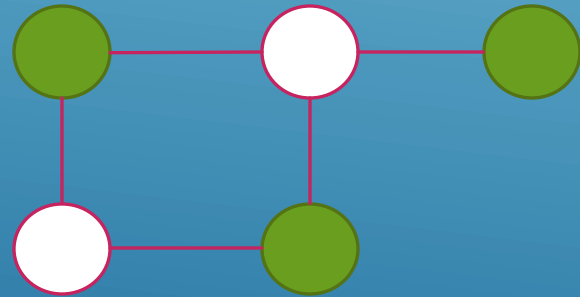


THE LIMITS OF KNOWLEDGE

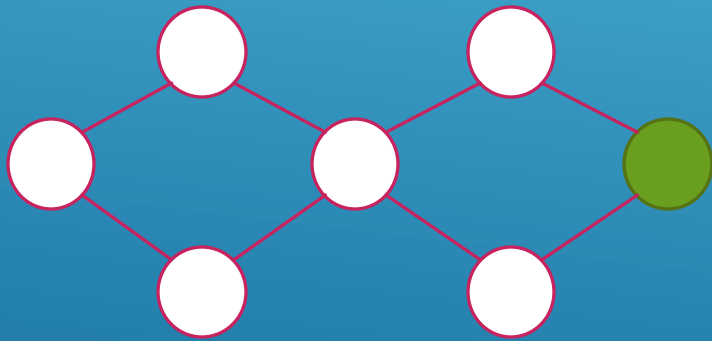


Given an undirected graph $G = \langle V, E \rangle$ and integer k , you want to find a set of nodes $S \subseteq V$ such that $|S| \geq k$, and there are no edges with both endpoints in S .



INDEPENDENT SET

Always choose the node with smallest degree?



GREEDY?



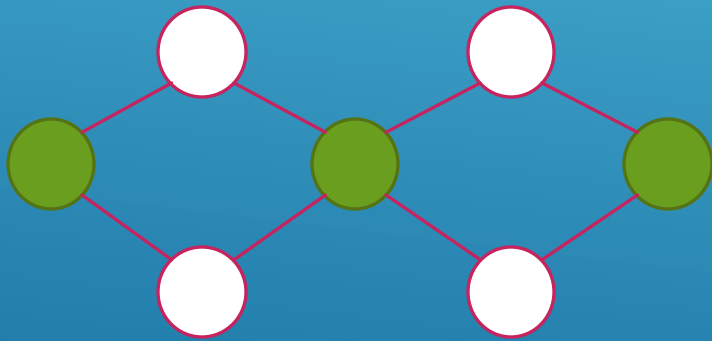
DYNAMIC PROGRAMMING?

Do I include the next node?

- ▶ This depends on all of the previous nodes you have selected.
- ▶ You will need 2^n subproblems to keep track of which nodes are still valid choices.

...this is a **hard** problem.

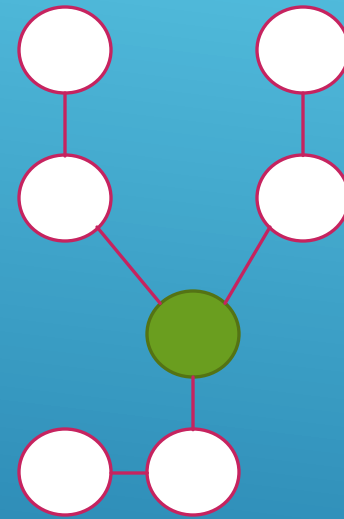
Given an undirected graph $G = \langle V, E \rangle$ and integer k , you want to find a set of nodes $S \subseteq V$ such that $|S| \leq k$, and all edges have an endpoint in S .



VERTEX COVER

Always choose the node with highest degree?

GREEDY?



I'M CHEATING...

These two questions, given together, were an **unfair** test.

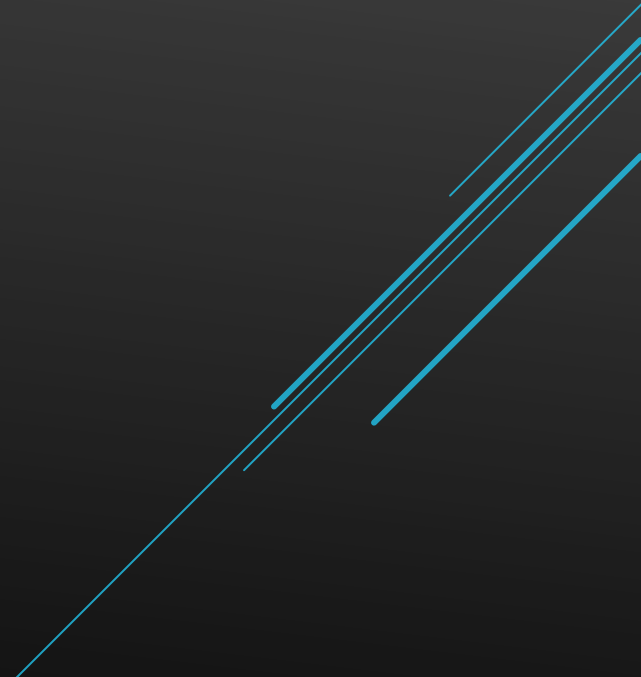
- ▶ They were actually the **same problem**.
- ▶ More specifically, $IS \leq_p VC$, and $VC \leq_p IS$.
- ▶ So, if you could have solved either of these problems, you could have solved the other as well.

Take a min-sized vertex cover C .

- ▶ $V - C$ is a max-sized independent set.

Take a max-sized independent set I .

- ▶ $V - I$ is a min-sized vertex cover.



Given a set U of n elements, m subsets $S_1, \dots, S_m \subseteq U$, and integer k , you want to find a collection of $\leq k$ subsets whose union is U .

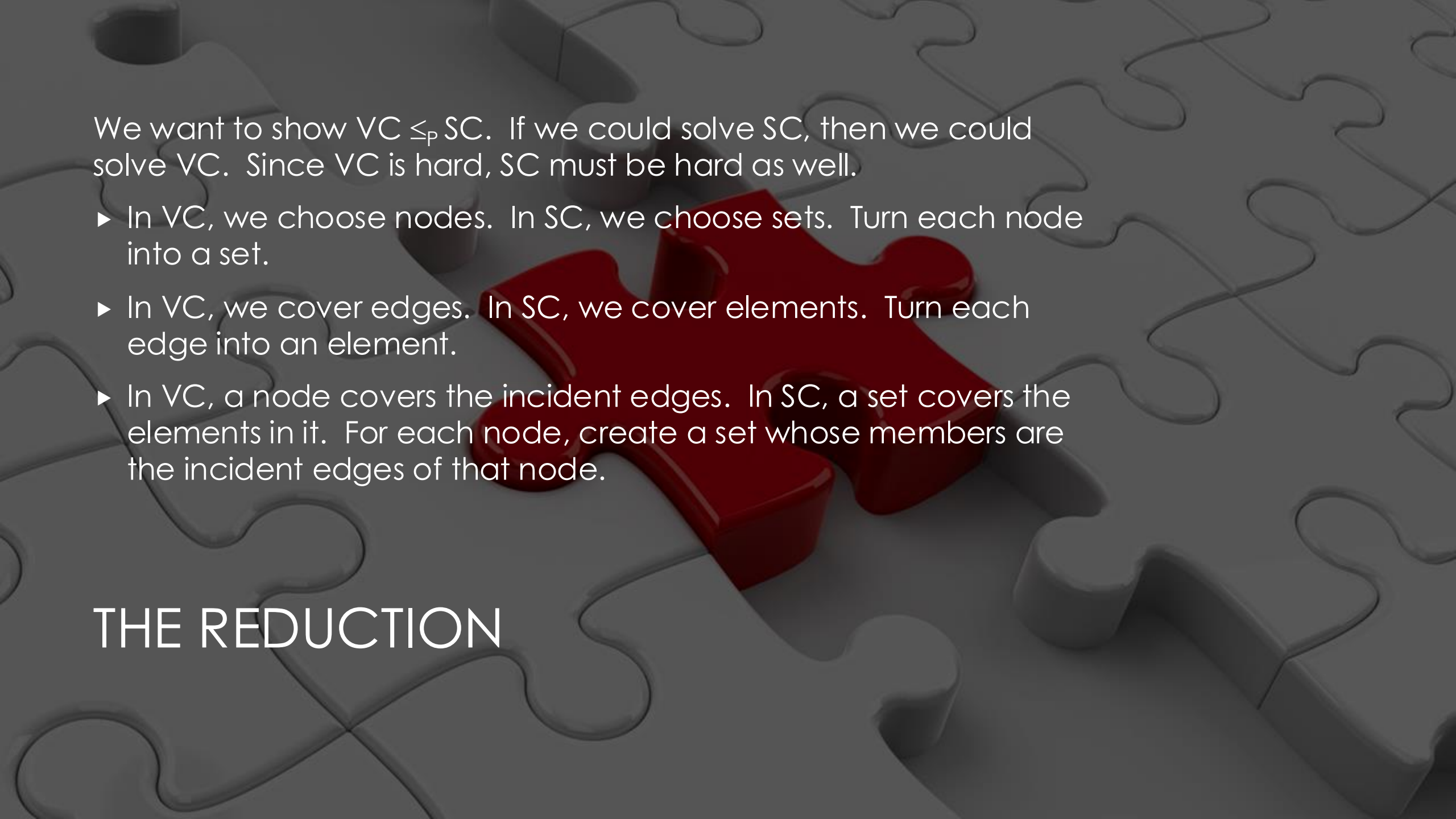
Prove that Set Cover is a hard problem, given that Independent Set and Vertex Cover are hard problems.

Do we want to show $SC \leq_p VC$?

- No, that would only show that a specific solution to SC is bad: there might be a much better solution.

SET COVER

	Contents
S_1	{1,2}
S_2	{2,3,5}
S_3	{2,6}
S_4	{2,4,5}
S_5	{3,6}
S_6	{1,5}

A 3D puzzle with one red piece standing out. The puzzle is composed of many grey pieces, and one red piece is in the center, slightly raised. The background is a dark grey with a subtle pattern of puzzle pieces.

We want to show $VC \leq_p SC$. If we could solve SC, then we could solve VC. Since VC is hard, SC must be hard as well.

- ▶ In VC, we choose nodes. In SC, we choose sets. Turn each node into a set.
- ▶ In VC, we cover edges. In SC, we cover elements. Turn each edge into an element.
- ▶ In VC, a node covers the incident edges. In SC, a set covers the elements in it. For each node, create a set whose members are the incident edges of that node.

THE REDUCTION

Suppose you have a difficult IS instance you cannot solve.

- ▶ Albert Einstein walks in, looks at it, and says “Yes! There is an Independent Set of size k ,” and then he points out the correct nodes.
- ▶ How long would it take you to **verify** his solution is a valid Independent Set of size k ?
- ▶ $\Theta(n+m)$: count up the nodes in the Independent Set, and verify each edge has at most one endpoint in the set.

The class **NP** is the set of problems with polynomial-time **verifiers**.

THE CLASS NP

It is much easier to verify an answer is correct than to come up with that solution on your own.

- ▶ NP stands for **Nondeterministic Polynomial-Time**
- ▶ Formally, a problem is in NP if every instance that is supposed to return “yes” has a polynomial-length “solution” or “hint” that will convince a grader the answer is “yes” in polynomial-time.

Which of the problems that we have seen are in NP?

- ▶ All of them (IS, VC, SC, Sorting, Shortest Path, Array Search, etc)

NP, CONTINUED

PROPERTIES OF NP

Is $P \subseteq NP$?

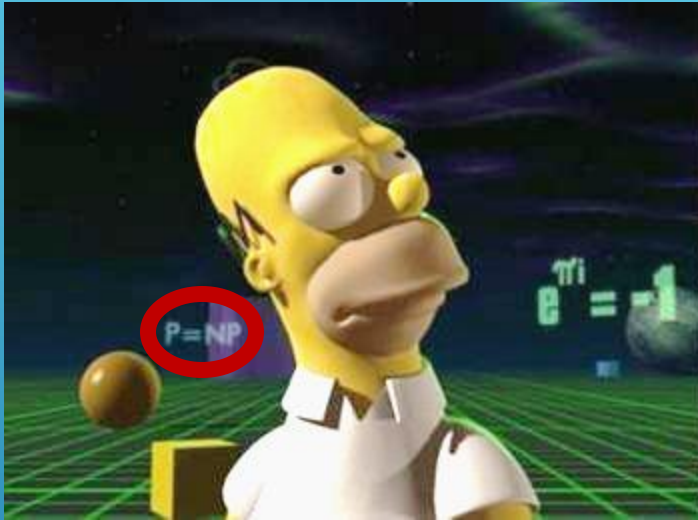
- ▶ Yes. If a problem is poly-time solvable, you can simply ignore the solution/hint and solve it yourself in the given time limit.

Is $P \subsetneq NP$?

- ▶ That's the million dollar question!

Are there problems that are definitely not in NP?

- ▶ Yes: list all permutations of an input string of length n (the length of the answer is longer than polynomial, so you can't even read it, much less verify it, in time).



P = NP?

THE HARDEST PROBLEM IN NP

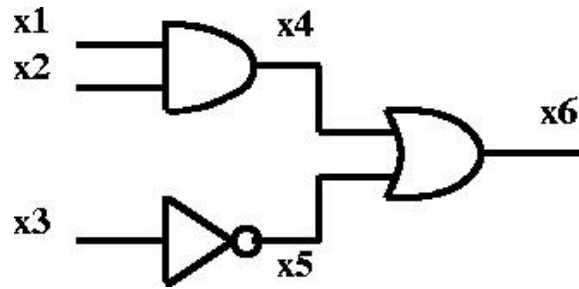
If we want to claim the million dollar prize, here is a strategy:

1. Identify the “hardest” problem in NP.
2. Either give a poly-time algorithm for it, or prove no such algorithm is possible.

It is not clear what it would even **mean** to be the hardest problem in NP. What kind of traits would such a problem have?

► $\forall x \in \text{NP}, x \leq_p \text{“hardest problem”}$

The “hardest problem” in NP is a problem called **circuit-SAT**.



Given a digital circuit of AND, OR, and NOT gates, is there a way to set the inputs s.t. the output will be 1?

- ▶ Is circuit-SAT \in NP?
- ▶ Yes, given the inputs, we can trace outputs through the circuit and calculate the answer in linear time.

The Cook-Levin Theorem asserts that all problems in NP are poly-time reducible to circuit-SAT.

- ▶ The proof is beyond the scope of this course. If you really want to know, you can take CSCI 475 with me at some point in the future.

CIRCUIT-SAT

The Cook-Levin Theorem established Circuit-SAT as an **NP-Complete** problem:

1. $\text{Circuit-SAT} \in \text{NP}$
2. $\forall x \in \text{NP}, x \leq_p \text{Circuit-SAT}$

If you have a new problem Z , and you show:

1. $Z \in \text{NP}$
2. $\text{Circuit-SAT} \leq_p Z$

Then what does this imply?

- Z is also NP-Complete!

NP-COMPLETE PROBLEMS

If you have another problem Y , and you show:

1. $Y \in \text{NP}$
2. $Y \leq_p \text{Circuit-SAT}$

What does this imply?

- ▶ Nothing other than that $Y \in \text{NP}$. We already knew all problems in NP could be reduced to Circuit-SAT, and we reinvented the wheel.
- ▶ This simply gives a **bad** solution to Y . There might be a much better one!

PROPERTIES OF NP-COMPLETE PROBLEMS

You are given a CNF formula with n variables x_1, \dots, x_n , and m clauses C_1, \dots, C_m , where each clause is a disjunction of exactly 3 literals.

- ▶ A CNF formula is a conjunction of the clauses.
- ▶ A CNF clause is a disjunction of variables and negations of variables.
- ▶ $(x_1 \vee \neg x_2 \vee x_4) \wedge (x_2 \vee x_3 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee \neg x_4)$
- ▶ Setting all variables to true works here.
- ▶ Is 3-SAT \in NP?
- ▶ Walk through the clauses, evaluating if each one is true: takes linear time.
- ▶ We will show $\text{Circuit-SAT} \leq_p \text{3-SAT}$ (verify this is the correct direction!)

3-SAT

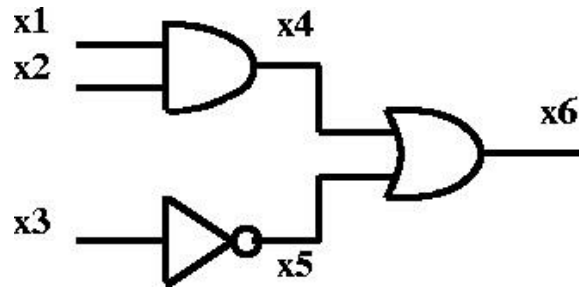
What do you think of this reduction?

- ▶ Given a 3-SAT formula, connect the 3 literals in each clause with an OR-gate, and all the clauses with an AND-gate. Put NOT-gates in front of negated variables.

This reduction goes the *wrong* way. Circuit-SAT is a more generalized version of 3-SAT, and is thus the harder problem. Getting a poly-time reduction in the correct direction will be much trickier.

- ▶ We are given a circuit-SAT circuit, and we need to show how to transform it into a 3-SAT formula in polynomial time!

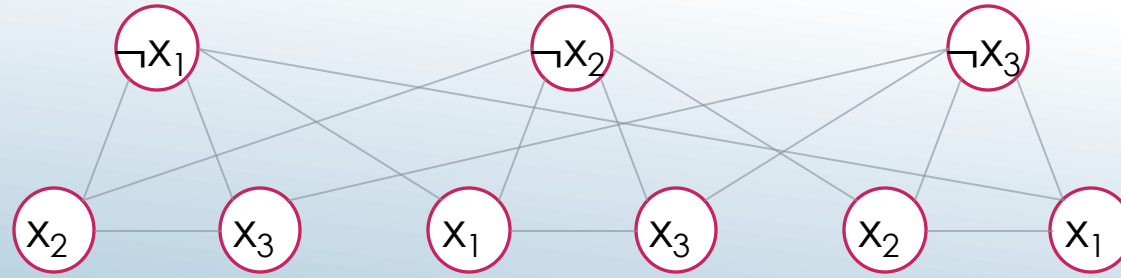
A REDUCTION?



1. Add variable names to every intermediate wire in the circuit.
2. Hard-code the output: $C_1 = (x_6)$
3. Transform the NOT-gates: $C_2 = (x_3 \vee x_5)$ and $C_3 = (\neg x_3 \vee \neg x_5)$
4. Transform the OR-gates: $C_4 = (x_6 \vee \neg x_4)$, $C_5 = (x_6 \vee \neg x_5)$ and $C_6 = (\neg x_6 \vee x_4 \vee x_5)$
5. Transform the AND-gates: $C_7 = (x_1 \vee \neg x_4)$, $C_8 = (x_2 \vee \neg x_4)$ and $C_9 = (\neg x_1 \vee \neg x_2 \vee x_4)$
6. Add repeat literals to make every clause length-three: $C_1 = (x_6 \vee x_6 \vee x_6)$

This was only an example! The above reduction works regardless of the input circuit!

A REDUCTION!



3-SAT \leq_p IS:

- ▶ You are given a 3-SAT instance such as $(\neg X_1 \vee X_2 \vee X_3) \wedge (X_1 \vee \neg X_2 \vee X_3) \wedge (X_1 \vee X_2 \vee \neg X_3)$
- ▶ Add 3 nodes in a triangle for each clause.
- ▶ Add edges between inconsistent variable assignments.
- ▶ Find an IS of size m , which is the variable to satisfy each clause.

INDEPENDENT SET

That means we have 5
NP-Complete problems now!

Given a set U of n elements, m subsets $S_1, \dots, S_m \subseteq U$, and an integer k . You want to find a collection of k subsets which have no overlap.

IS \leq_p SP:

- ▶ In IS we choose nodes, and in SP we choose subsets. Each node becomes a subset.
- ▶ In IS we conflict if there is an edge in common, and in SP we conflict if there is an element in common. Each edge becomes an element.
- ▶ In IS, a node conflicts with the edges incident to it, and in SP, a subset conflicts with the elements in it. So, each node becomes a set containing the edges incident to it.

SET PACKING

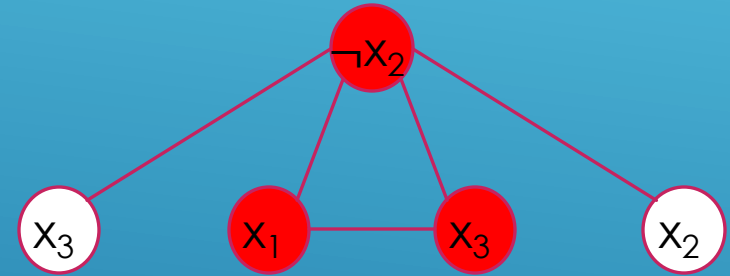
	Contents
S_1	$\{1,2\}$
S_2	$\{2,3,5\}$
S_3	$\{2,6\}$
S_4	$\{2,4,5\}$
S_5	$\{3,6\}$
S_6	$\{1,5\}$

Given an undirected graph G and integer k , is there a set S of k nodes such that there is an edge between every pair of nodes in S ?

K -Clique is in NP: given a clique, verify every pair of edges exists in $O(m)$ time.

$IS \leq_p k$ -Clique:

- ▶ Create $G' = \langle V, E' \rangle$, where $\langle u, v \rangle \in E'$ iff $\langle u, v \rangle \notin E$
- ▶ Find a clique of size k .
- ▶ These nodes form an independent set on G .



K-CLIQUE

DOES $P = NP$?

There are a lot of NP-Complete problems. That's why we're pretty sure $P \neq NP$

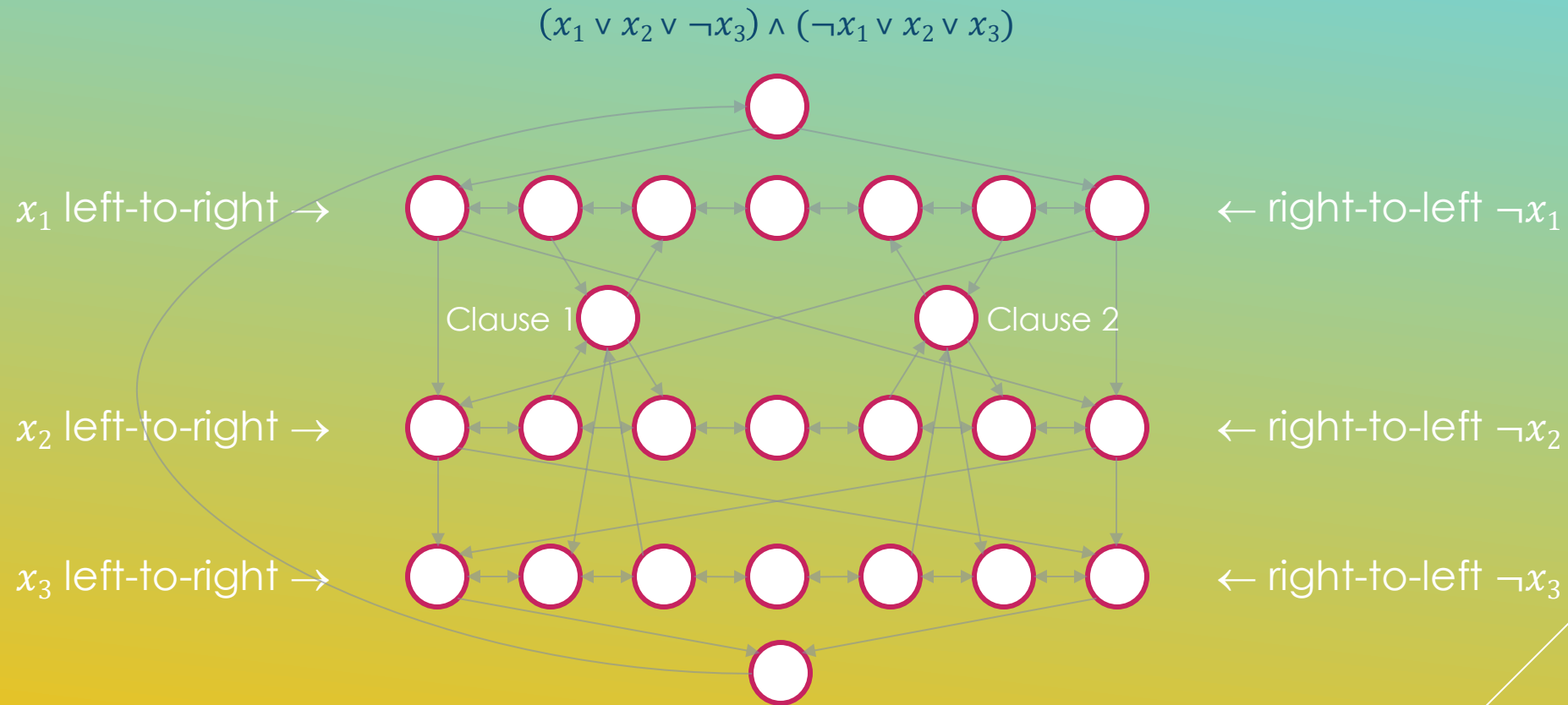
- ▶ To prove $P = NP$, all we need to do is find a poly-time solution for any one of these problems.
- ▶ Given how much effort we've put into this, we would have expected to solve the problem by now if $P = NP$.
- ▶ Showing a problem *doesn't* have a polytime solution is much harder, so we think that this reflects reality.

DIRECTED HAMILTONIAN CYCLE

Given a directed graph G , find a cycle that visits every node.

- ▶ Given a cycle, check off each node to ensure you visit every node exactly once and end where you started, and verify there is an edge between each adjacent pair of nodes in the cycle. This can be done in linear time, so $\text{DHC} \in \text{NP}$
- ▶ We will show $3\text{-SAT} \leq_p \text{DHC}$
- ▶ We will make a bi-directional chain of nodes for each variable. You can traverse the chain either left-to-right, or right-to-left, corresponding to whether that variable is true or false.
- ▶ We will add clause nodes that can only be visited if you are traversing the chain of nodes in the correct direction.

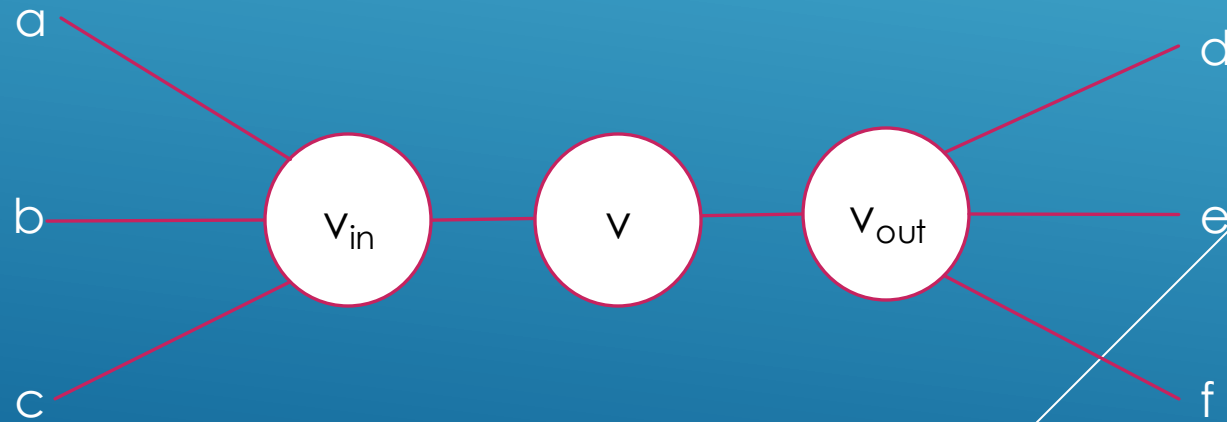
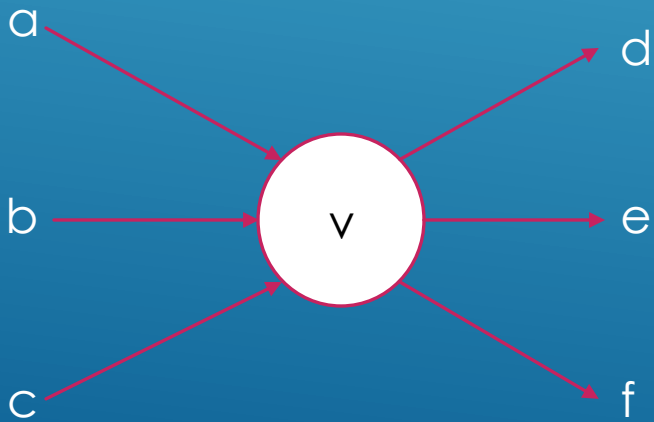
THE REDUCTION




UNDIRECTED HAMILTONIAN CYCLE

Given an **undirected** graph G , is there a cycle that visits every node?

- ▶ In NP for the same reason as DHC.
- ▶ We will show $\text{DHC} \leq_p \text{UHC}$
- ▶ This problem is potentially easier than DHC, as any undirected graph can be represented as a directed graph.



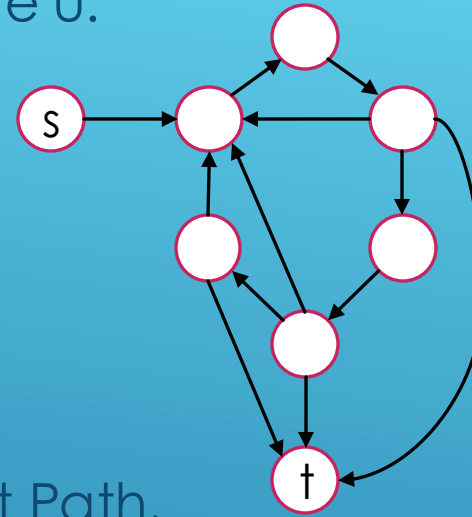


Given a directed graph G and integer k , is there a path in G of length $\geq k$?


- ▶ Directed Longest Path is in NP, because given a path you can check off the nodes to ensure there are no repeats, and you can verify that every required edge exists, all in linear time.
- ▶ We will show $\text{DHC} \leq_p \text{DLP}$

DIRECTED LONGEST PATH & DIRECTED HAMILTONIAN PATH

- ▶ Add a node s , connected to an arbitrary node u .
- ▶ Add a node t , connected from all nodes with an edge to u .
- ▶ Find a path of length $n+2$, or a Hamiltonian Path, on the new graph.
- ▶ This reduction takes linear time.
- ▶ This reduction works for both Directed Longest Path, and Directed Hamiltonian Path.



THE REDUCTION

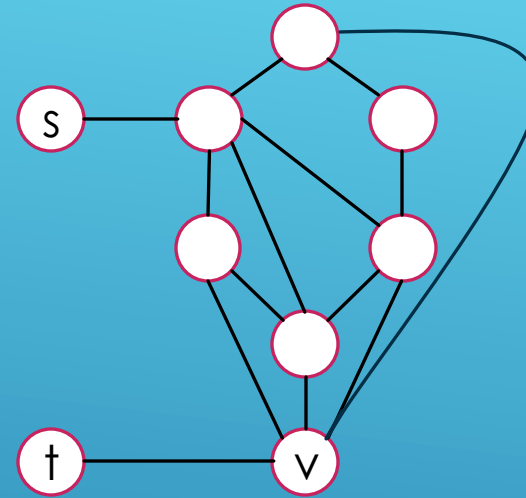


UNDIRECTED LONGEST PATH & UNDIRECTED HAMILTONIAN PATH

Given an undirected graph G and integer k , is there a path in G of length $\geq k$?

- ▶ Undirected Longest Path is in NP, because given a path you can check off the nodes to ensure there are no repeats, and you can verify that every required edge exists, all in linear time.
- ▶ We will show $\text{UHC} \leq_p \text{ULP}$

- ▶ Add a node s with an edge to an arbitrary node u
- ▶ Add a node v with edges from all nodes that have edges to u .
- ▶ Add a node t with an edge to v .
- ▶ Find a path of length $n+3$, or a Hamiltonian Path, on the new graph.
- ▶ This reduction takes linear time.
- ▶ This reduction works for both Undirected Longest Path, and Undirected Hamiltonian Path.



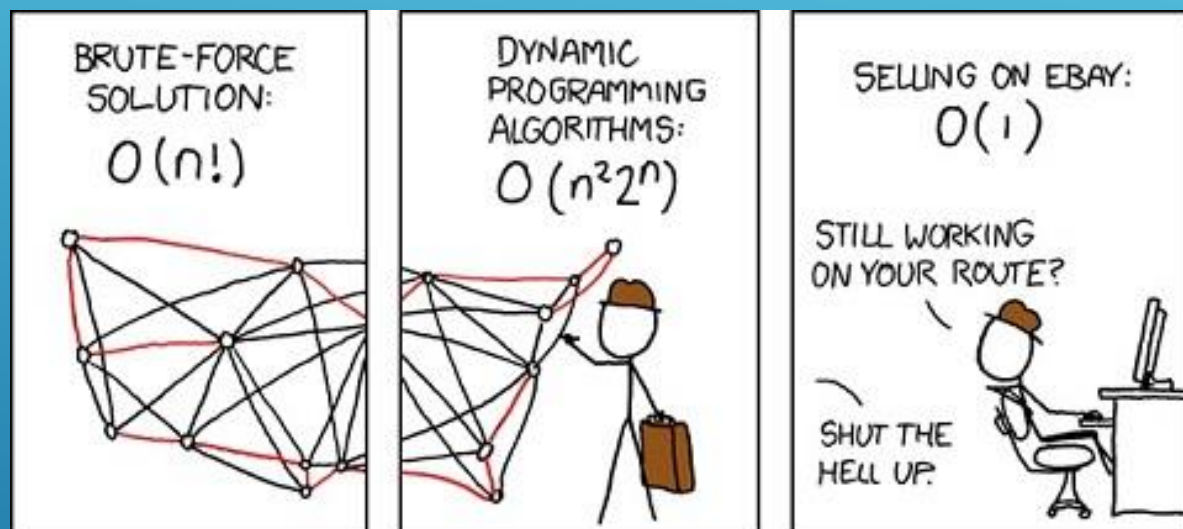
THE REDUCTION

Given a set of n cities, where the distance between cities u and v is specified by $d(u,v)$ which satisfies triangle inequality, and a value D , find a tour of length $\leq D$.

- ▶ TSP \in NP: Check off cities when visited to ensure everything is covered, and sum up the total distance traveled to ensure it is bounded by D . Takes linear time.
- ▶ $\text{UHC} \leq_p \text{TSP}$
- ▶ Add edge weight 1 to all existing edges.
- ▶ Add all missing edges with weight 2.
- ▶ Find a tour with $D=n$.
- ▶ Ensures that we never use the missing edges, while also maintaining triangle inequality.

TRAVELLING SALESPERSON PROBLEM

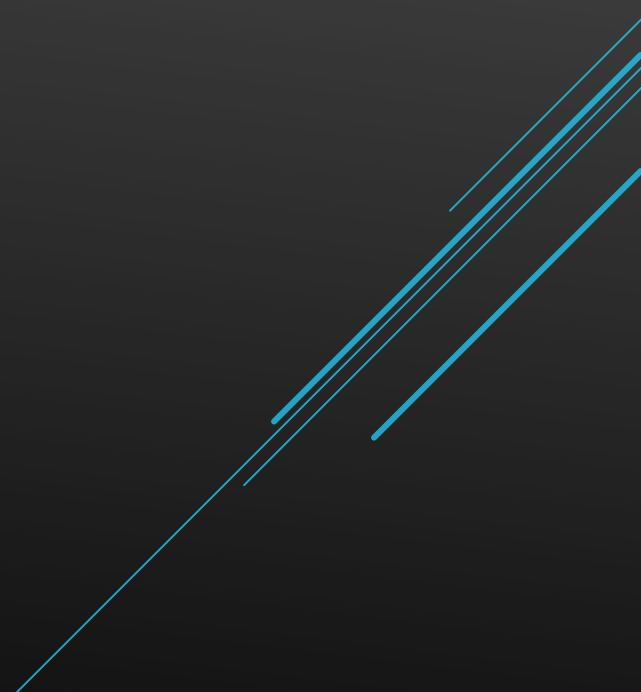
XKCD #399



SUBSET SUM

In the Subset Sum problem, you are given a set of integers w_1, \dots, w_n and an integer W , and you want to determine if there is a subset of the integers that add up exactly to W .

- ▶ Subset Sum is in NP, because you can add up the provided integers in linear time to determine if they equal the target.
- ▶ We will show $3\text{-SAT} \leq_p \text{SS}$



As an example, consider:

$$(x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$$

	x_1	x_2	x_3	C_1	C_2	C_3	C_4
w_1	1	0	0	1	0	0	1
$\neg w_1$	1	0	0	0	1	1	0
w_2	0	1	0	0	0	0	1
$\neg w_2$	0	1	0	1	1	1	0
w_3	0	0	1	0	0	1	1
$\neg w_3$	0	0	1	1	1	0	0
W	1	1	1	4	4	4	4

	x_1	x_2	x_3	C_1	C_2	C_3	C_4
d_1	0	0	0	1	0	0	0
d_1'	0	0	0	2	0	0	0
d_2	0	0	0	0	1	0	0
d_2'	0	0	0	0	2	0	0
d_3	0	0	0	0	0	1	0
d_3'	0	0	0	0	0	2	0
d_4	0	0	0	0	0	0	1
d_4'	0	0	0	0	0	0	2

THE REDUCTION

HOW DO YOU RECOGNIZE AN NP-COMPLETE PROBLEM?

There are no shortcuts: you have to find the reduction or the algorithm to know for sure.

- ▶ Longest Path is NP-Complete
- ▶ Longest Path on a DAG is easy!
- ▶ 3-SAT is NP-Complete
- ▶ 2-SAT is easy!
- ▶ Independent Set is NP-Complete
- ▶ Independent Set on a tree is easy!

3-COLOR

Given an undirected graph G , you want color each node one of 3 colors Cardinal, Gold, and Blue, so that no edge has the same color on both endpoints.

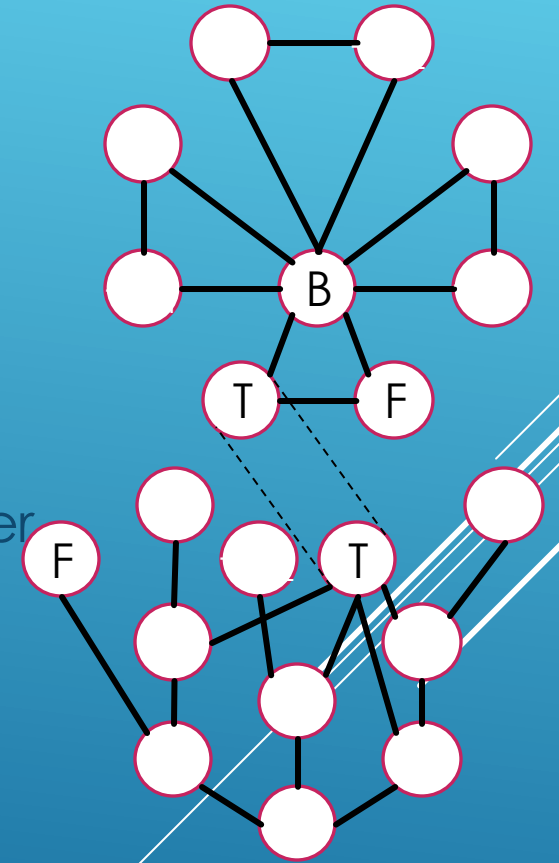
- ▶ 3-Color is in NP, because given a coloring, you can iterate over the edges and verify they don't have the same color on their endpoints. This takes linear time.

We will show $3\text{-SAT} \leq_p 3\text{-Color}$

Create 3 nodes in a triangle: the True node, the False node, and the Base node.

- ▶ Create two nodes for each variable, connected to each other, and to the base node.
- ▶ Create 6 nodes for each clause, connected to each other, the true node, the false node, and the three literals in that clause, in the manner specified by the diagram.

THE REDUCTION



Given an undirected graph G , you want color each node one of k colors, so that no edge has the same color on both endpoints.

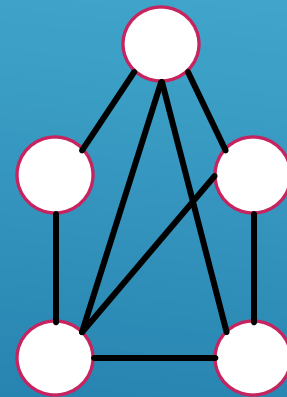
- ▶ k -Color is in NP, because given a coloring, you can iterate over the edges and verify they don't have the same color on their endpoints. This takes linear time.
- ▶ We will show that $k\text{-Color} \leq_p (k+1)\text{-Color}$

K-COLOR

Create a new node, with an edge to all existing nodes.

- ▶ The new node must be a different color than all other nodes, so this will require exactly one more color than the original instance.

THE REDUCTION

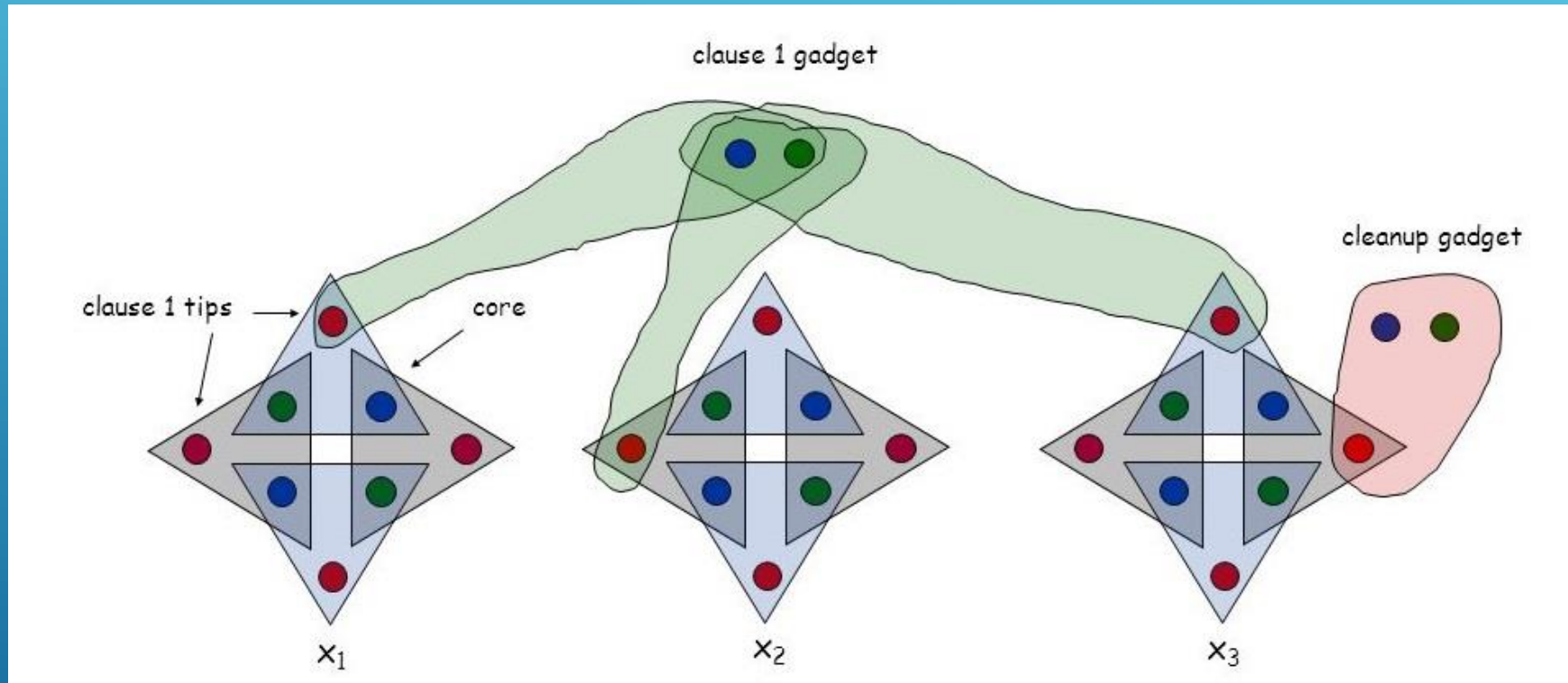


Given n instructors, n courses, n times, and a list 3-tuples indicating valid pairings of courses, times, and instructors, find an assignment where all instructors teach one class, all courses are taught by one instructor, and every timeslot is utilized by one class.

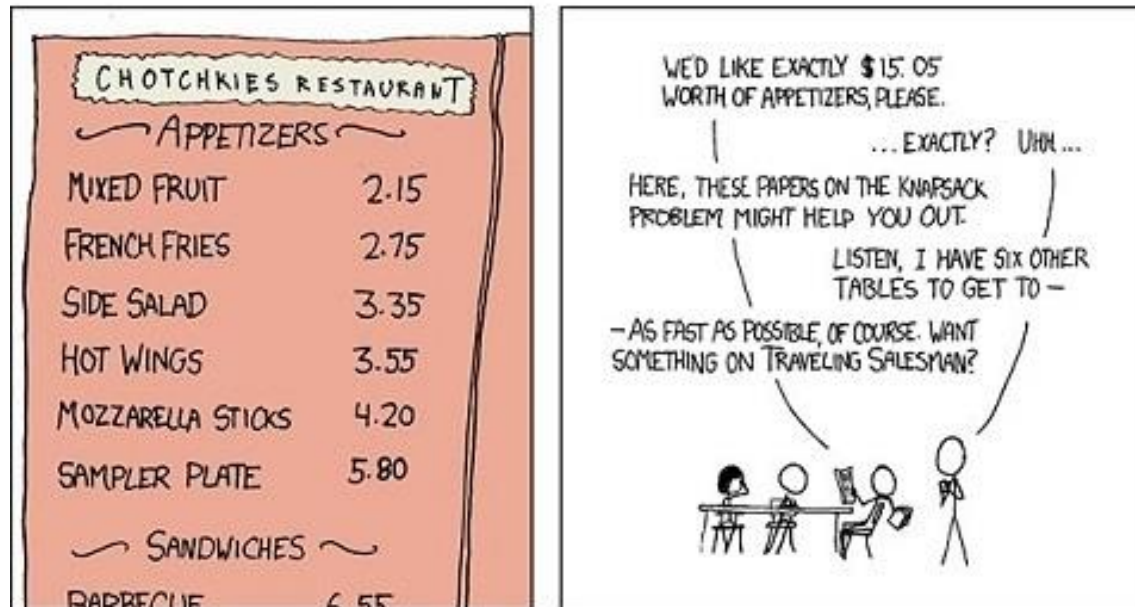
Instructor	Aaron	Aaron	Aaron	Brendan	Brendan	Xing	Xing	Xing
Course	170	170	270	104	170	104	170	270
Time	1pm	11am	11am	1pm	11am	3pm	3pm	1pm

3-D MATCHING

THE REDUCTION



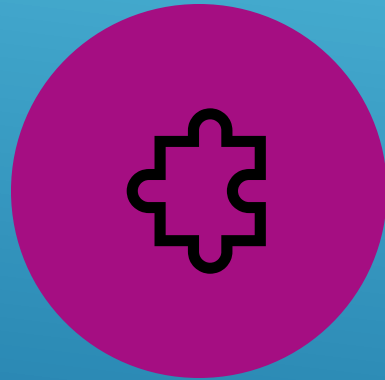
MY HOBBY:
EMBEDDING NP-COMPLETE PROBLEMS IN RESTAURANT ORDERS



XKCD #287: NP- COMPLETE

General solutions
get you a 50% tip.

EXTRA PRACTICE



CHAPTER 28



EXERCISES 5, 6, 7, 8, 12,
14, 17, 26, 28, 29, 36