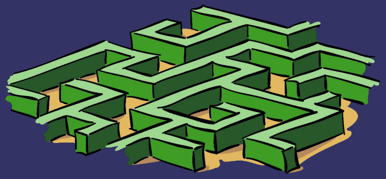
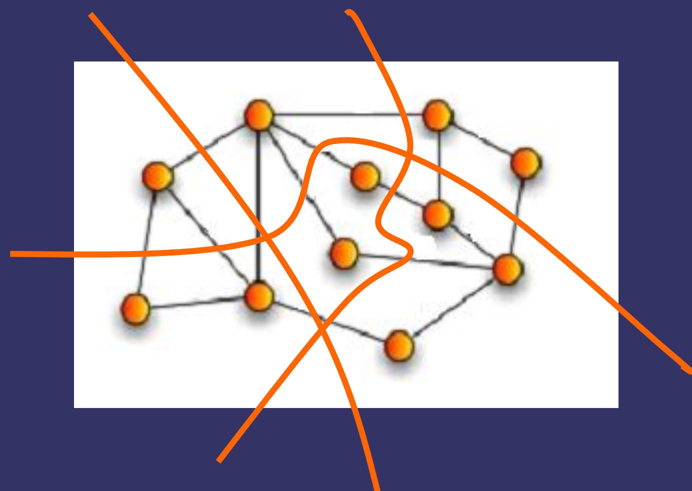
# *Max Cut Problem*

Daniel Natapov



# *Problem Definition*

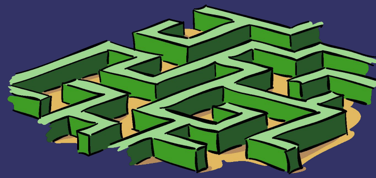
- Given an undirected graph G(V,E), find a cut between the vertices, such that the number of edges crossing the cut is maximal.



# *Max Cut is NP-Hard!*

We show that it is NP-Hard by a reduction from the NAE-3-SAT Problem.

The Not All Equal-3-SAT Problem is very similar to the 3-SAT problem, and can easily be shown to be NP-Hard by a reduction from Circuit SAT.



# *NAE-3-SAT Problem*

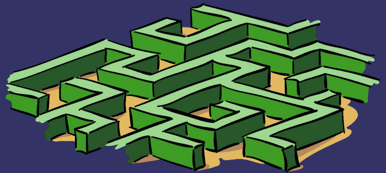
Not All Equal-3-SAT:

- A circuit consisting of a big AND of clauses
- Each clause is the OR of at most 3 literals
- Each literal is a variable or its negation.
- Each clause has at least one true literal and at least one false literal.

does it have a satisfying assignment X?

$\boxed{\text{F T F F T F}}$

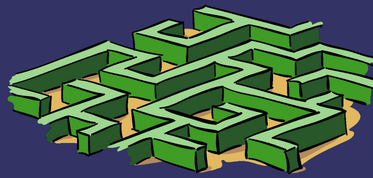
$\overline{x} \text{ or } y \text{ or } \overline{z}$ AND $x \text{ or } \overline{w} \text{ or } a$ AND …



# The Reduction – Step 0

# Max Cut ∈ NP

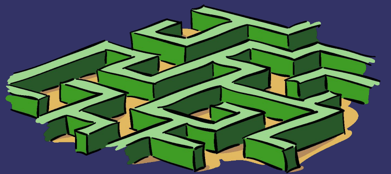
- Change problem to: “Is there a cut of size ≥ K?”
- We can easily check in poly-time, that the size of a given cut is ≥ K.



# *The Reduction – Step 1*
# *What to reduce it to?*

Reduce to NAE-3-SAT

NAE-3-SAT $\leq$ Max Cut



# The Reduction – Step 2

# What is what?

$P_{ne}$ = Max Cut

$P_{is}$ = NAE-3-SAT

w

NP-comp

$I_{ne}$

$S_{ne}$

$I_{is}$

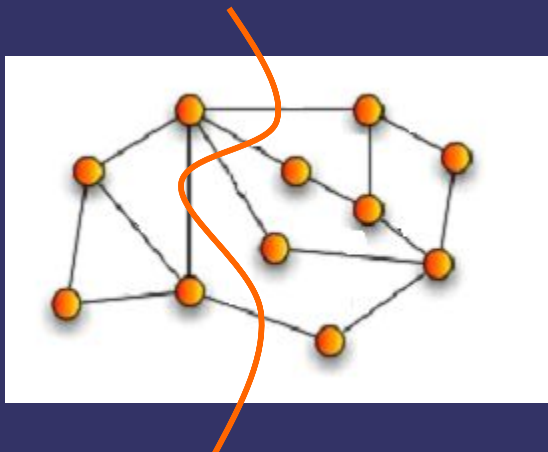
$S_{is}$

w

w

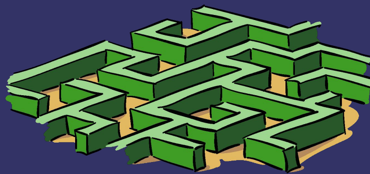
NP-comp

NP-comp



F T F F T F

$\bar{x}$ or y or $\bar{z}$ AND x or $\bar{y}$ or y AND …



# The Reduction – Step 3
# Direction of Reduction and Code

- Want to show that Max Cut is hard
- NAE-3-SAT $\leq$ Max Cut
- Then, since we know NAE-3-SAT is hard, Max Cut must be hard too.

$Alg_{alg}$=NAE-3-SAT
$Alg_{oracle}$=Max Cut

**algorithm** $Alg_{alg}(I_{alg})$

$\langle \textbf{\textit{pre-cond}} \rangle$: $I_{alg}$ is an instance of $P_{alg}$.

$\langle \textbf{\textit{post-cond}} \rangle$: Determine whether $I_{alg}$ has a solution $S_{alg}$ and if so returns it.

```
begin
    I_oracle = InstanceMap(I_alg)
    <ans_oracle, S_oracle> = Alg_oracle(I_oracle)
    if( ans_oracle = Yes) then
        ans_alg = Yes
        S_alg = SolutionMap(S_oracle)
    else
        ans_alg = No
        S_alg = nil
    end if
    return(<ans_alg, S_alg>)
end algorithm
```

# *The Reduction – Step 4*
# *Look for Similarities*

## NAE-3-SAT

Literals

X ¬X Y ¬Y Z

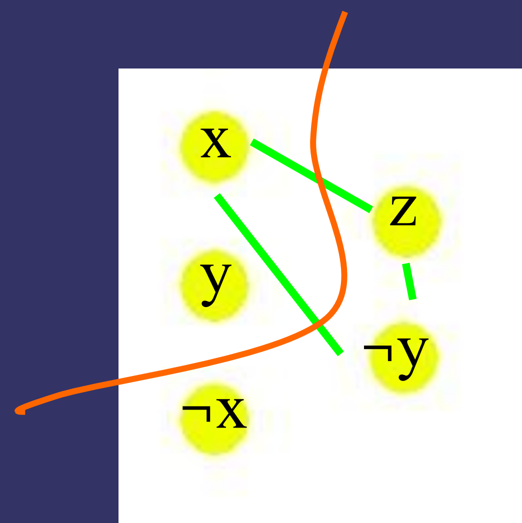
Clauses

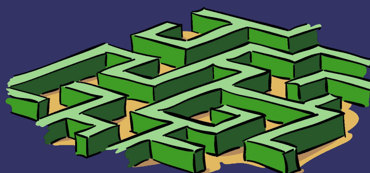
(X v ¬Y v Z)

Boolean Assignment

F T F F T F

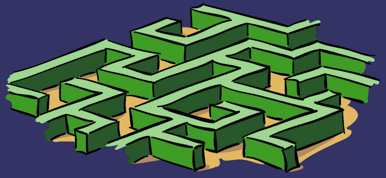
## Max Cut



# The Reduction – Step 5
# Instance Maps

- For every clause $C_i$( A v B v C), i= 1..m, produce a triangle (A, B, C) in the graph.
- If two literals in the clause are the same, the “triangle” has a double edge.
- Finally, for each literal $x_i$, create an edge between $x_i$ and $\neg x_i$ for each time $x_i$ or $\neg x_i$ appear.

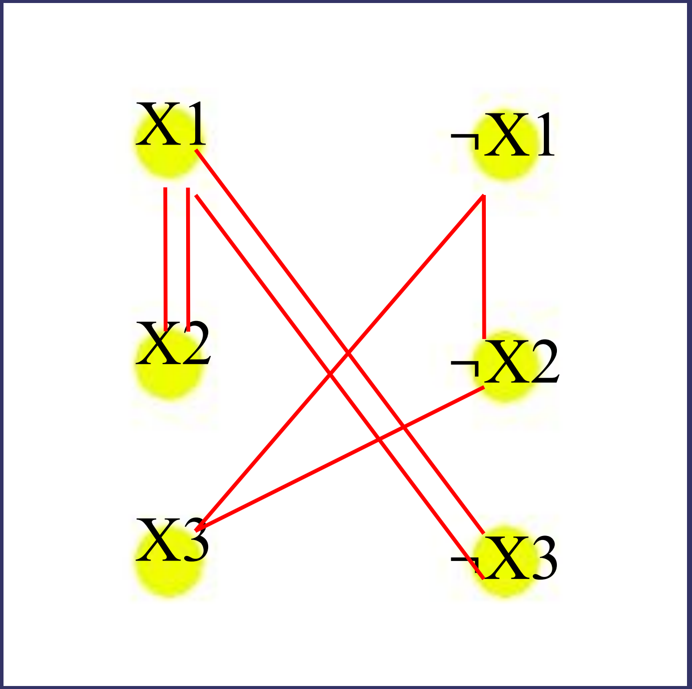


# The Reduction – Step 5
# Instance Maps

For example:

(X1 v X2 v X2) AND (X1 v ¬X3 v ¬X3) AND (¬X1 v ¬X2 v X3)

- For every clause $C_i$( A v B v C), i= 1..m, produce a triangle (A,B, C) in the graph.

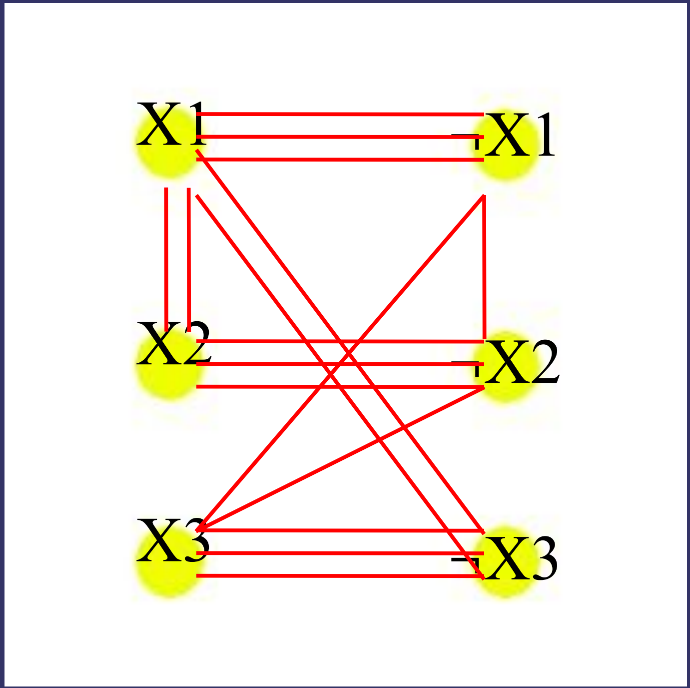


# *The Reduction – Step 5*
# *Instance Maps*

For example:

(X1 v X2 v X2) AND (X1 v ¬X3 v ¬X3) AND (¬X1 v ¬X2 v X3)

- For each literal $x_i$, create an edge between $x_i$ and $\neg x_i$ for each time $x_i$ or $\neg x_i$ appear.

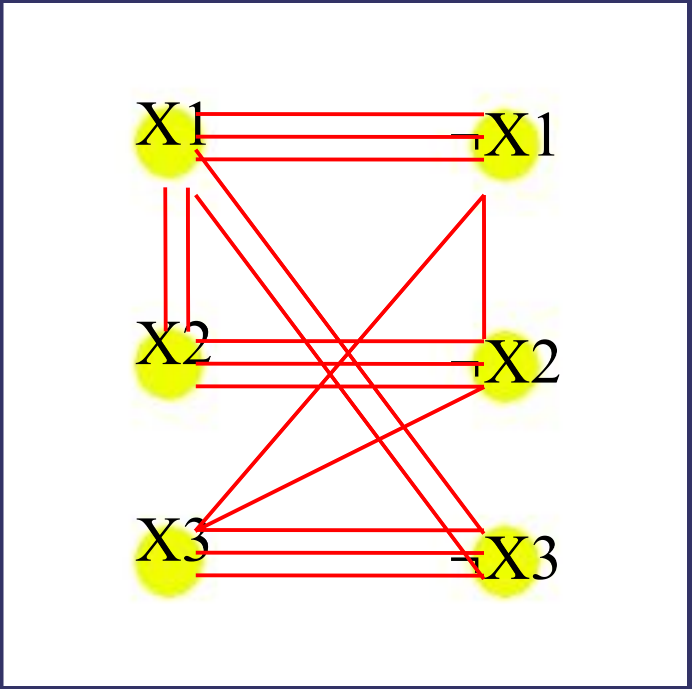


# *The Reduction – Step 5*
# *Instance Maps*

For example:

(X1 v X2 v X2) AND (X1 v ¬X3 v ¬X3) AND (¬X1 v ¬X2 v X3)

- We ask Max Cut, if there exists a cut of size K, where K ≥ 5(number of clauses).
- If yes, then there exists a valid assignment for NAE-3-SAT.

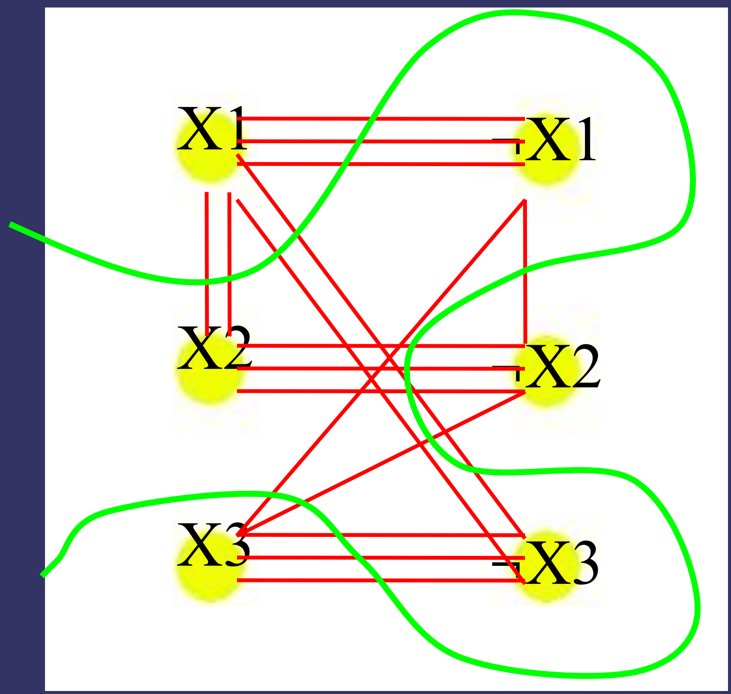


# The Reduction – Step 6
# Solution Map

For example:

(X1 v X2 v X2) AND (X1 v ¬X3 v ¬X3) AND (¬X1 v ¬X2 v X3)

- Max Cut (The oracle) returns a cut. To find the solution to NAE-3-SAT, we assign all vertices on one side of the cut to True, and the ones on the other side to False.

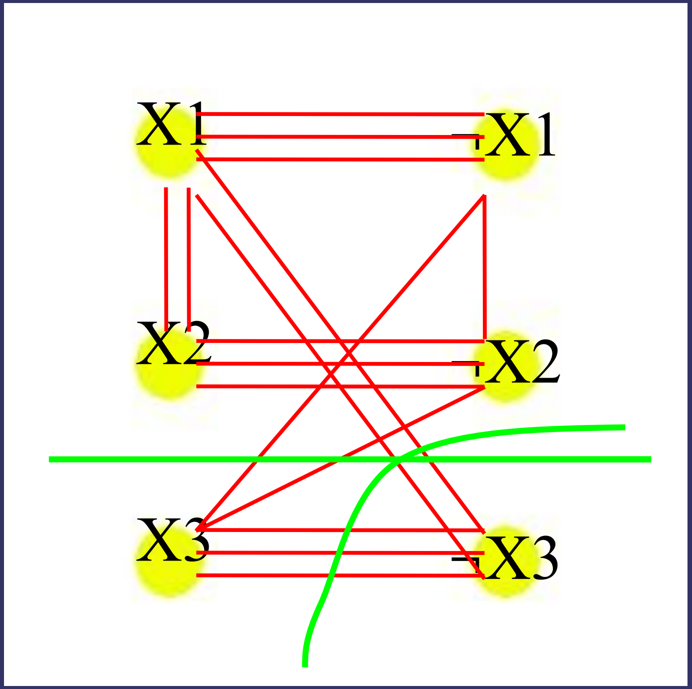



Here: X1=True, X2=False, X3=True

# The Reduction – Step 7
# Valid to Valid

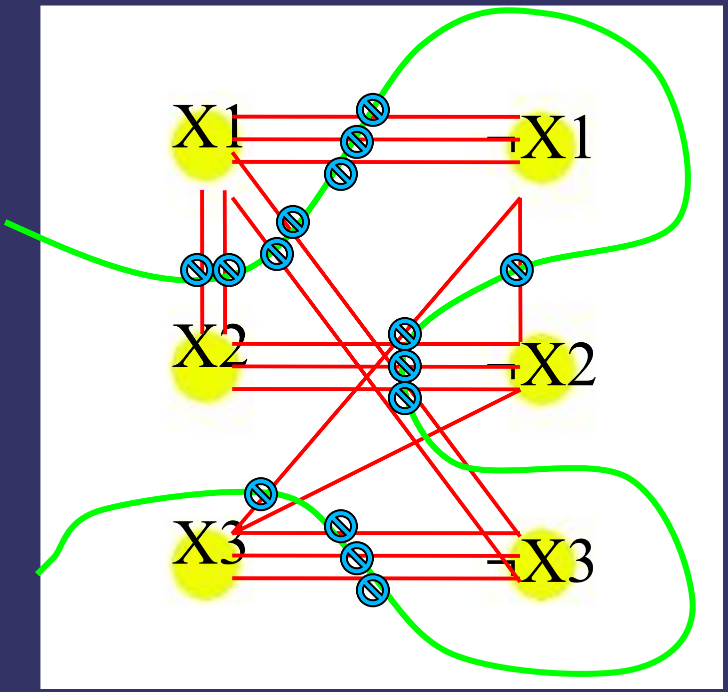
Assume the oracle (Max Cut), finds a cut of size 5(number of clauses).

We can safely assume that for this cut, all Xi are separated from ¬Xi by the cut. If they are on the same side of the cut, they contribute at most 2n edges. Splitting them up would yield n edges from Xi to ¬Xi, plus at least half what they were contributing before, so there is no decrease.



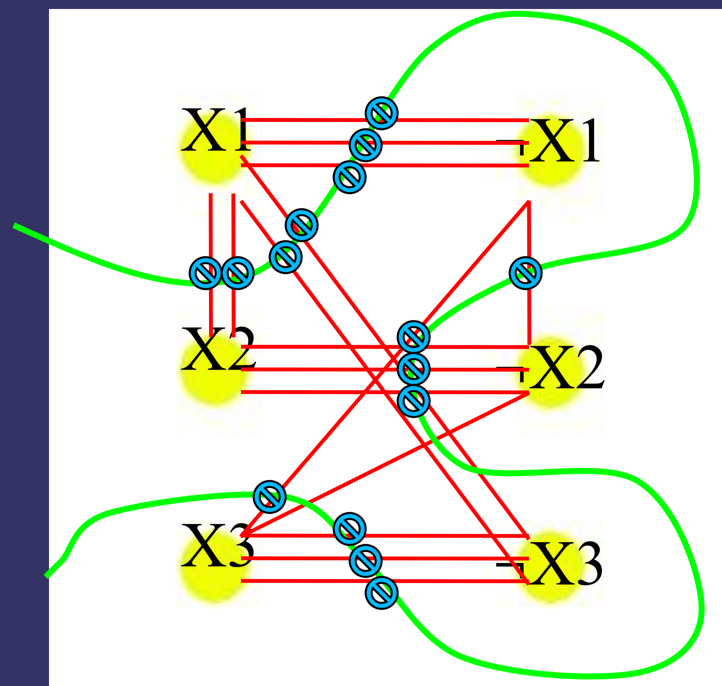
# The Reduction – Step 7
# Valid to Valid

- For our example, we had 3 clauses. Here is one cut whose size is=15. (5*m)
- The number of edges in the cut that connect Xi to ¬Xi is 3m (in our case 9). Basically one edge for every literal.
- The other 2m edges (in our case 6), must come from the triangles.
- Each triangle can contribute at most 2 edges to a cut. Therefore, all *m* triangles are split by the cut.



# *The Reduction – Step 7*
# *Valid to Valid*

- All m triangles are split by the cut.
- Since a triangle is actually a clause of three literals, and every "clause" is split by the cut, by assigning True to one side of the cut and False to the other side of the cut, we ensure that every clause has at least one True and one False literal.
- This satisfies NAE-3-SAT, so if the cut returned my Max Cut is valid, our solution is valid.



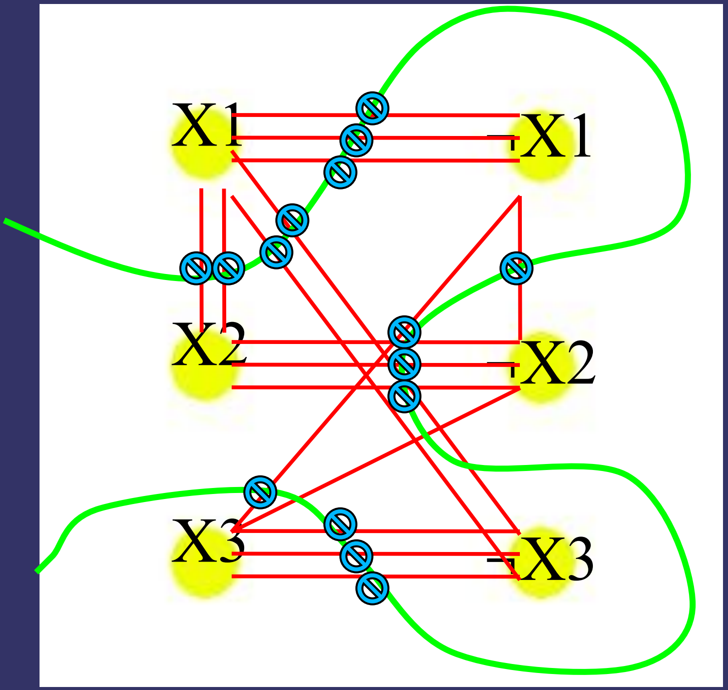
# *The Reduction – Step 7*
# *Valid to Valid*

For our example:

(X1 v X2 v X2) AND (X1 v ¬X3 v ¬X3) AND (¬X1 v ¬X2 v X3)
( T v F v T ) AND (T v F v F) AND ( F v T v F )

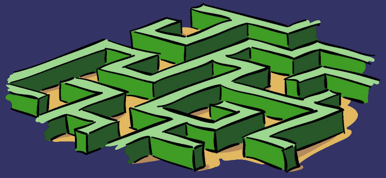
X1 ¬X2 X3 ¬X1 ¬X2 ¬X3

X1: T X2: F X3: T



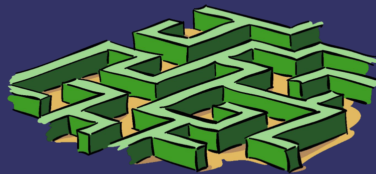
# The Reduction – Steps 8&9
# Reverse Solution Map

Conversely, it is also possible for a valid solution for Max Cut to be found using a NAE-3-SAT oracle, (but it is not covered in this presentation).



# The Reduction – Step 10
# Working Algorithm

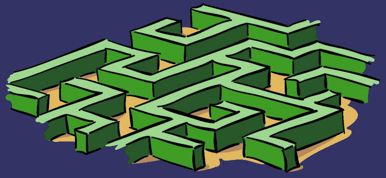
- We now have a working algorithm for the NAE-3-SAT problem.
- We translate the inputted list of clauses into a graph, and ask our Oracle: “Given this graph, is there a cut of size 5m?”
- If the Oracle says yes, and returns a cut, we assign True to all literals on one side of the cut, and False to all literals on the other side of the cut.
- We have a valid assignment.



# The Reduction – Step 11
# Running Time?

- We can create an instance map (clauses -> graph) in polynomial time.
- We can also create a solution map (cut -> Boolean assignment) in polynomial time.
- If our Max Cut Oracle can answer the question in polynomial time, we can solve NAE-3-SAT in poly time!

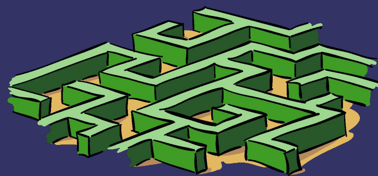
- (Of course, so far no known polynomial time algorithm for Max Cut is known).



# *Max Cut – Running Time*

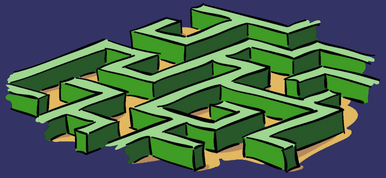
- The best known algorithm for finding an optimal solution for the Max Cut problem runs in $2^{\theta(n)}$ time.

- Is there a better way?



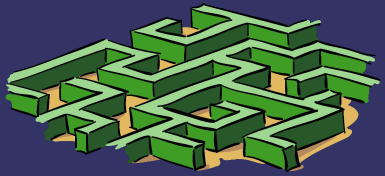
# *Max Cut – Randomized Algorithm*

- Here is a simple approximation Max Cut Algorithm instead:
- The cut divides the vertices into two sets. For each vertex…



# *Max Cut – Randomized Algorithm*


- Flip a coin! To see in which of the two sets the vertex lies.
- Each edge crosses over the cut with probability ½. The expected number of edges to cross over the cut is $|E|/2$.
- Since the optimal solution can not have more than all the edges cross over the cut, the expected solution is within a factor of 2.
- And the Randomized Algorithm runs in $\theta(n)$ time!



# *That's it!*

Questions? Comments? Praise?