedge (x_2 \vee x_4 \vee \bar{x}_3)$

Each variable: A vertex

Each clause: A clique

Joining x & \bar{x} by an edge



- We just have to pick one from each clique, if there are repetitions then it is fine.
- Picking 1 x , does not mean the other x , is already picked. But if any 1 of them is picked it is sufficient for it to become 1.

NPcomplete proof

- We need a witness & a verification algorithm to check if the graph has a k sized independent set.

Witness: set of vertices $[k]$ which form the independent set.

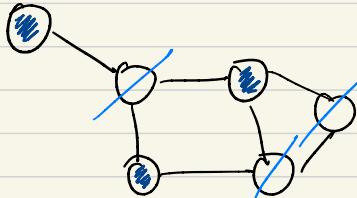
Verification: • Pick a vertex from the set & remove all its neighbours from the vertex.

• Repeat until the set is exhausted, if a vertex which is present in the set gets eliminated then its invalid.

G has k sized independent set $\Leftrightarrow \bar{G}$ has $n-k$ sized clique

Vertex Cover

- Set of vertices which contain at least 1 end of each edge.
- Finding the minimum vertex cover is the NP-hard problem.



Given graph, does it have vertex cover of size k .

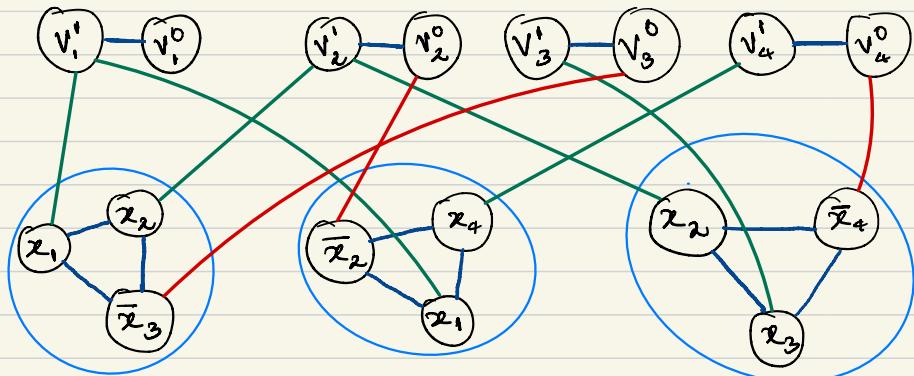
- Independent set \rightarrow Vertex cover of k has $n-k$ sized independent set.

From 3SAT

v_i^0 & v_i^1 are 2 vertices representing variable x_i , they have an edge between them.

A 3-clique represents a clause.

$$(x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_2 \vee x_4 \vee x_1) \wedge (x_2 \vee \bar{x}_4 \vee x_3)$$



- Joining x_i with v_i^1
- Joining \bar{x}_i with v_i^0

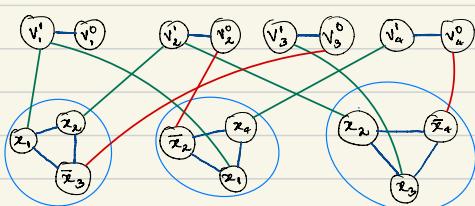
Conditions to form a vertex cover

- v_i^0 or v_i^1 has to be picked since there is an edge between them.
- If you pick v_i^0 then \bar{x}_i has to be picked since there is an edge b/w v_i^0 & \bar{x}_i which will be uncovered.
- If $x_i = 1$ pick v_i^1 & other vertices in the clause which has x_i .

Number of vertices picked = 2 · clauses + 1 · variables
 (x_i) (v_i)

Solving 3SAT will solve min vertex cover of $k = 2m+n$ for that graph.

For the same graph, if we have a vertex cover of size $2m+n$ then we solve 3SAT.



Edge b/w v_i^1 & v_i^0 is present, 1 of them has to be picked. Eliminates n vertices

Among the 3 vertices in a clause clique we have to pick at least 2. Eliminates the rest.

Cook Levin Theorem

Checking the satisfiability of a boolean expression is NP complete.

SAT is NP

- Given an assignment we can verify if the assignment satisfies b or not.

SAT is NP hard

- We need to reduce all NP problems to 3SAT to prove it.

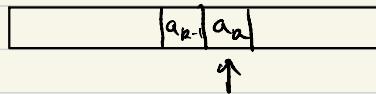
- Any SAT can be reduced to 3SAT with boolean algebra.

Let A be some NP problem. A has a polynomial time verifier

$$\exists c \quad V(x, c) = 1 \quad x \in A \quad \forall c \quad V(x, c) = 0 \quad x \notin A$$

↓
polynomial in $|x|$

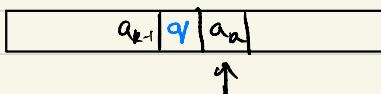
Every TM has a configuration, it is the union of states & the tape



state $\rightarrow q$

$$C = T \cup Q$$

This can be written as



→ This contains the entire configuration of the given TM.

$m = T(n) \rightarrow$ running time [number of steps]

• After each step there can be a different configuration, all of them belong to C

So complete configuration $\in C^m$

This can also be proven that $T(n) \geq S(n)$ is a loop since we come back to previous state & loop.

Tableau \rightarrow Sequence of configuration

#	q_0	w_1	w_2	w_n	#
#	w'_1	q_1	w_2	w_n	#
#	#	
#	#	
#	#	

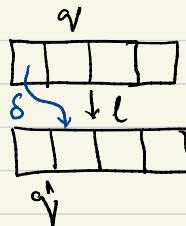
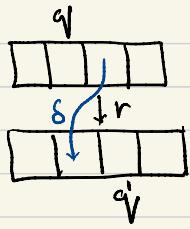
—
No loop $\Rightarrow S(n) \leq T(n)$
—
 $T(n)$
—

—
 $S(n)$

Validity of a Tableau \rightarrow Sequence should be obtained from a valid transition table (S)

$q_1^1, w_0, w_1^1, w_2 \xleftarrow{l} w_0, q_1, w_1, w_2 \xrightarrow{r} q_1^1, w_1^1, w_2$

by default the block after the state is the header



In general, it can be seen that the block depends upon the 3 blocks above it in the previous configuration.

Variables in SAT

There are $|C|$ variables for each position on the tableau

$$x_{i,j,s} = T \quad \text{if} \quad \text{Tableau } i,j = s$$

Encoding all the necessary conditions for the tableau based on these variables.

Conditions:

- Starts at q_{start} at the beginning of the tape
- Each transition corresponds to the transition table
- Reaches q_{accept} somewhere.

$A \rightarrow \text{SAT}$

Converting conditions to formula

$\forall i, j \quad x_{i,j,s} = T \quad \exists s \in C \quad [\text{only 1 should satisfy}]$

$$\phi_{i,j} = \bigvee_{s \in C} \left(x_{i,j,s} \bigwedge_{\substack{s' \in C \\ s' \neq s}} \neg x_{i,j,s'} \right) \quad \text{size} \rightarrow O(|C|^2)$$

That was for a single cell. we have it for all the cells

$$\bigwedge_{i,j \in T} \phi_{i,j} \quad \rightarrow \text{condition for variable to be valid}$$

\rightarrow Initial configuration

$\forall s_{\text{start}} \rightarrow \text{initial state} \quad \omega \rightarrow \langle w_0, w_1, \dots, w_n \rangle$ [input string]

$\boxed{s_{\text{start}} \quad w_0 \quad \dots \quad w_n} \rightarrow \text{The required start}$

$$\phi_{\text{start}} = x_{0,0,w_0} \wedge x_{0,1,w_1} \wedge \dots \wedge x_{0,n+1,w_n}$$

\rightarrow Valid transitions

- We check 3×2 windows \rightarrow There are $(S(n)-2)(T(n)-1)$ 3×2 windows

Total possibilities $\rightarrow |C|^6$

Valid windows based on the S function $\subseteq C^6$

We check if every 2×3 window is valid.

General window \rightarrow $x_{i,j} \ x_{i+1,j} \ x_{i+2,j} \rightarrow B_{ij}$
 $x_{i,j+1} \ x_{i+1,j+1} \ x_{i+2,j+1}$

Now suppose we have some valid window $W_i \rightarrow \begin{matrix} a & b & c \\ d & e & f \end{matrix}$

Suppose there are K valid windows

To check if B_{ij} matches W_i we do

$B_{ij} \rightarrow x_{i,j,a} \wedge x_{i+1,j,b} \dots \wedge x_{i+2,j+1,f}$

Validity of $B_{ij} \rightarrow \bigvee_{k=0}^K B_{ijk}$

Now we have to check validity of all the windows in the tableau

$\bigwedge_{i,j} (B_{ij})$

\rightarrow Accepting configuration

$\Phi_{\text{accept}} \rightarrow \bigvee_{i,j} x_{i,j,\text{vac}}$ At least 1 state has to be vac.

\rightarrow Combining the constraints

$\underline{\Phi_{A,w} = \Phi_{\text{cell}} \wedge \Phi_{\text{start}} \wedge \Phi_{\text{windows}} \wedge \Phi_{\text{accept}}}$

→ Polynomial time reduction

Checking at all i, j , so the complexity will be $O(T(n)S(n))$

→ SAT to CNF3SAT

$$(l_1 \vee l_2 \vee l_3 \dots l_n) \rightarrow (l_1 \vee l_2 \vee z) \wedge (\bar{z} \vee l_3 \vee \dots l_n)$$

→ perform it recursively

This is polynomial time, steps taken $\rightarrow \frac{n}{2}$