P. vs. NP vs NP Completeness vs. NP hard

P. vs. not P

- *P* is good and *exp* is bad – no brainer, done
- Need to look a little deeper

Not P

- Many problems
  - *exp* steps to *find* solutions
  - A lot less than *exp* steps to *verify* solutions
  - Puzzles (Sudoku, crossword puzzles)
  - Hamiltonian cycle
    - A tour through all vertices without repetition
  - Max Cliques
    - Largest fully connected sub graph
  - Vertex cover problem
    - using 3 colors
  - Subset sum
    - Any subset sums to zero?

NP

- Non-deterministic computation (lucky guess) to *find* solutions
  - Oracle will always return true if at all possible
  - Certificates can be found in *P* time
- Deterministic computation (*P* time) to *verify* solutions
- More formally a decision problem, i.e., answer is yes or no

Why “non-deterministic”

- Why creates such an unrealistic computational model?
- Intuition:
  - There are very hard problems (to figure out solutions)
  - Need a very powerful computer (oracle or “non-deterministic”)
  - Without it, no polynomial solutions
  - But we know such a powerful machine do not exist, so such problems are “probably” intrinsically hard

Caveat

- The same problem can be in *P* or *NP* depends on parameters
  - K-Cliques $O(k^2n^k)$ vs. max cliques
    - $O(n^k)$ subgraphs of size k
    - $O(k^2)$ time to check if one is fully connected
  - 2Sat (satisfiability) vs. 3Sat
  - 2-color vs. 3-color vertex covering

not P

- Certainly, there are even harder problems
  - Is there a best way to play chess (go)?
    - Hard to find solution
    - Hard to verify solution
  - Turing machine halting problem
    - No solution at all

Reduction

- A can be reduced to B (A<=B) iff
  - There is a mapping from input A to input B
  - There is a mapping from output B to output A
  - If you can solve B, you can solve A
  - B is at least as hard or harder than A
  - The escape route is simpler than the solution itself (otherwise, don't bother)
  - For decision problems
    - If B is yes, A is yes
    - If B is no, A is no

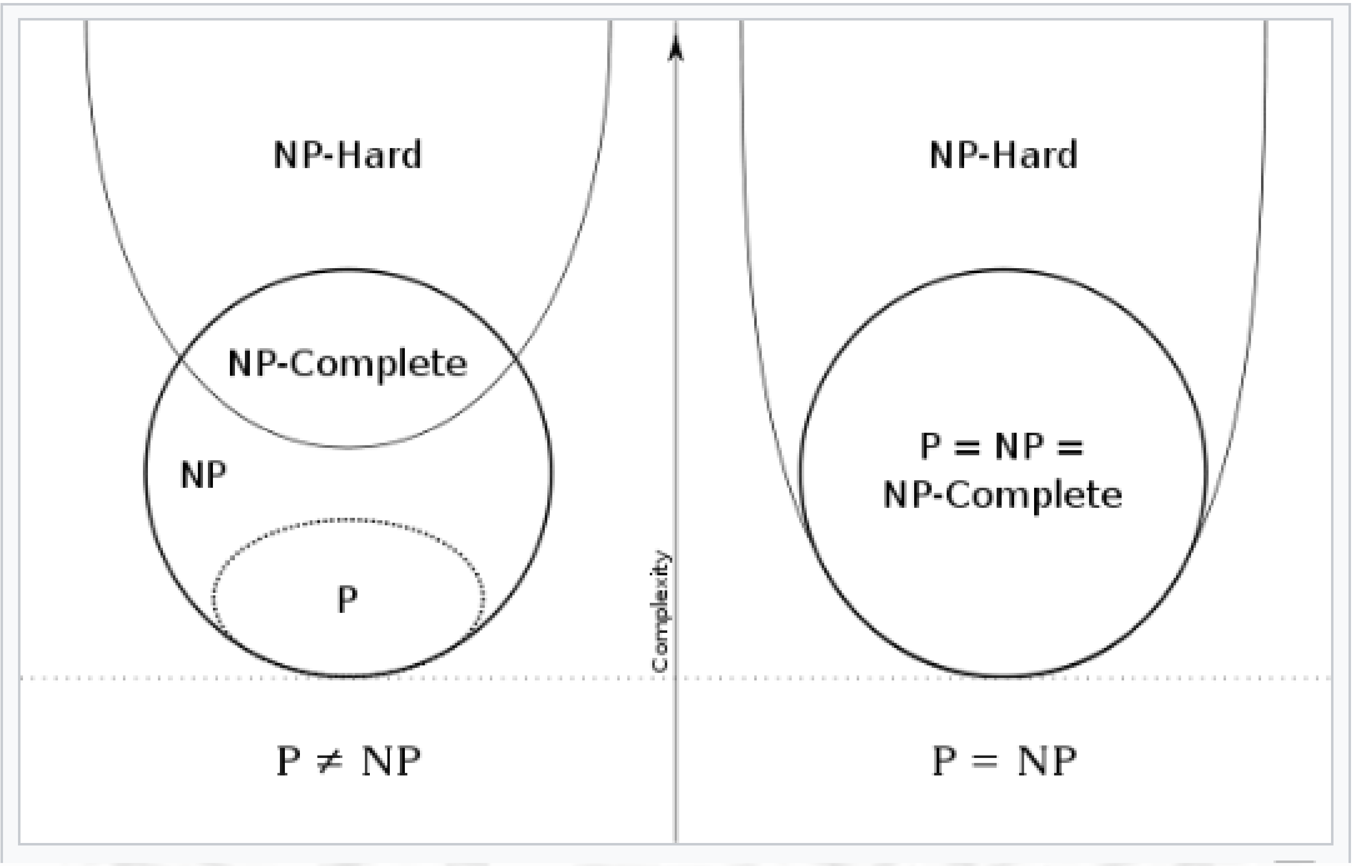
Simple Example of Reduction

- Sorting < convex hull
- Convex hull is as hard as or harder than sorting
- Sort 8, 3, 15, 7
  - Generate 2D point (8, 64), (3, 9), (15, 225), (7, 49)
  - Find convex hull of these points (a parabolic curve)
  - Read out CH vertices along x
  - Transform step is O(n), less than the solver complexity of O(nlogn)

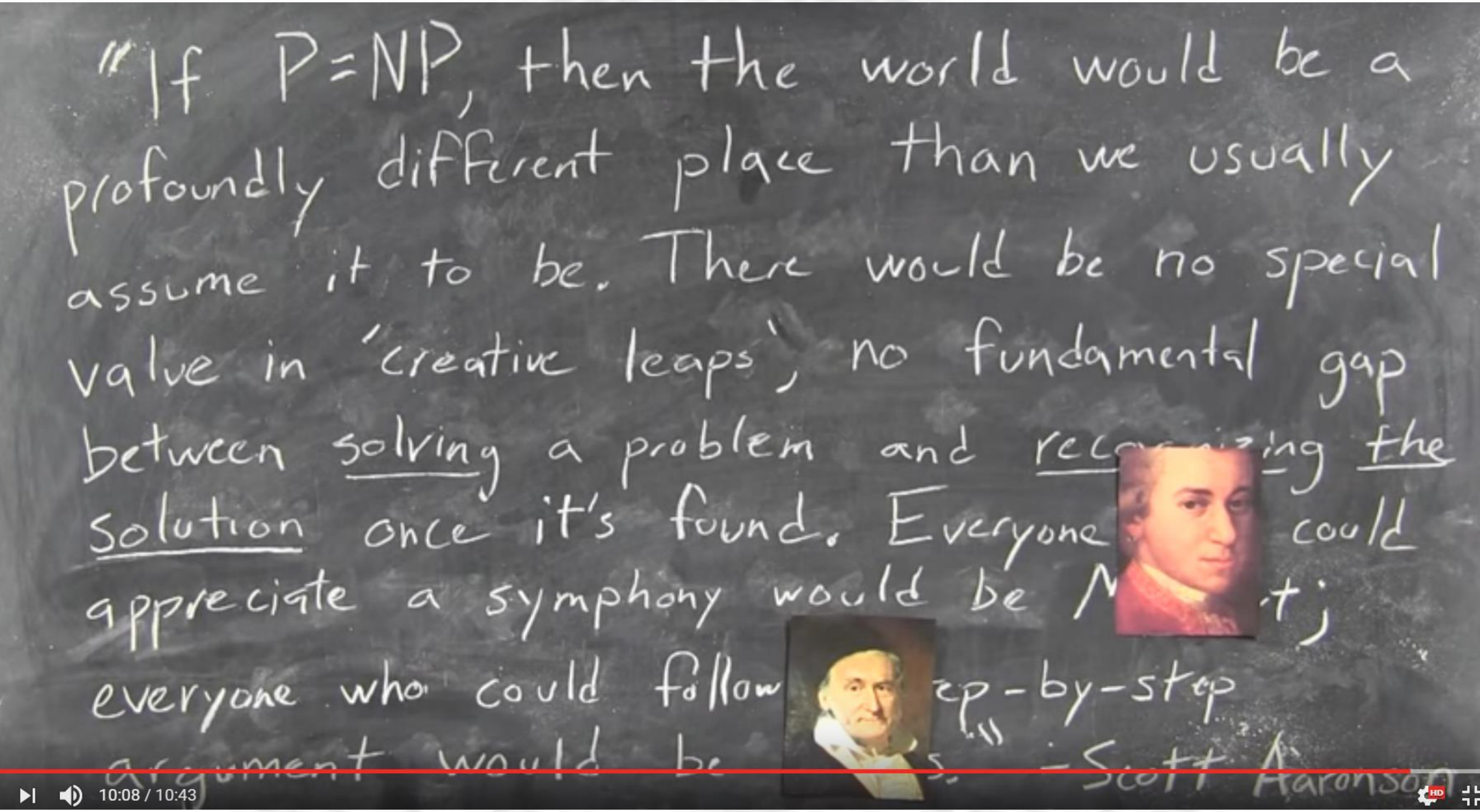
Complexity Classes

- P: solvable in polynomial time
- NP:
  - Decision problem
  - *Solvable* in nondeterministic time
  - *Verifiable* in polynomial time
- NP hard: all NP problem can be reduced to (NP < NP hard)
  - Does not have to be a decision problem
  - Does not even need to have a solution
- NP complete
  - Every NP problem can be reduced to (NP < NP complete)
  - NP complete = (NP ^ NP hard)

P == NP ?


NP-Hard
NP-Complete
NP
P
Complexity
P ≠ NP
NP-Hard
P = NP =
NP-Complete
P = NP

P==NP?



https://www.youtube.com/watch?v=YX40hbAHx3s

Examples: Hamilton Cycle

- Start from a vertex, visit every vertex in the graph (without repetition and back)

- N! permutation of vertices
- Check O(n)

- 1 permutation of vertices (lucky guess)
- Check O(n)

Examples: 3D matching

- N x, N y, Nz (3 genders instead of 2)
- Total N^3 of triplet (x_i, y_i, z_i)
- T (size n) subset of (x.y.z) such that
  - (x1,y1,z1) and (x2, y2, z2) in T, then x1!=x2, y1!=y2, z1!=z2

- C(N^3, N) <- exponential time
- Check O(N^4)

- N triples (N lucky guesses)
- Check O(N^4)

Examples: Sudoku

- ?? solutions
- Check $O(n^2)$

- 1 final configuration (at most $O(n^2)$ lucky guess)
- Check $O(n^2)$

		4					9	
7				6				5
	9		5	4	1	8	7	2
			1	8	7		4	
2	4	3	6		5	1	8	7
	8		3	2	4			
9	2	1	8	7	6		5	
6				1				8
	3					7		

Examples: SAT (Satisfiability)

- Any number of conjunction of disjunction of Boolean variables $(x)\wedge(x + \bar{y})\wedge(z + y + \bar{w})$

- 2^n guesses
- Check: go through each clause and verify true results

- N lucky guesses
- Check: go through each clause and verify true results
- A first problem proven NP complete (Cook)

Use of Reduction (A<B)

- A reduces to B and A is hard (NP-complete)
  - B must be hard, if not
    - every NP-complete problem can be solved
    - Every NP problem can be solved
    - P==NP
- A reduces to B and B is easy
  - A must be easy (assume the “escape” route is not expensive to build)
  - Sorting < convex hull
    - Reduction is O(n)
    - Conex hull is O(nlogn)
    - Sorting is O(nlogn)

Mistake in use of Reduction

- A reduces to B and B is hard (NP-complete)
- 2Sat < 3Sat
  - 2Sat
    - Any number of variables
    - Each clause has exactly two variables
    - Polynomial algorithm
  - 3Sat
    - Any number of variables
    - Each clause has exactly 3 variables
    - NP-complete
  - Only show that 2Sat is easier than 3Sat

Proof of NP Complete

- Proof of NP
  - Problem in NP
    - Certificate in *P* time using lucky guess
    - Verification in *P* time
  - Problem in NP complete
    - Find an NP complete problem (e.g., SAT, Cook-Levin Theorem)
    - NP complete problem < current problem
    - The reduction should run in polynomial time

3Sat

- 3Sat
  - A special form of Sat
  - Conjunction norm form (CNF)
  - Conjunction of clauses
  - Each clauses are made of 3 literals
  - N literals
  - M clauses
  - N lucking guesses and M verification O(nm)
  - Even though it is “special” case of Sat, it is not any easier (all NP-complete problems are equivalent)

Sat reduces to 3Sat

- Clauses of 1 $(x) \leftrightarrow (x+a+b)(x+\bar{a}+b)(x+a+\bar{b})(x+\bar{a}+\bar{b})$
- Clauses of 2 $(x+\bar{y}) \leftrightarrow (x+\bar{y}+c)\,(x+\bar{y}+\bar{c})$
- Clause3 of 3
  - Do nothing
- Make sure that the old clause and the new clause has exactly the same truth table
  - x true, the new clause is true
  - x false, the new clause is false

Sat reduces to 3Sat

- Clauses of 4 or higher
  - Create "link" variables
- *P* time reduction O(mn)
- Sat is true, 3Sat is true
- 3Sat is true, Sat is true

$$(x + \bar{y} + w + u) \leftrightarrow (x + \bar{y} + l)(\bar{l} + w + u)$$

$$(x + \bar{y} + z + w + u + v) \leftrightarrow (x + \bar{y} + l_1)(\bar{l}_1 + z + l_2)(\bar{l}_2 + w + l_3)(\bar{l}_3 + u + v)$$

3Sat to Sat

- Reduction is to do nothing, as 3 Sat is Sat
- An example of all NP problems are equivalent

3 Colorability < Sat

- Graph -> Boolean formula
- Can be 3-colored -> Boolean formula true
- Cannot be 3-colored -> Boolean formula false

3 Colorability < Sat

- In NP
  - Lucky guess (one each for n vertices, O(v))
  - Verify take O(e)
  - Total time is *P*
  - One and only one color for a node
    - $a1\overline{a2a3} + a2\overline{a1a3} + a3\overline{a1a2}$
  - No adjacent nodes are of the same color

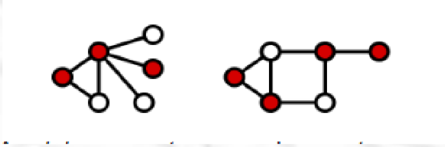
$$a1 \rightarrow \overline{b1}, a2 \rightarrow \overline{b2}, a3 \rightarrow \overline{b3}$$
$$b1 \rightarrow \overline{a1}, b2 \rightarrow \overline{a2}, b3 \rightarrow \overline{a3}$$

  - Reduction is polynomial time O(3v+6e)

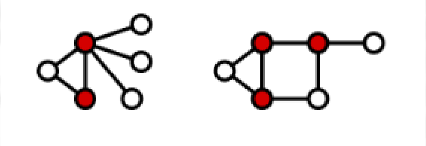
- Every logic formula can be put into CNF

Minimum Vertex Cover

- Vertex cover: a subsect of vertices in a graph so that every edge is incident on at least one vertex in the set



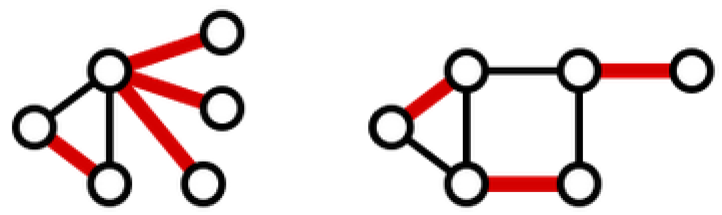
- Min vertex cover: a vertex cover with the smallest # of vertices



- Visualization
  - Edges: streets
  - Vertices: intersection of streets
  - Put a convenient store at enough intersections that everyone on every street can get to at least one directly

Sidebar: Min Edge Cover

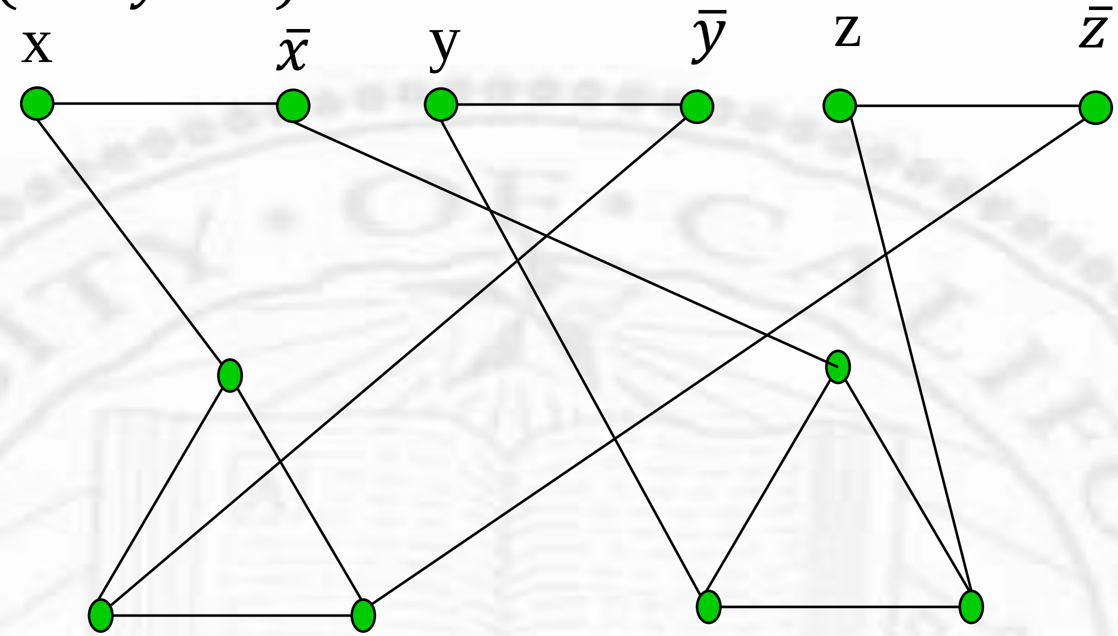
- Edge cover: a subsect of edges in a graph so that every vertex is an end point of one of the edge in the set
- Min edge cover: an edge cover with the smallest # of edges
- Marriage problems (max matching that is also a min edge cover) with polynomial solution $O(n^3)$



3-Sat < Vertex Cover

- Boolean formula is true <-> Vertex cover of size n (literals) + 2m (clauses)
- Construction
  - Literals (variables) <-> an edge (x<->$\bar{x}$)
  - Clause <-> triangle
  - Additional edge <-> literals in clauses

$(x+\bar{y}+\bar{z})(\bar{x}+y+z)$



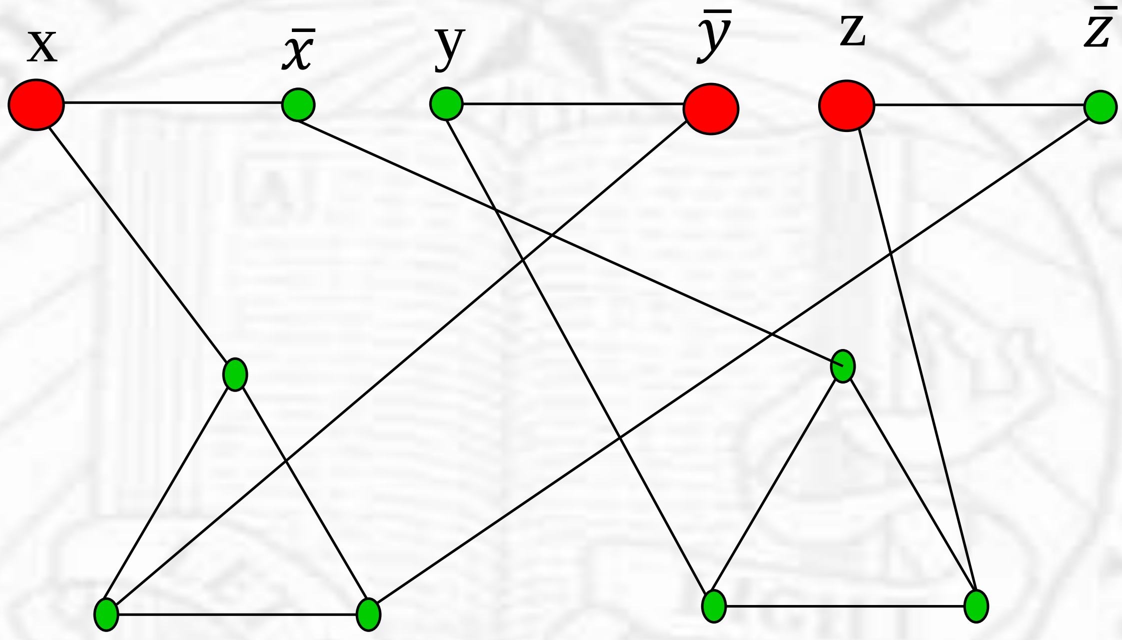
- 3 literals <-> 3 edges (top)
- 2 clauses <-> 2 triangles (bottom)
- Edges (top to bottom) <-> literals to clauses
- Vertex cover
  - One vertex/top edge (n)
  - Two vertices/bottom triangle (2m)
  - One vertex/top to bottom connection (?)

$$(x + \bar{y} + \bar{z})(\bar{x} + y + z)$$

$$x = T$$

$$y = F$$

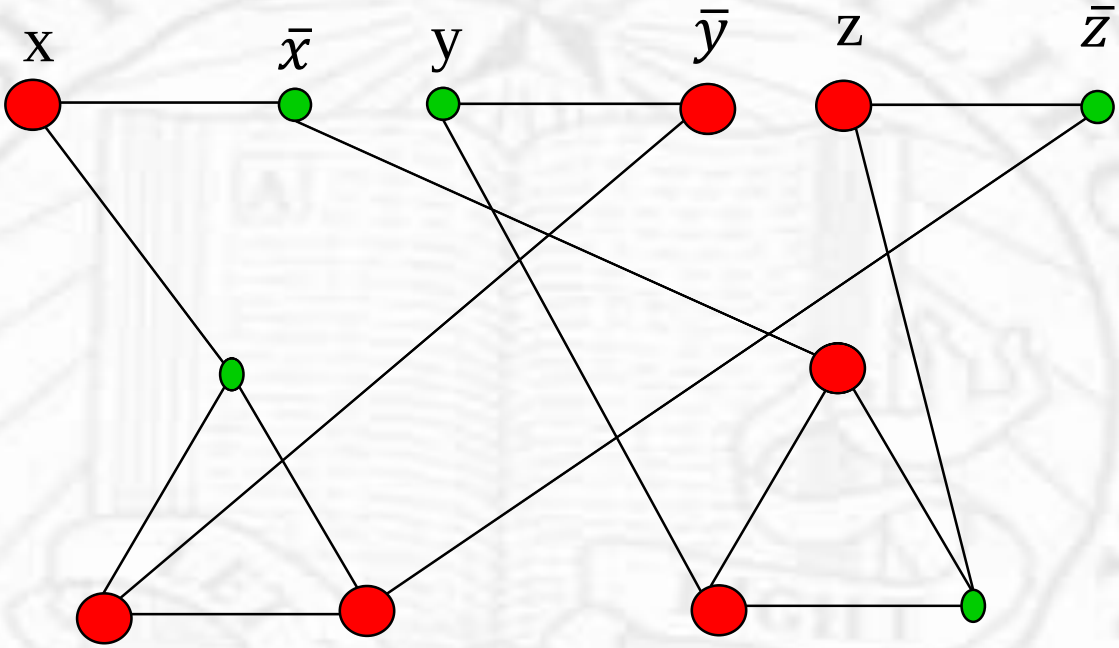
$$z = T$$



- Formula can be true <->
- Vertex cover of n+2m exists

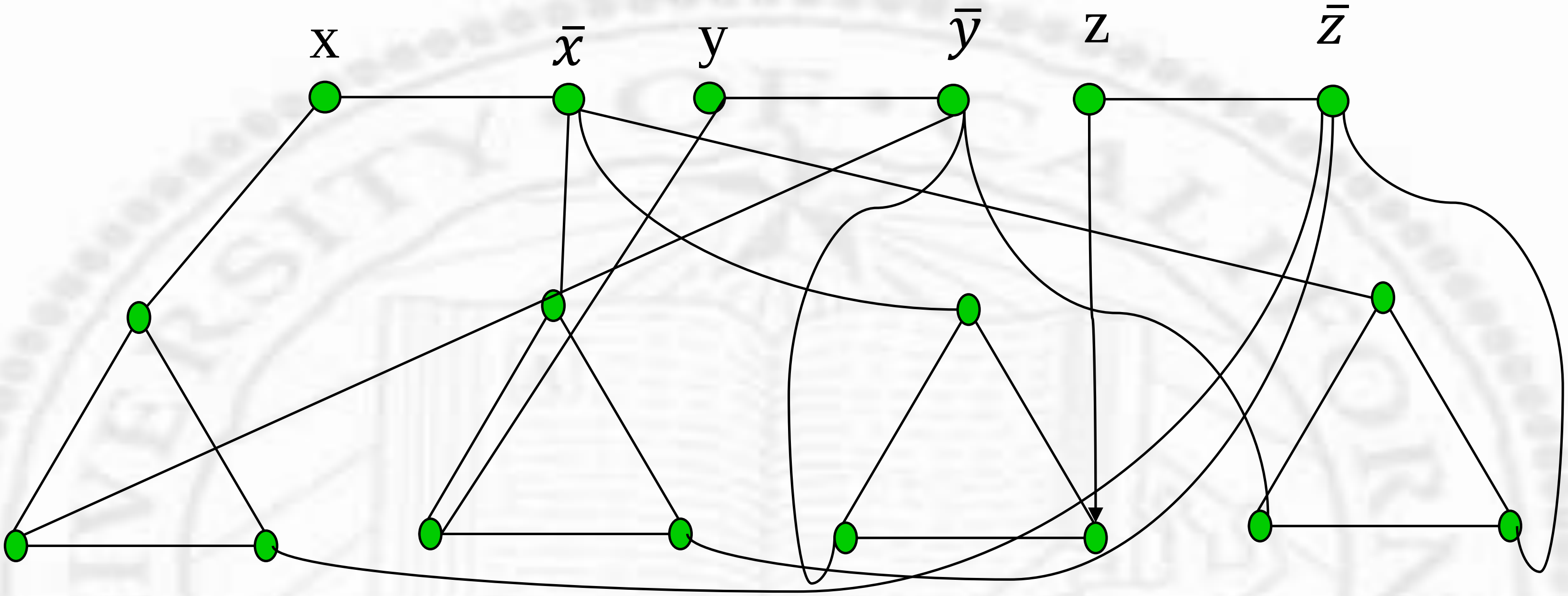
$$(x + \bar{y} + \bar{z})(\bar{x} + y + z)$$

$$x = T$$
$$y = F$$
$$z = T$$



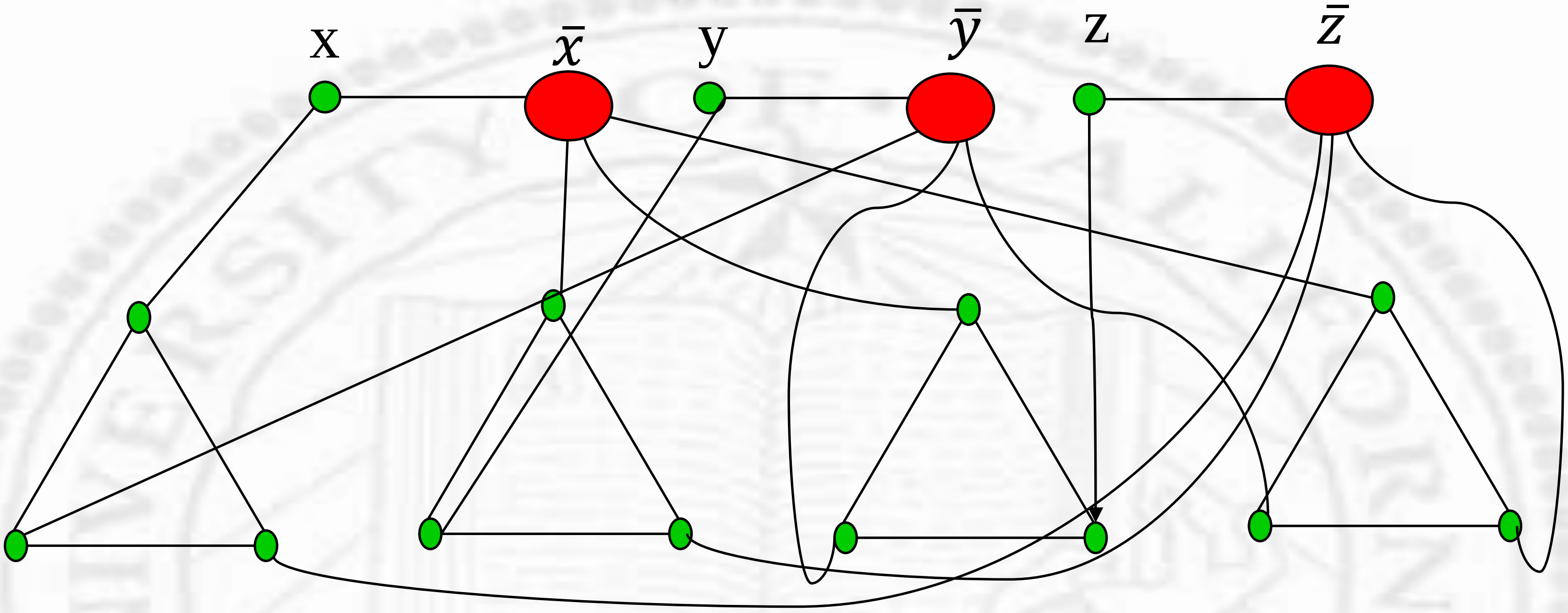
- Formula can be true <->
- Vertex cover of n+2m exists

$(x + \bar{y} + \bar{z})(\bar{x} + y + \bar{z})(\bar{x} + \bar{y} + z)(\bar{x} + \bar{y} + \bar{z})$



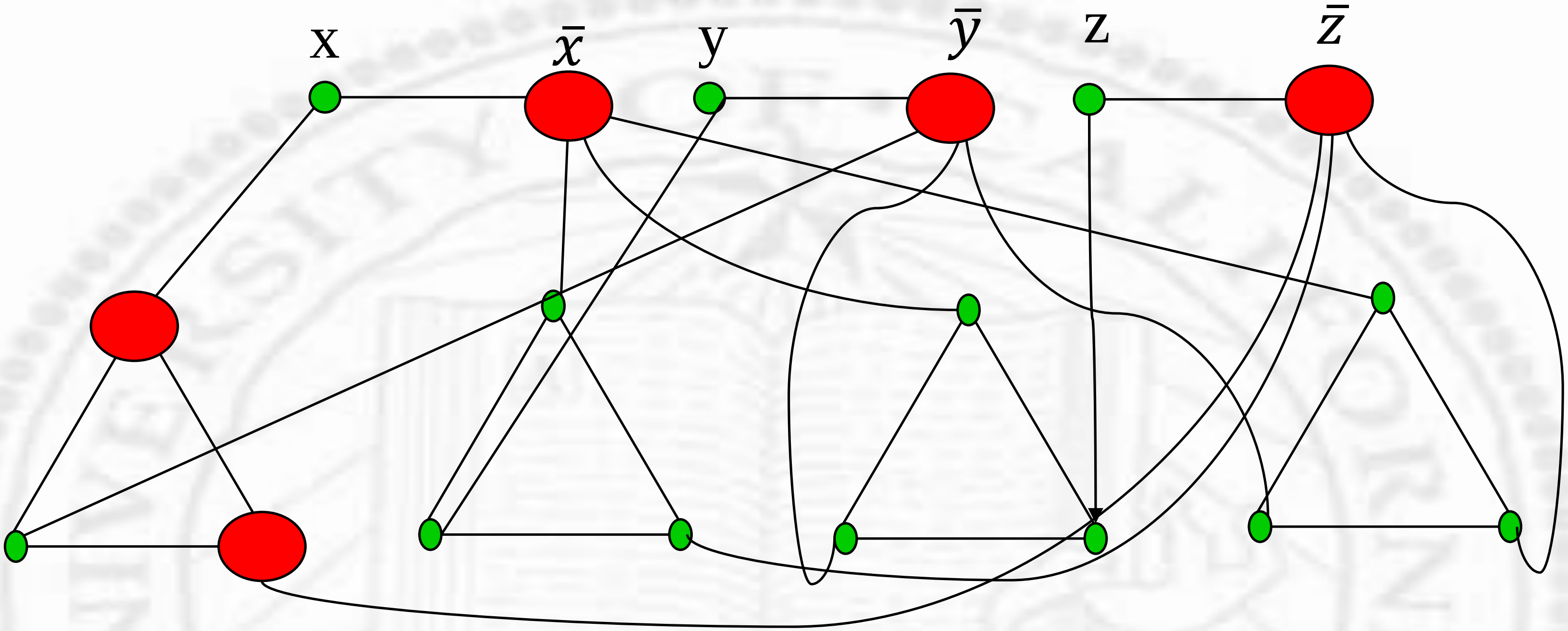
- 3 literals <-> 3 edges (top)
- 4 clauses <-> 4 triangles (bottom)
- Edges (top to bottom) <-> literals to clauses
- Find a vertex cover n+2m = 3+2*4 = 11-> formula can be true

$$(x+\bar{y}+\bar{z})(\bar{x}+y+\bar{z})(\bar{x}+\bar{y}+z)(\bar{x}+\bar{y}+\bar{z})$$



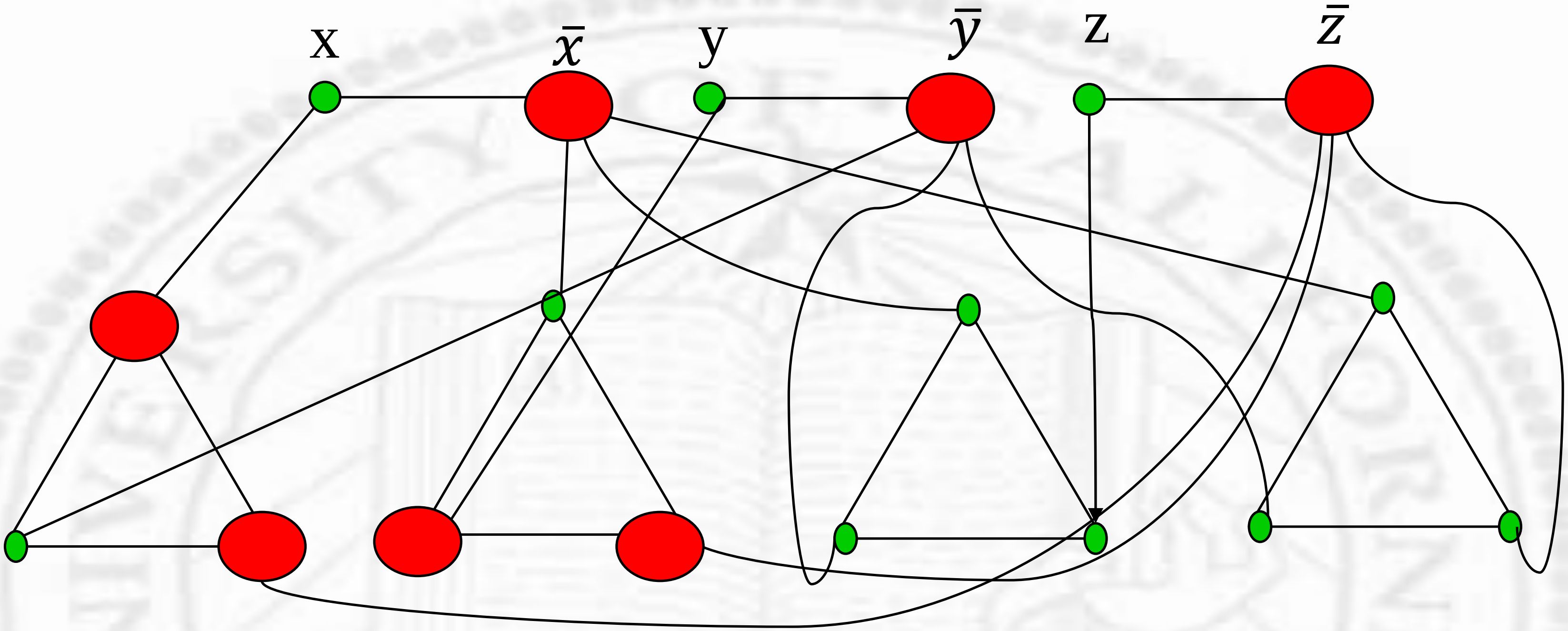
- Choose x, y, z negated on top

$$(x+\bar{y}+\bar{z})(\bar{x}+y+\bar{z})(\bar{x}+\bar{y}+z)(\bar{x}+\bar{y}+\bar{z})$$



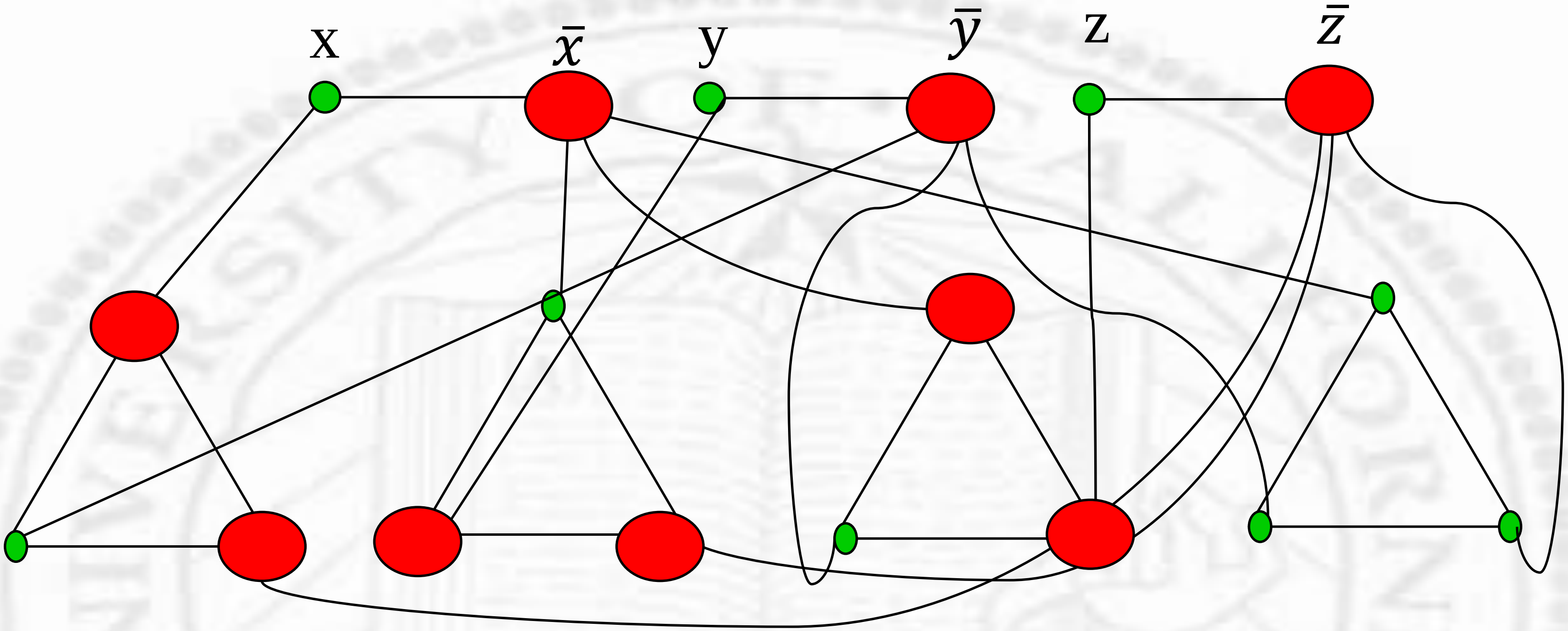
- First clause

$$(x+\bar{y}+\bar{z})(\bar{x}+y+\bar{z})(\bar{x}+\bar{y}+z)(\bar{x}+\bar{y}+\bar{z})$$



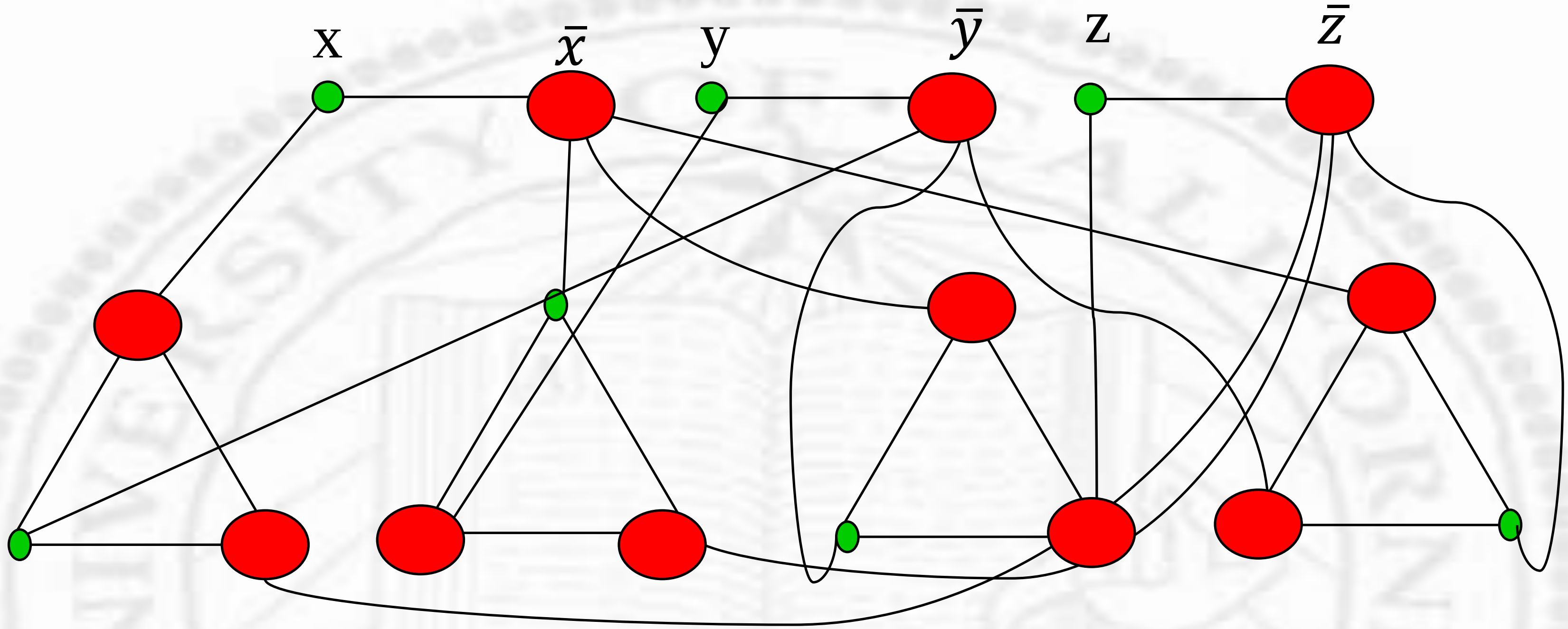
- 2^{nd} clause

$$(x+\bar{y}+\bar{z})(\bar{x}+y+\bar{z})(\bar{x}+\bar{y}+z)(\bar{x}+\bar{y}+\bar{z})$$



- 3rd clause

$$(x+\bar{y}+\bar{z})(\bar{x}+y+\bar{z})(\bar{x}+\bar{y}+z)(\bar{x}+\bar{y}+\bar{z})$$



- 4th clause
- x=y=z=false
- Vertex cover -> variable assignment to make clause true

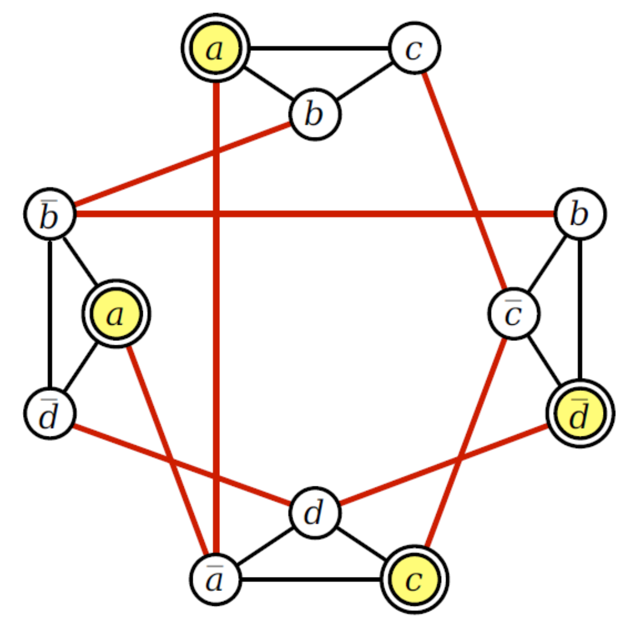
Max Independent Set

- Independent set in a graph G is a subset of vertices with no edges between them
- NP
  - Lucky guess O(n)
  - Verification O(n^2)
- NP completeness
  - 3 Sat -> graph

3Sat < Max Independent

- Nodes: one node for each instance of each literal
- Edges:
  - Correspond to literal in the same clause
  - Correspond to a literal and its inverse

$$(a \vee b \vee c) \wedge (b \vee \bar{c} \vee \bar{d}) \wedge (\bar{a} \vee c \vee d) \wedge (a \vee \bar{b} \vee \bar{d})$$

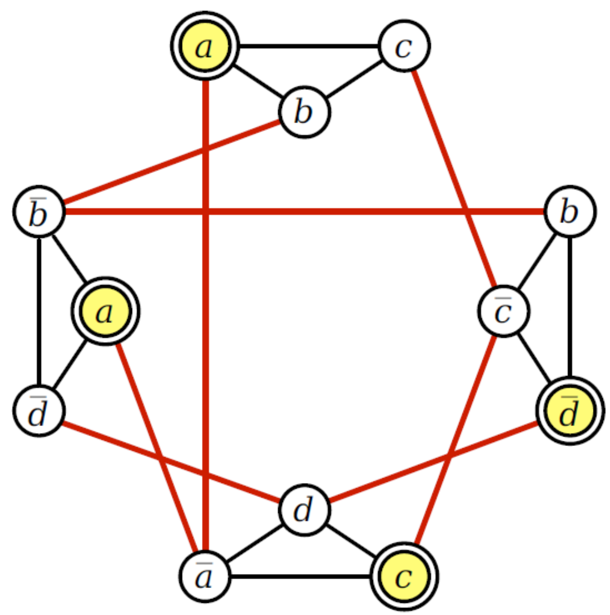


3Sat < Max Independent

- For K clauses
  - Formula is satisfiable iff graph has an independent set of k

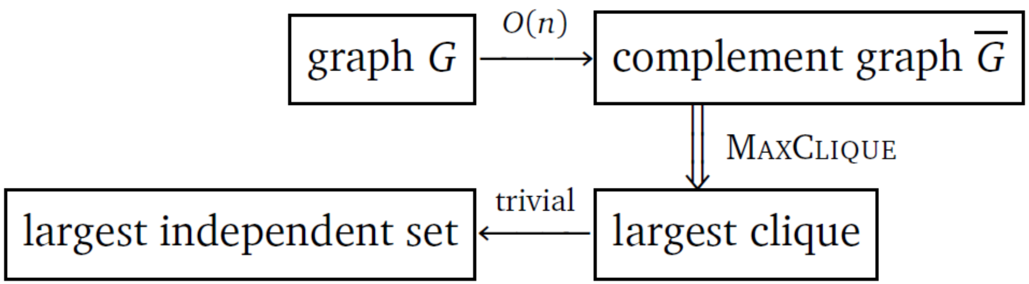
$$(a \vee b \vee c) \wedge (b \vee \bar{c} \vee \bar{d}) \wedge (\bar{a} \vee c \vee d) \wedge (a \vee \bar{b} \vee \bar{d})$$

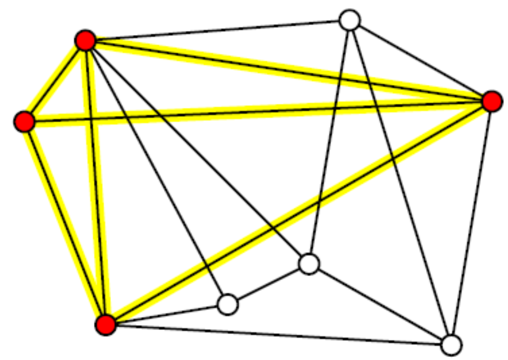
- Proof?
  - All clauses must be true
  - At least one literal in each clause must be true
  - The inverse of that literal in all other clauses must be false



Max Clique Size

- Clique: fully connected subgraph
- Max independent > max clique
- G-> G' (same vertex, opposite set of edges)
- Independent in G iff clique in G'





A graph with maximum clique size 4.