Reducibility and NP-Completeness

Computational Complexity Theory

- **Computational Complexity Theory** is the study of how much of a given resource (such as time, space, parallelism, algebraic operations, communication) is required to solve important problems.

Classification of Problems

- Q. Which problems will we be able to solve in practice?
- A working definition. [Cobham 1964, Edmonds 1965, Rabin 1966] Those with polynomial-time algorithms.

Yes	Probably no
Shortest path	Longest path
Matching	3D-matching
Min cut	Max cut
2-SAT	3-SAT
Planar 4-color	Planar 3-color
Bipartite vertex cover	Vertex cover

Primality testing	Factoring

Tractable Problems

- We have generally studied tractable problems (solvable in polynomial time).

- Algorithm design patterns. Examples.
  - Greed. $O(n \log n)$ activity scheduling.
  - Divide-and-conquer. $O(n \log n)$ merge sort.
  - Dynamic programming. $O(n \log n)$ activity scheduling with profits.
  - Duality. $O(n^3)$ bipartite matching.

Intractable Problems

- There are other problems that provably require exponential-time.
- Examples:
  - Given a Turing machine, does it halt in at most k steps on any finite input?
  - Given a board position in an n-by-n generalization of chess, can black guarantee a win?

Impossible Problems

- There are other problems that cannot be solved by any algorithm.

The Halting Problem

- The halting problem is a particular decision problem:
  - Given a description of a program and a finite input, decide whether the program will halt or run forever on that input.
- A general algorithm to solve the halting problem for all possible program-input pairs cannot exist: The halting problem is ***undecidable.***

NP Completeness

- Bad news. Huge number of fundamental problems have defied classification for decades.

- Some good news. Using the technique of reduction, we can show that these fundamental problems are "computationally equivalent" and appear to be different manifestations of one really hard problem.

Optimization Problems

Ingredients:

- Instances: The possible inputs to the problem.
- Solutions for Instance: Each instance has an exponentially large set of solutions.
- Cost of Solution: Each solution has an easy to compute cost or value.

Optimization Problems

Specification of an Optimization Problem

- Preconditions: The input is one instance.
- Postconditions:
The output is one of the valid solutions for this instance with optimal cost.
(minimum or maximum)

Eg: Given graph G, find biggest clique.

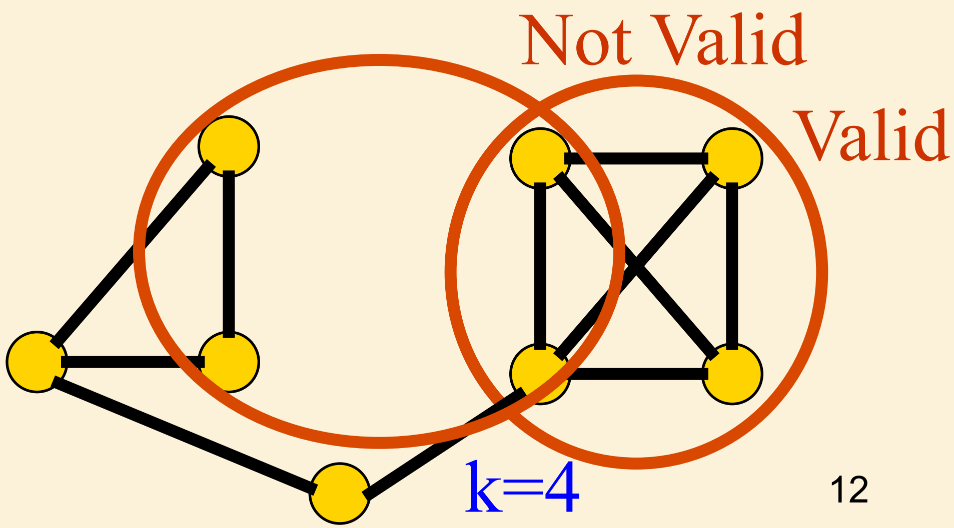
Non-Deterministic Poly-Time Decision Problems (NP)

- An optimization problem
- Each solution is either valid or not (no cost)
- The output is
  - Yes, it has an valid solution.
  - No, it does not
  - the solution is not returned
- Eg: Given graph and integer <G,k>,

  does G have a clique of size k?

Non-Deterministic Poly-Time Decision Problems (NP)

- Key: Given
  - an instance I (= <G,k>)
  - and a solution S (= subset of nodes)
  - there is a poly-time alg Valid(I,S) to test whether or not S is a valid solution for I.
  - Poly-time in |I| not in |S|.



Which are more alike?

Network Flow

Bipartite Matching

Polynomial time algorithm

Similar structure

Graph Colouring

Circuit Satisfiability

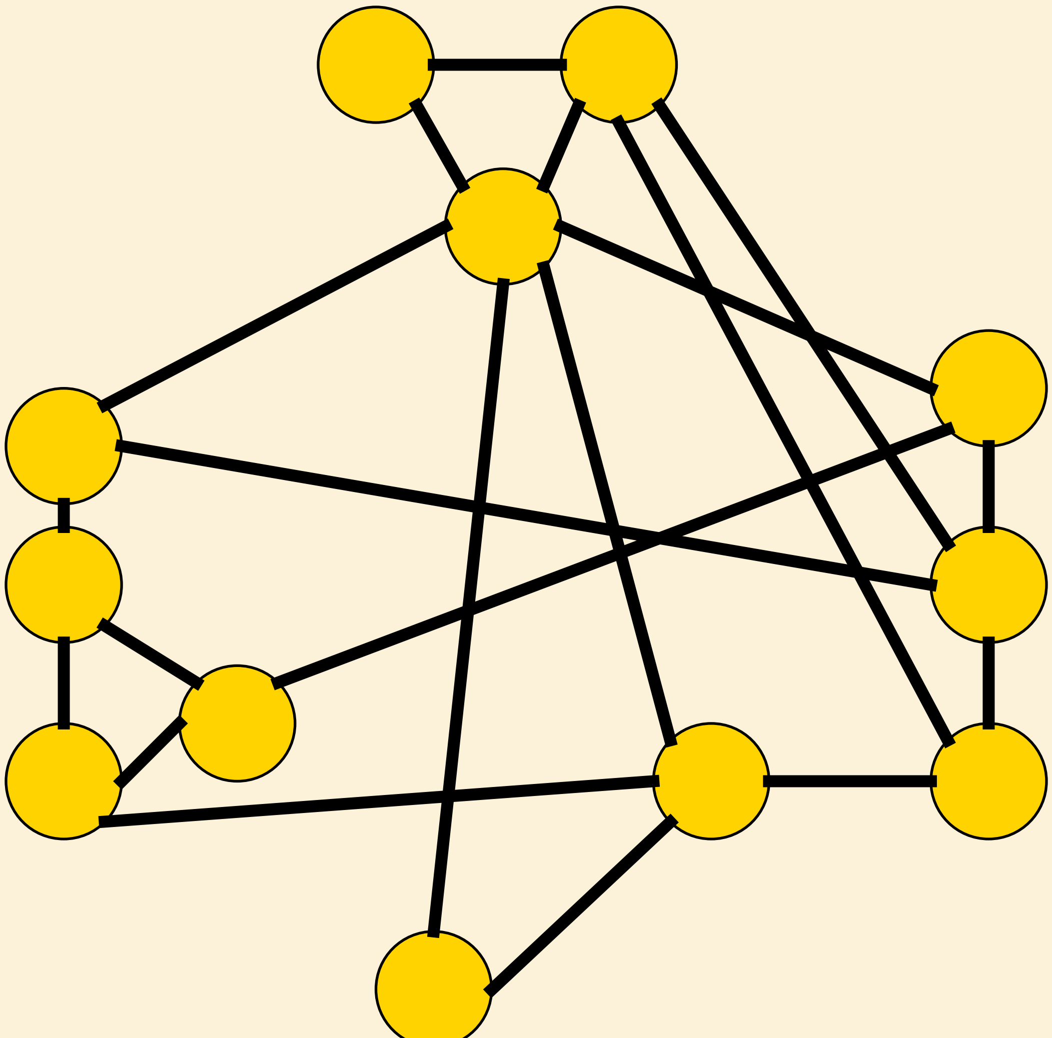
Best known algorithm exponential time

Similar structure

Non-Deterministic Poly Time Complete

Reducibility

A Graph Named “Gadget”



K-COLORING

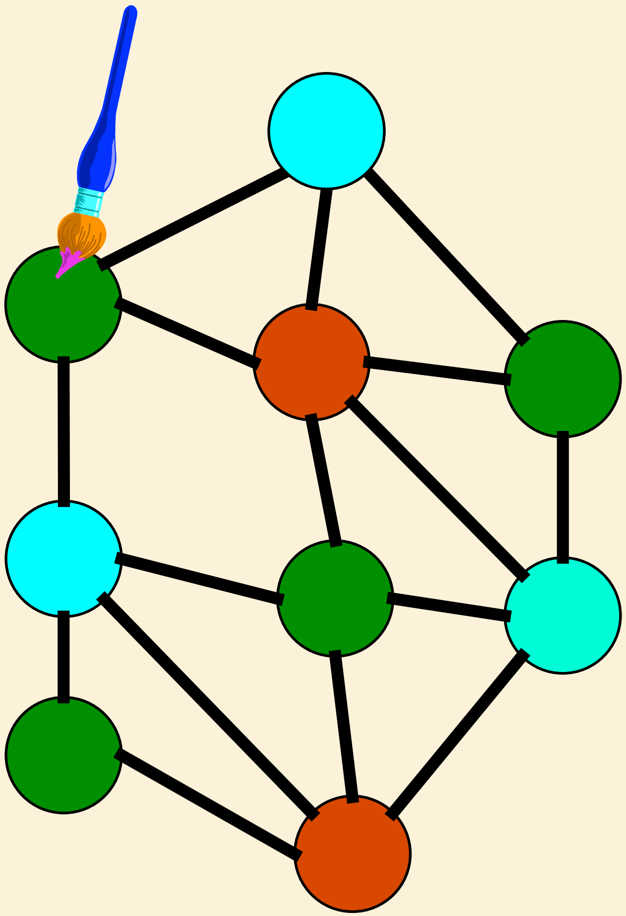
- A k-coloring of a graph is an assignment of one color to each vertex such that:
  - No more than k colors are used
  - No two adjacent vertices receive the same color

- A graph is called k-colorable iff it has a k-coloring

Course Scheduling Problem

Given the courses students want to take and the time slots available, schedule courses to minimize number of conflicts (Avoid scheduling two courses at the same time if a student wants to take both).

K-CRAYOLA Problem:

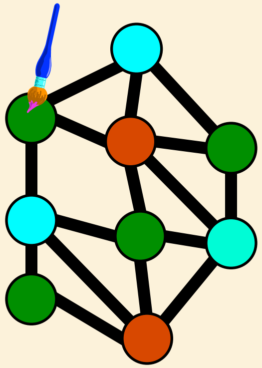


Colour each node.

Nodes with lines between them must have different colours.

- Given a graph G and a k, find a way to colour G with k colours.

Two Different Problems



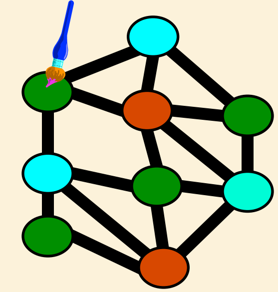
Schedule each course.
Courses that conflict
can't be at same time.

≈

Colour each node.
Nodes with lines
between them must
have different colours.

Two problems that are cosmetically different,
but substantially the same

Problems are the Same!



Schedule each course.
Courses that conflict can't be at same time.

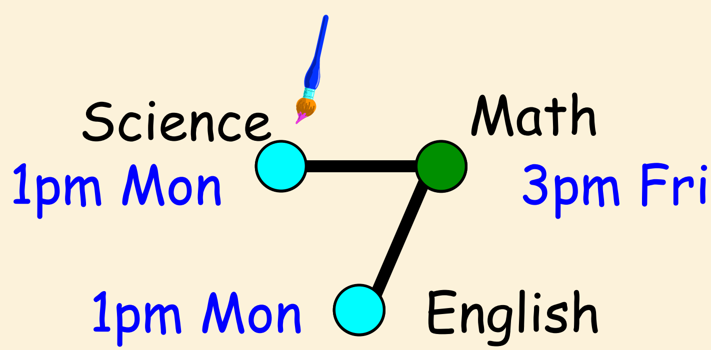
≈

Colour each node.
Nodes with lines between them must have different colours.

course ≈ node

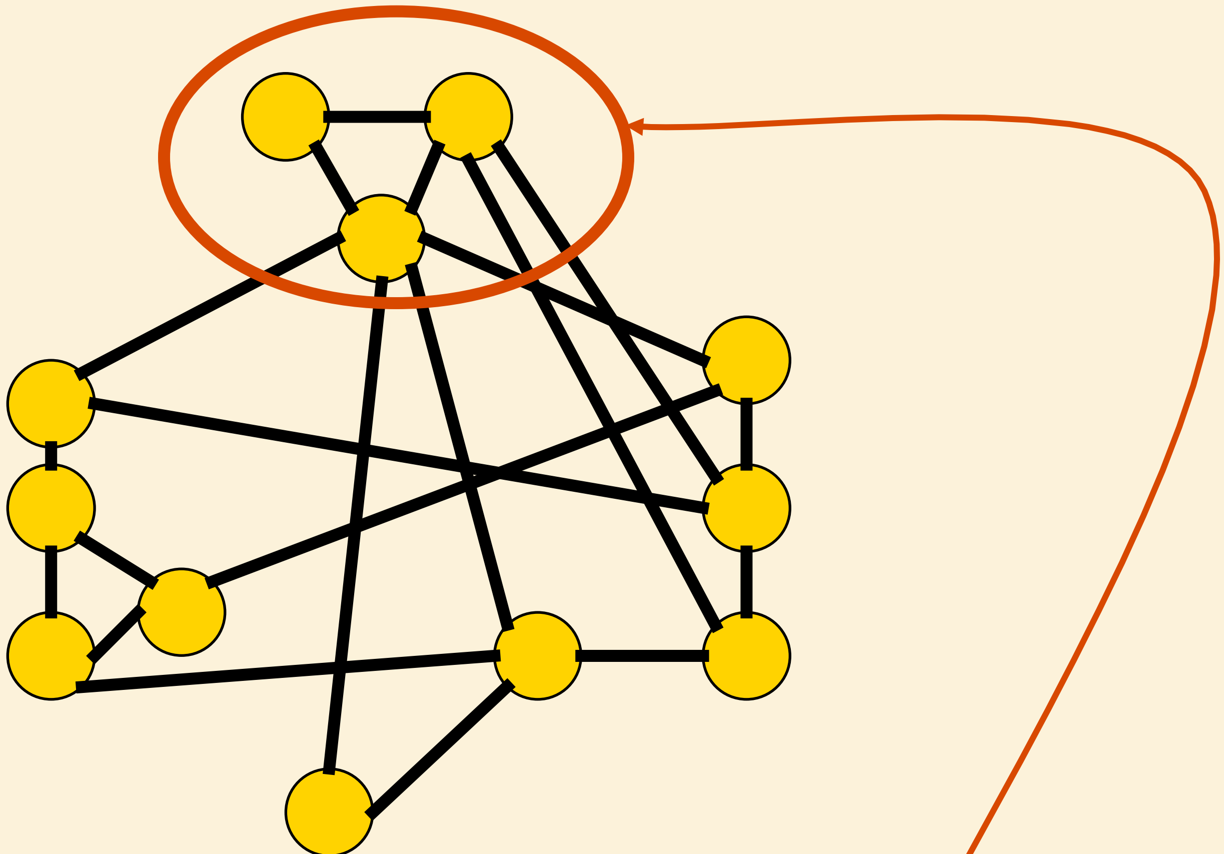
can't be scheduled at same time ≈ line between them

scheduled time ≈ colour



A CRAYOLA Question!

- Is Gadget 2-colorable?



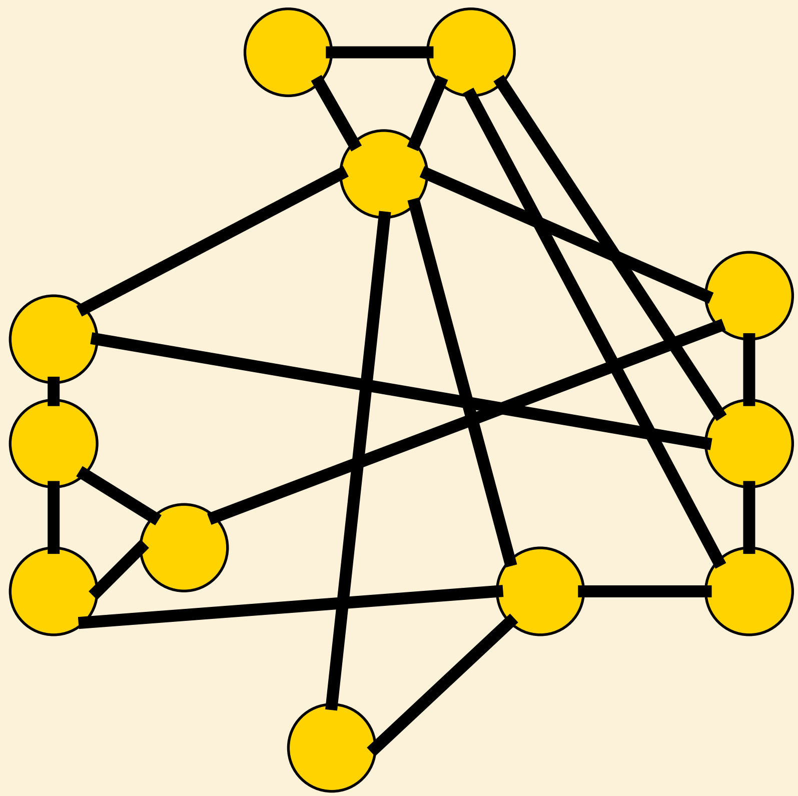
-

No: it contains a triangle!

A CRAYOLA Question!

- Is Gadget 3-colorable?

-



Yes!

2 CRAYOLAS

- Given a graph G, how do we decide if it can be 2-colored?

PERSPIRATION; BRUTE FORCE:
Try out all 2^n ways of 2 coloring G.

INSPIRATION !

A FAST ALGORITHM

GIVEN A GRAPH G.

- COLOR ONE NODE OF EACH CONNECTED COMPONENT OF G BLUE.
- WHILE SOME COLORED NODE "V" HAS SOME UNCOLORED NEIGHBORS DO

  COLOR UNCOLORED NEIGHBORS THE COLOR DIFFERENT FROM THE COLOR OF "V"
- IF ALL EDGES HAVE ENDS OF A DIFFERENT COLOR, OUTPUT "YES, HERE IS A VALID 2-COLORING..."

  OTHERWISE OUTPUT "SORRY IT IS NOT POSSIBLE TO 2-COLOR G"

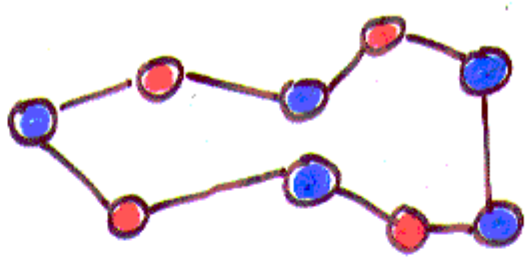
CORRECTNESS OF THE ALGORITHM

A <u>CYCLE</u> IS A SEQUENCE OF VERTICES $V_1, V_2, V_3, \ldots, V_K$ WITH EDGES BETWEEN V_1 AND V_2, V_2 AND V_3, V_3 AND V_4, $\ldots$, V_K AND V_1.

CLAIM: A GRAPH CAN BE 2-COLORED IFF IT CONTAINS NO CYCLE WITH AN ODD NUMBER OF NODES.

ODD CYCLE $\Rightarrow$ NO 2-COLORING ✓

NO ODD CYCLE $\Rightarrow$ ALGORITHM PRODUCES 2-COLORING



PLOT SUMMARY

WE SEEK AN OBJECT
(2-COLORING OF G)

FROM AMONG A HUGE SPACE OF POSSIBILITIES
(2^n ASSIGNMENTS OF 2 COLORS TO n VERTICES OF G)

PERSPIRATION, I.E., BRUTE FORCE SEARCH TAKES TOO LONG
($\Omega(2^n)$ TIME)

SO WE USE INSPIRATION INSTEAD!
(OUR FAST ALGORITHM FINDS A 2 COLORING IN TIME LINEAR IN THE NUMBER OF EDGES + NODES)

3 CRAYOLAS

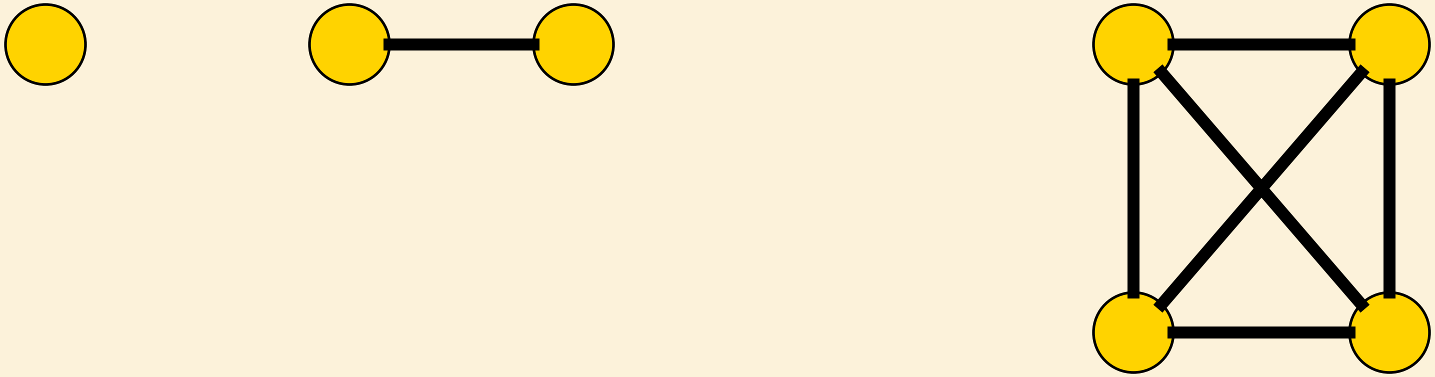
- Given a graph G, what is a fast algorithm to decide if it can be 3-colored?

? ? ? ? ? ? ? ?

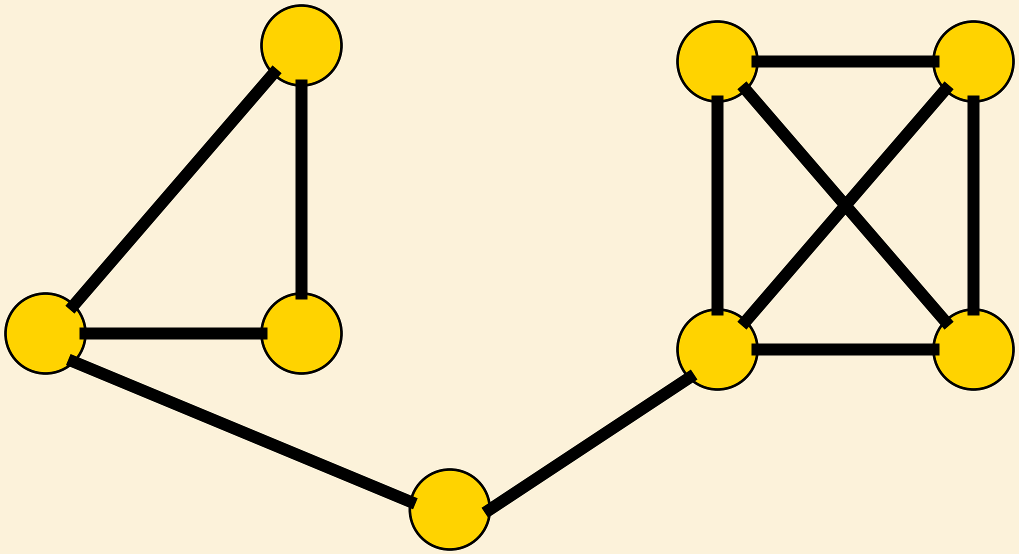
Let’s consider a completely different problem.

k-CLIQUES

- A k-clique is a set of k nodes with all k(k-1)/2 possible edges between them.



This graph contains a 4-clique



Given an n-node graph G and a number k, how can you decide if G contains a k-clique?

- PERSPIRATION: Try out all n-choose-k possible locations for the k clique

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \text{ possibilities}$$

e.g., $k = 3 \rightarrow \Theta(n^3)$

In general, $\Theta(n^k)$

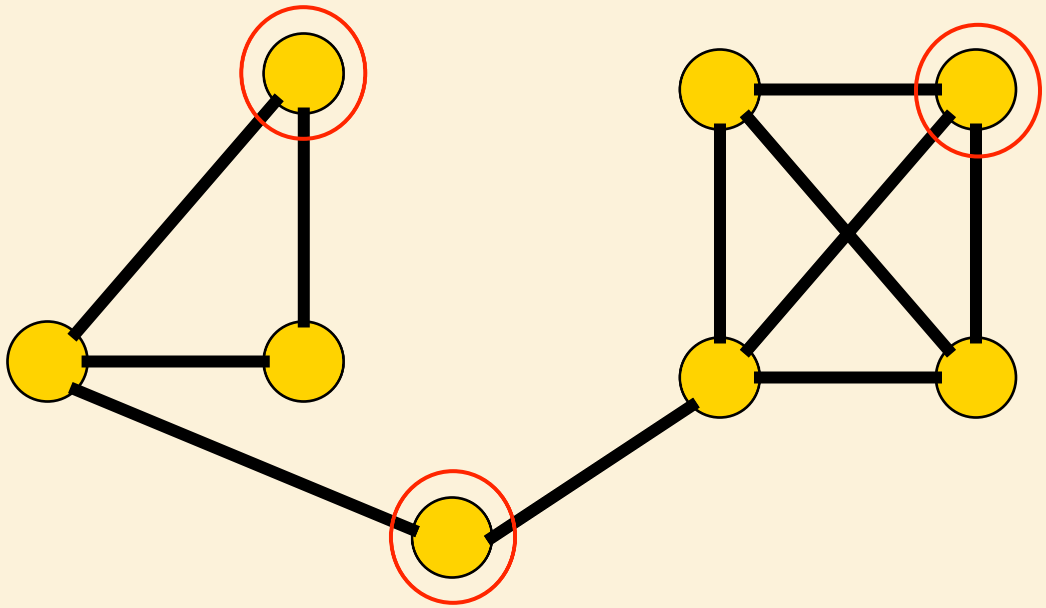
- INSPIRATION:

? ? ? ? ? ? ? ?

OK, how about a slightly different problem?

INDEPENDENT SET

- An **independent set** is a set of vertices with no edges between them.



This graph contains an independent set of size 3.

Given an n-node graph G and a number k, how can you decide if G contains an independent set of size k?

- PERSPIRATION: Try out all n-choose-k possible locations for independent set

- INSPIRATION:

? ? ? ? ? ? ? ?

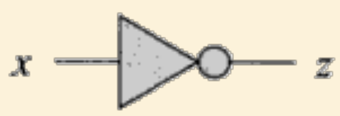
One more completely different problem

Combinational Circuits

- AND, OR, NOT, gates wired together with no feedback allowed (acyclic).

Logic Gates

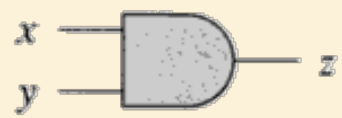
Not



x	$\neg x$
0	1
1	0

(a)

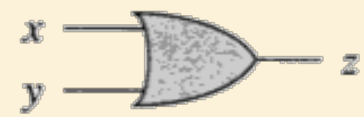
And



x	y	$x \wedge y$
0	0	0
0	1	0
1	0	0
1	1	1

(b)

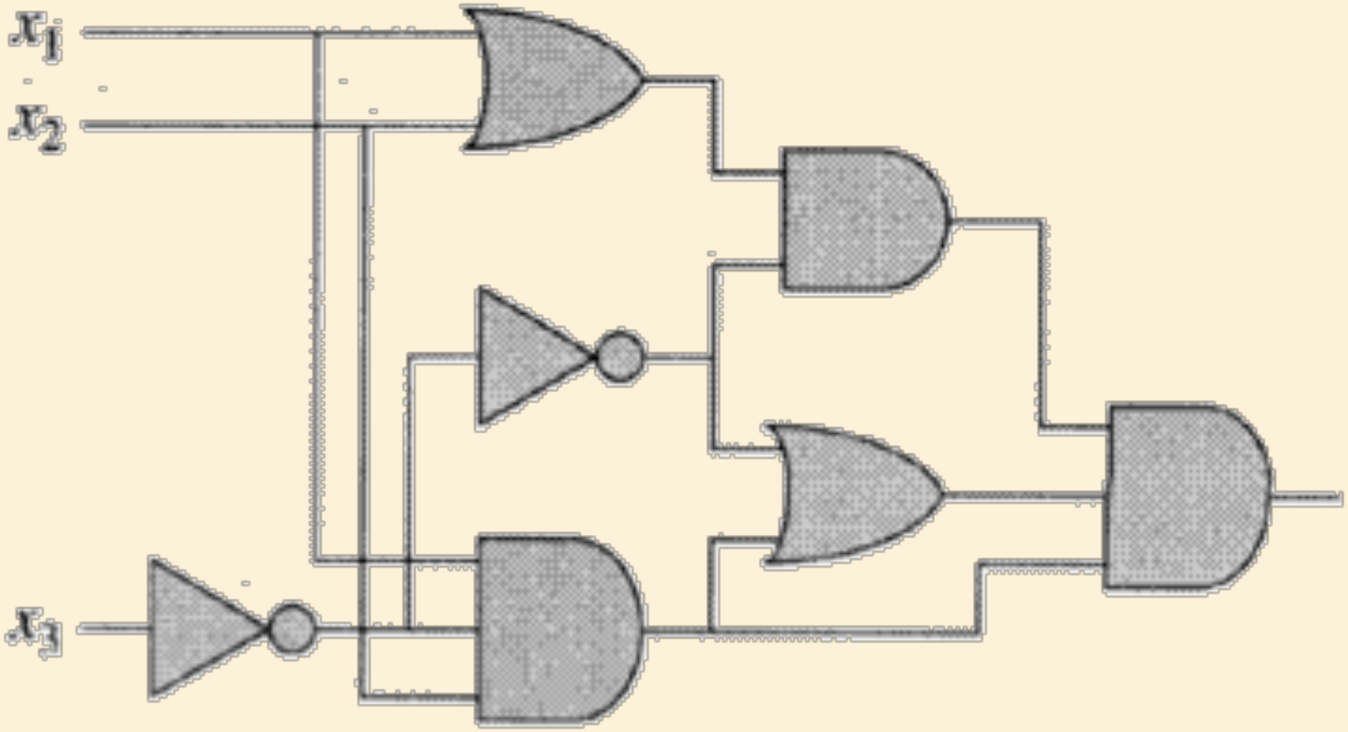
Or



x	y	$x \vee y$
0	0	0
0	1	1
1	0	1
1	1	1

(c)

Example Circuit



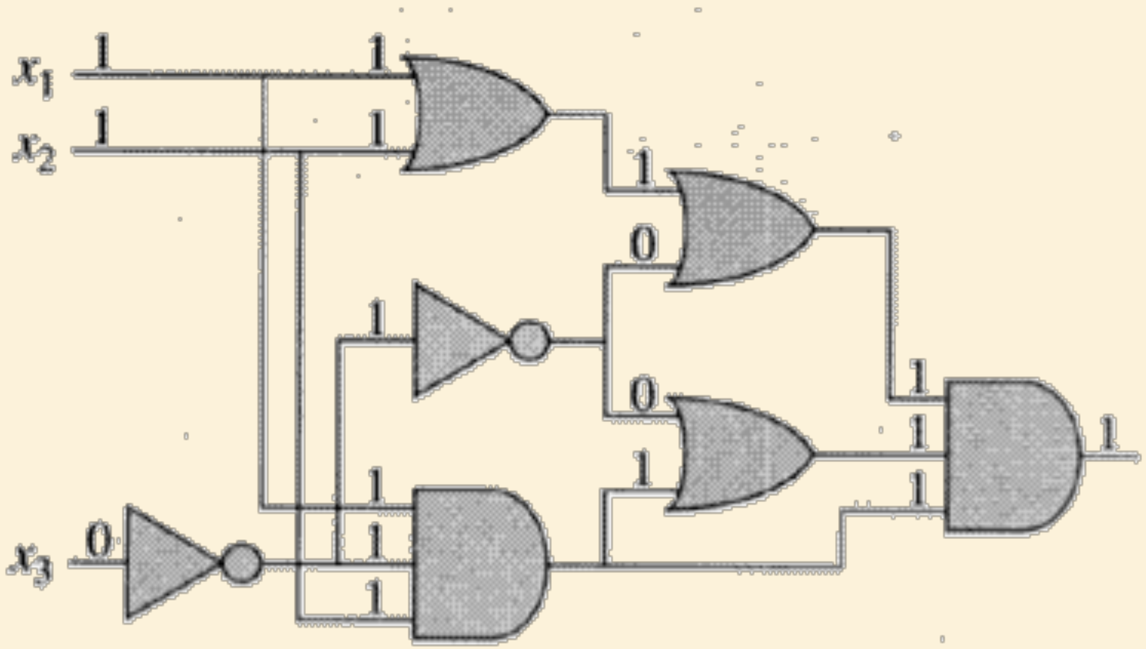
CIRCUIT-SATISFIABILITY (decision version)

- Given a circuit with n-inputs and one output, is there a way to assign 0-1 values to the input wires so that the output value is 1 (true)?

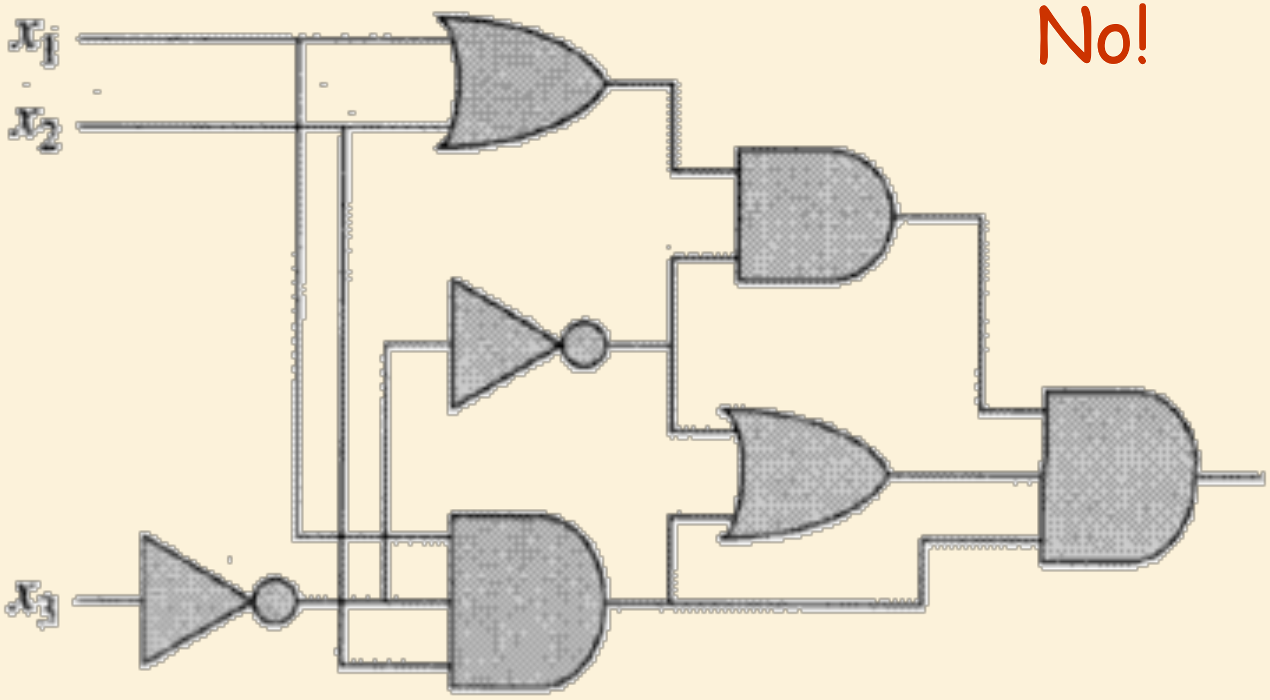
CIRCUIT-SATISFIABILITY (search version)

- Given a circuit with n-inputs and one output, find an assignment of 0-1 values to the input wires so that the output value is 1 (true), or determine that no such assignment exists.

Satisfiable Circuit Example



Satisfiable?



No!

Given a circuit, is it satisfiable?

- PERSPIRATION: Try out all 2^n assignments

- INSPIRATION:

? ? ? ? ? ? ? ?

We have seen 4 problems: coloring, clique, independent set, and circuit SAT.

They all have a common story: A large space of possibilities only a tiny fraction of which satisfy the constraints. Brute force takes too long, and ***no feasible algorithm is known.***

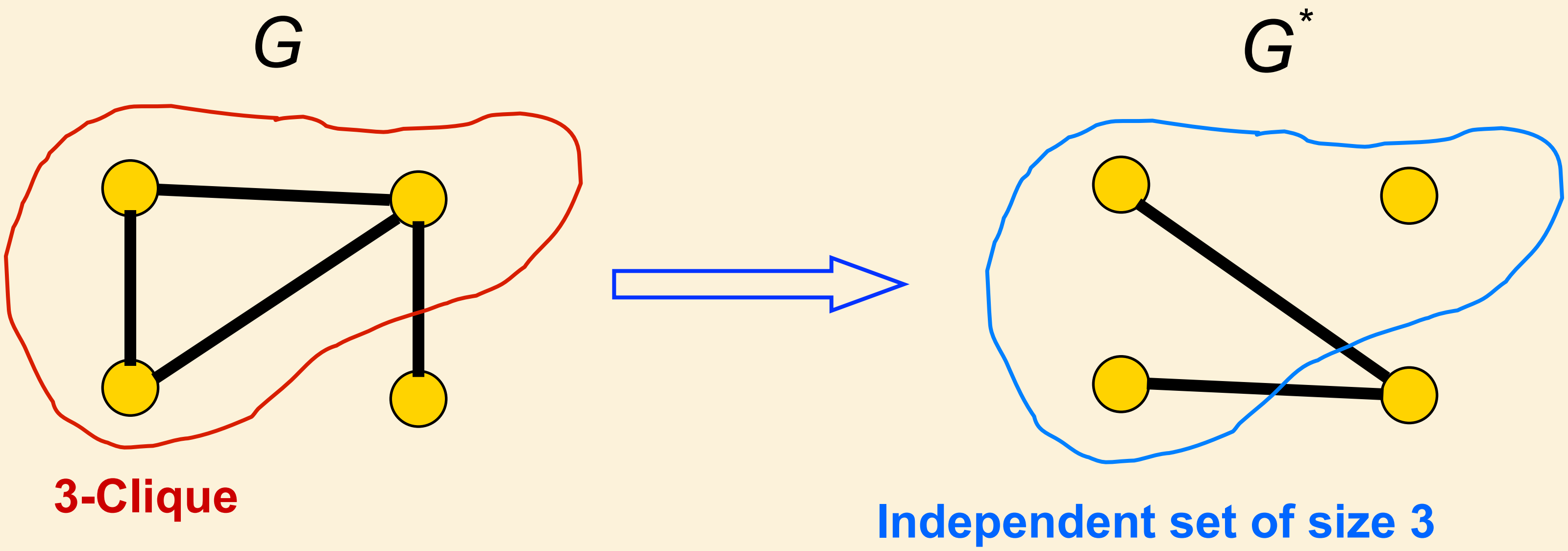
CLIQUE / INDEPENDENT SET

- Two problems that are cosmetically different, but substantially the same

Complement Of G

- Given a graph G, let G^*, the complement of G, be the graph obtained by the rule that two nodes in G^* are connected if and only if the corresponding nodes of G are not connected

Example



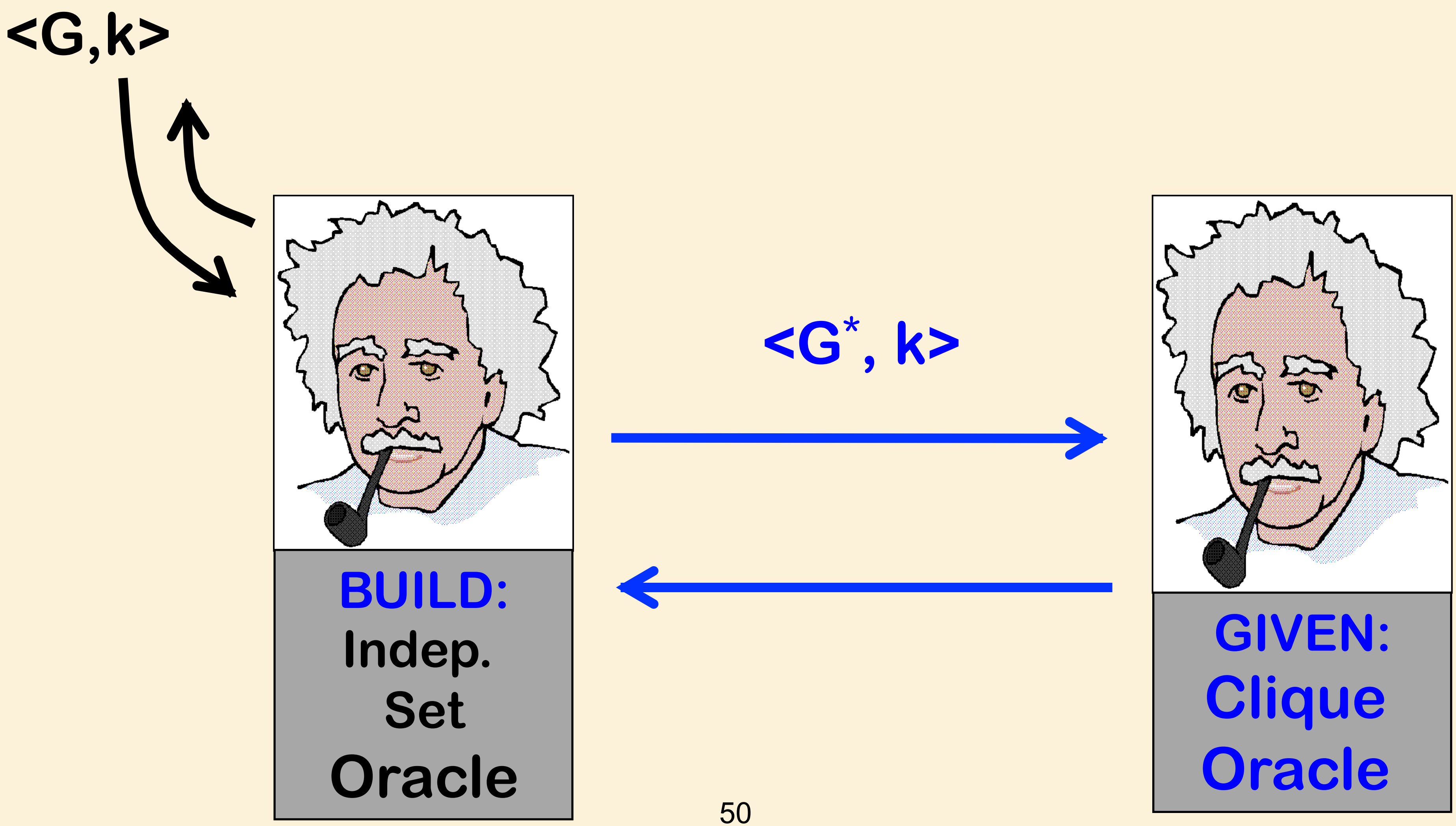
Reduction

- Suppose you have a method for solving the k-clique problem.
- How could it be used to solve the independent set problem?

Or what if you have an Oracle?

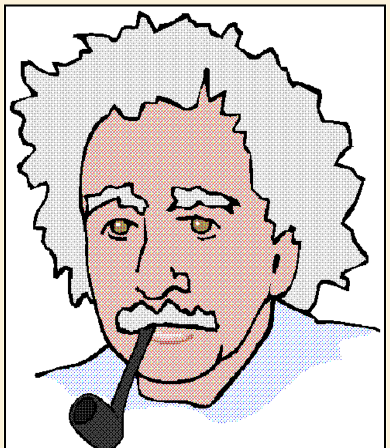
- or·a·cle
- Pronunciation: 'or-&-k&l, 'är-
- Function: noun
- Etymology: Middle English, from Middle French, from Latin oraculum, from orare to speak
- 1 a : a person (as a priestess of ancient Greece) through whom a deity is believed to speak
- 2 a : a person giving wise or authoritative decisions or opinions

Let G be an n-node graph.



Let G be an n-node graph.

<G,k>



BUILD:
Clique Oracle

<G*, k>

GIVEN:
Indep. Set Oracle

Thus, we can quickly reduce clique problem to an independent set problem and vice versa.

There is a fast method for one if and only if there is a fast method for the other.

Given an oracle for circuit SAT, how can you quickly solve 3-colorability?

$V_n(X,Y)$

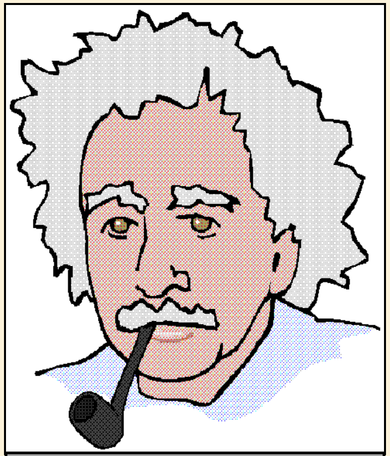
- Let V_n be a circuit that takes an n-node graph X and an assignment Y of colors to these nodes, and **verifies** that Y is a valid 3-colouring of X. i.e., $V_n(X,Y)=1$ iff Y is a 3-colouring of X.

- X is expressed as an n-choose-2 bit sequence. Y is expressed as a 2n bit sequence.

- Given n, we can construct V_n in time $O(n^2)$.

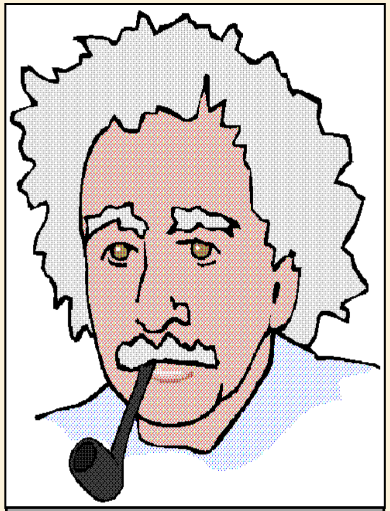
Let G be an n-node graph.

G



BUILD:
3-
color
Oracle

$V_n(G,Y)$



GIVEN:
SAT
Oracle

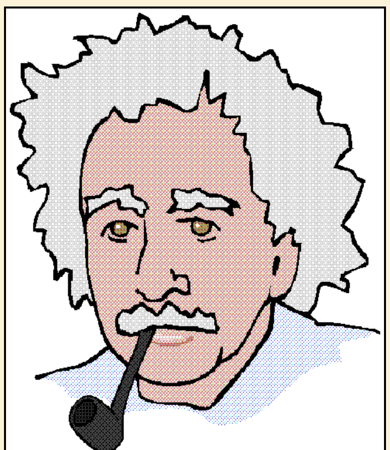
Given an oracle for circuit SAT, how can you quickly solve k-clique?

$V_{n,k}(X,Y)$

- Let V_n be a circuit that takes an n-node graph X and a subset of nodes Y, and **verifies** that Y is a k-clique X. I.e., $V_n(X,Y)=1$ iff Y is a k-clique of X.
- X is expressed as an n choose 2 bit sequence. Y is expressed as an n bit sequence.
- Given n, we can construct $V_{n,k}$ in time $O(n^2)$.

Let G be an n-node graph.

<G,k>



BUILD:
Clique
Oracle

$V_{n,k}(G,Y)$

GIVEN:
SAT
Oracle

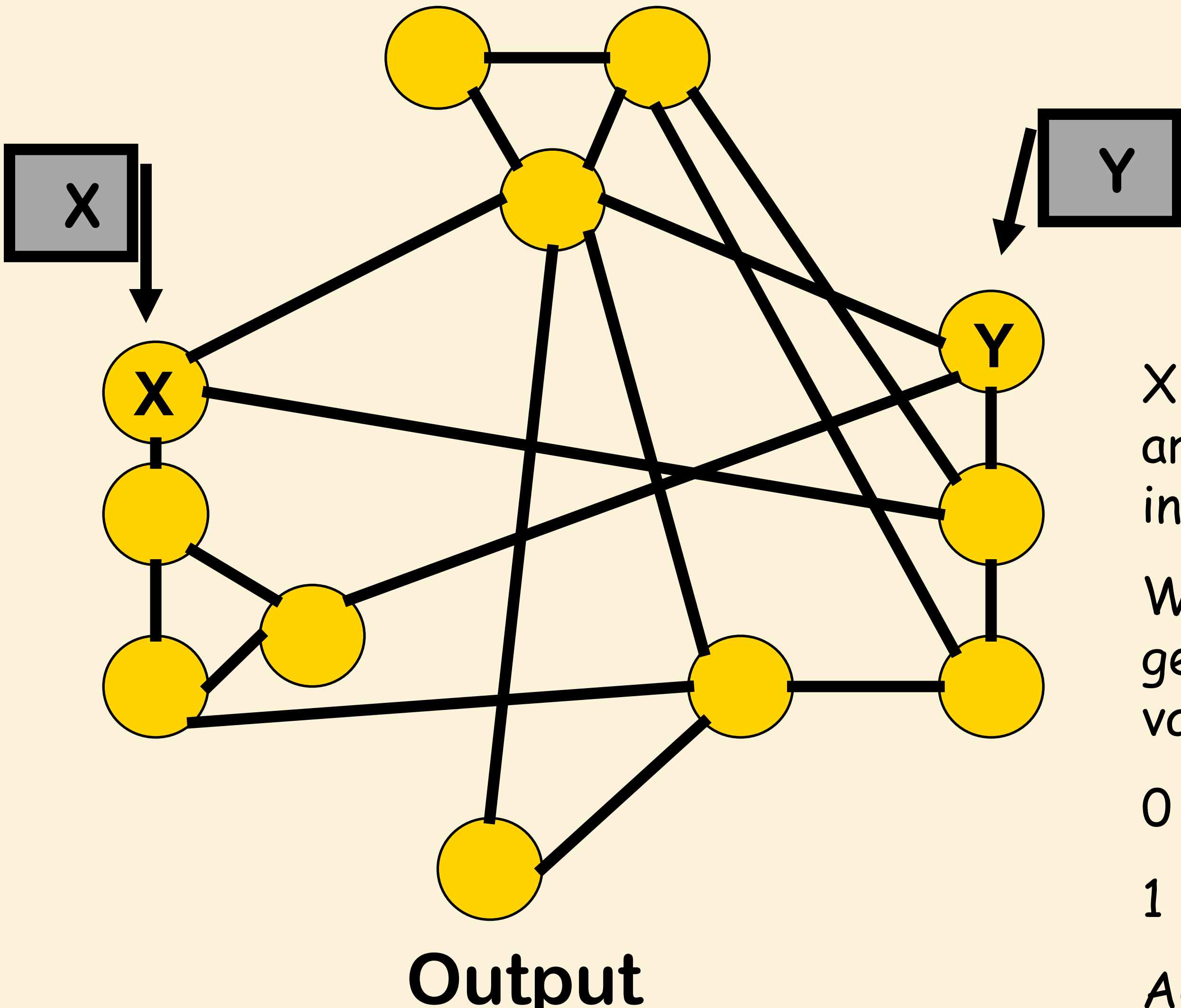
Given an oracle for 3-colorability, how can you quickly solve circuit SAT?

Reducing Circuit-SAT to 3-Colouring

- Goal: map circuit to graph that is 3-colourable only if circuit is satisfiable.

- How do we represent a logic gate as a 3-colouring problem?

Example



X and Y and Output are boolean variables in circuit.

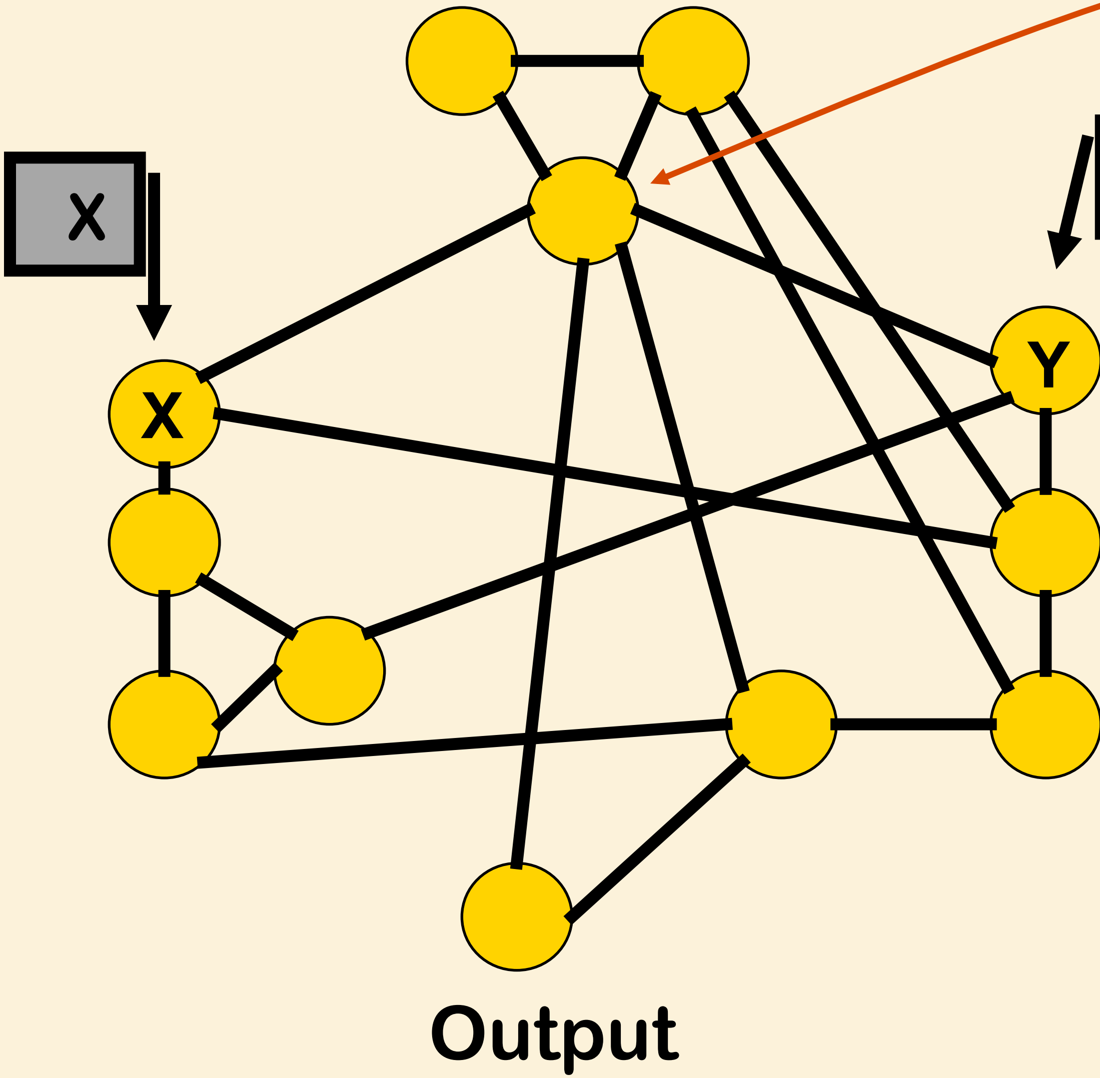
Without loss of generality, map truth values to colours, e.g.

0 $\leftarrow\rightarrow$ red

1 $\leftarrow\rightarrow$ green

Add base colour for encoding purposes, e.g. blue.

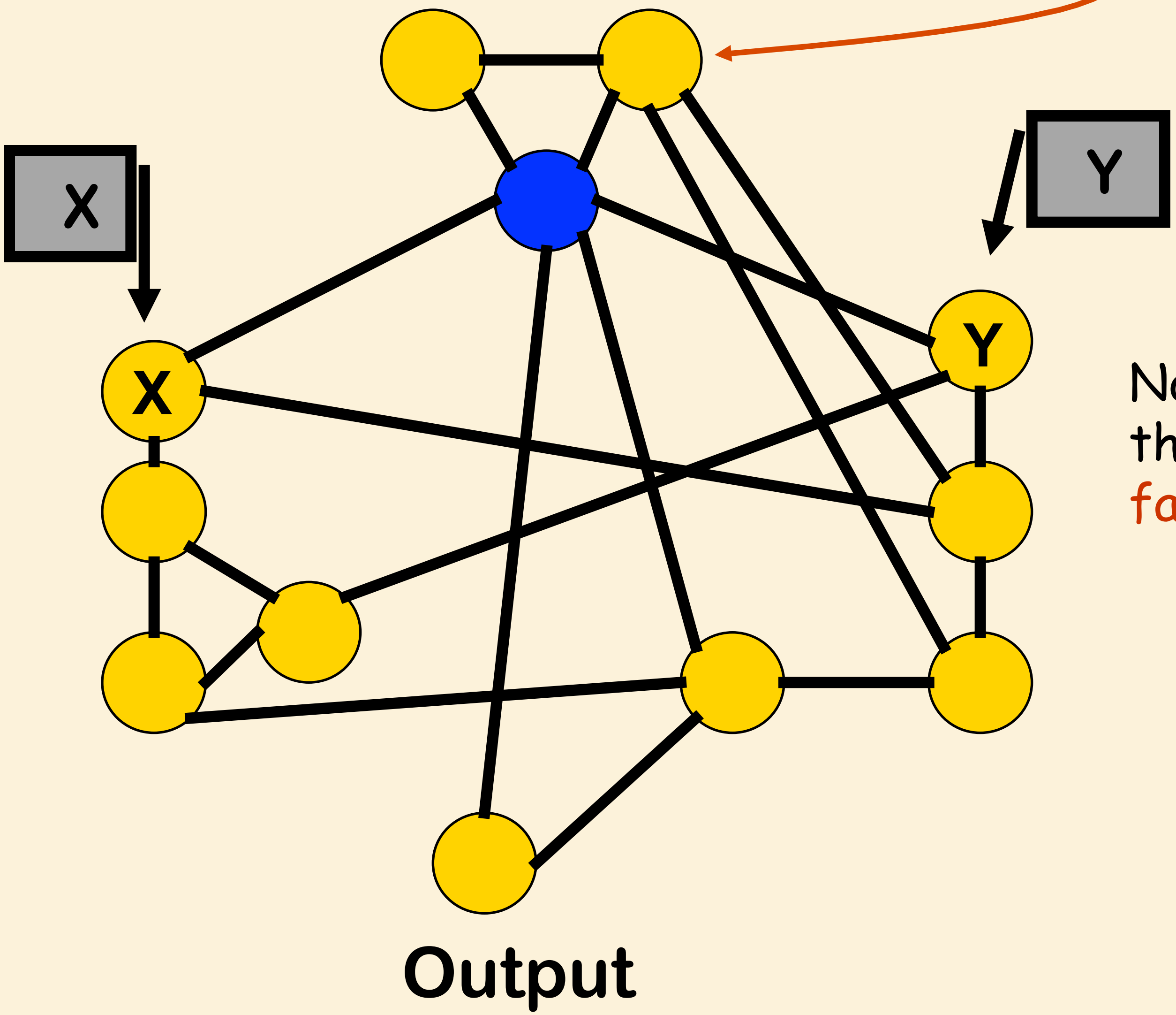
Example



Note that in a valid 3-colouring, this node cannot have the same colour as X, Y or Output.

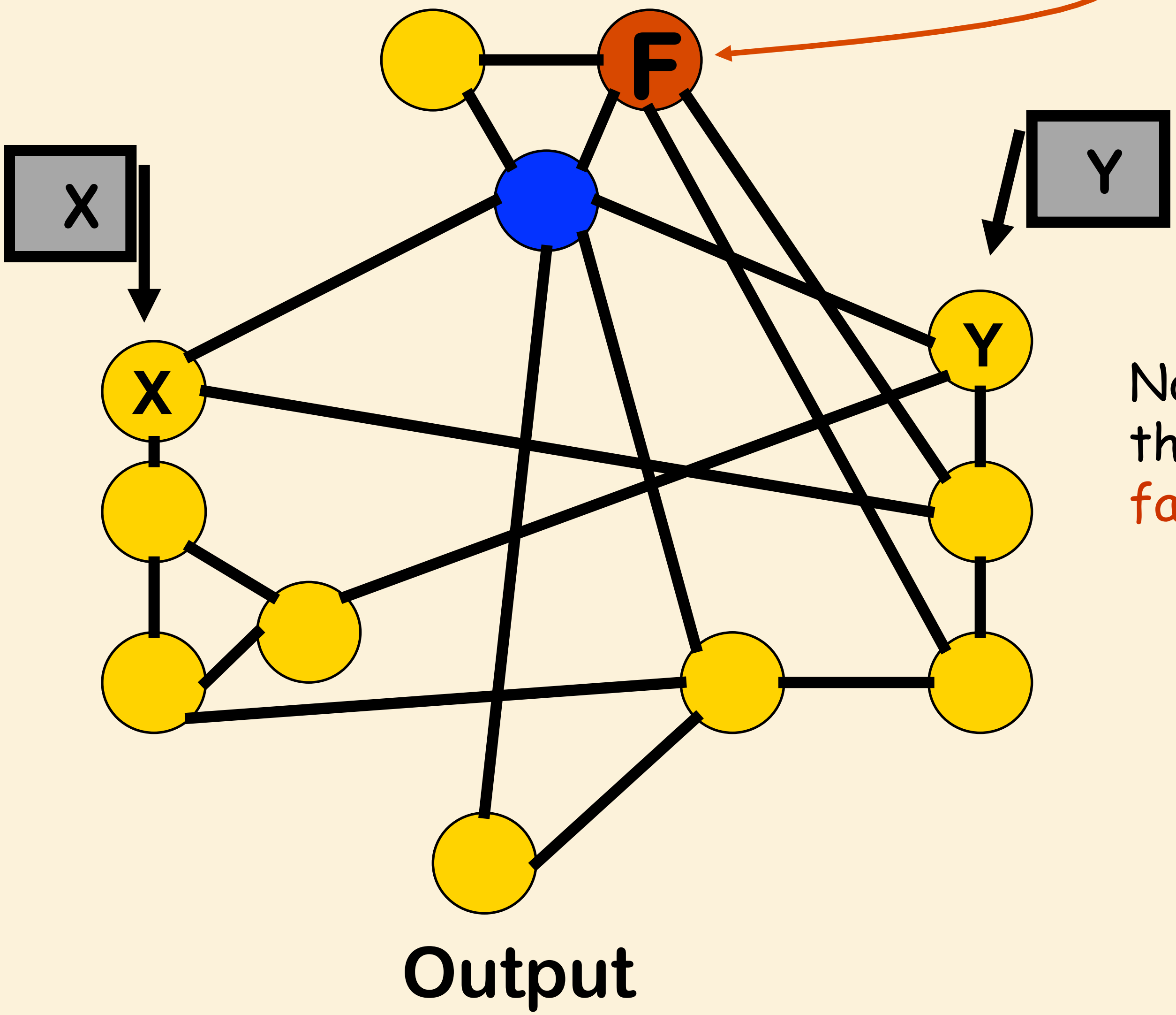
Thus, without loss of generality, we can assign it the base colour, blue.

Example



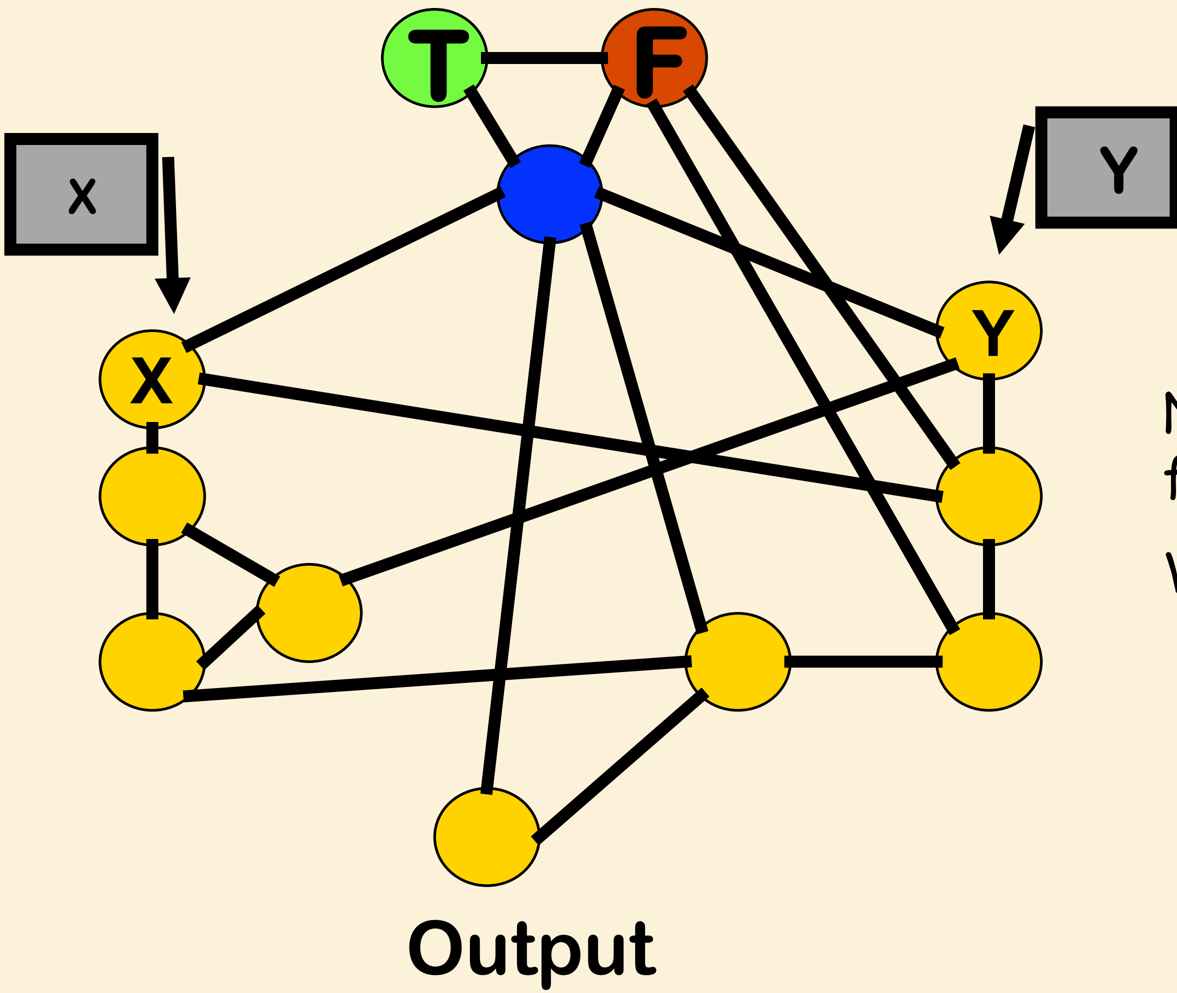
Now suppose we fix this node to represent false.

Example



Now suppose we fix this node to represent false.

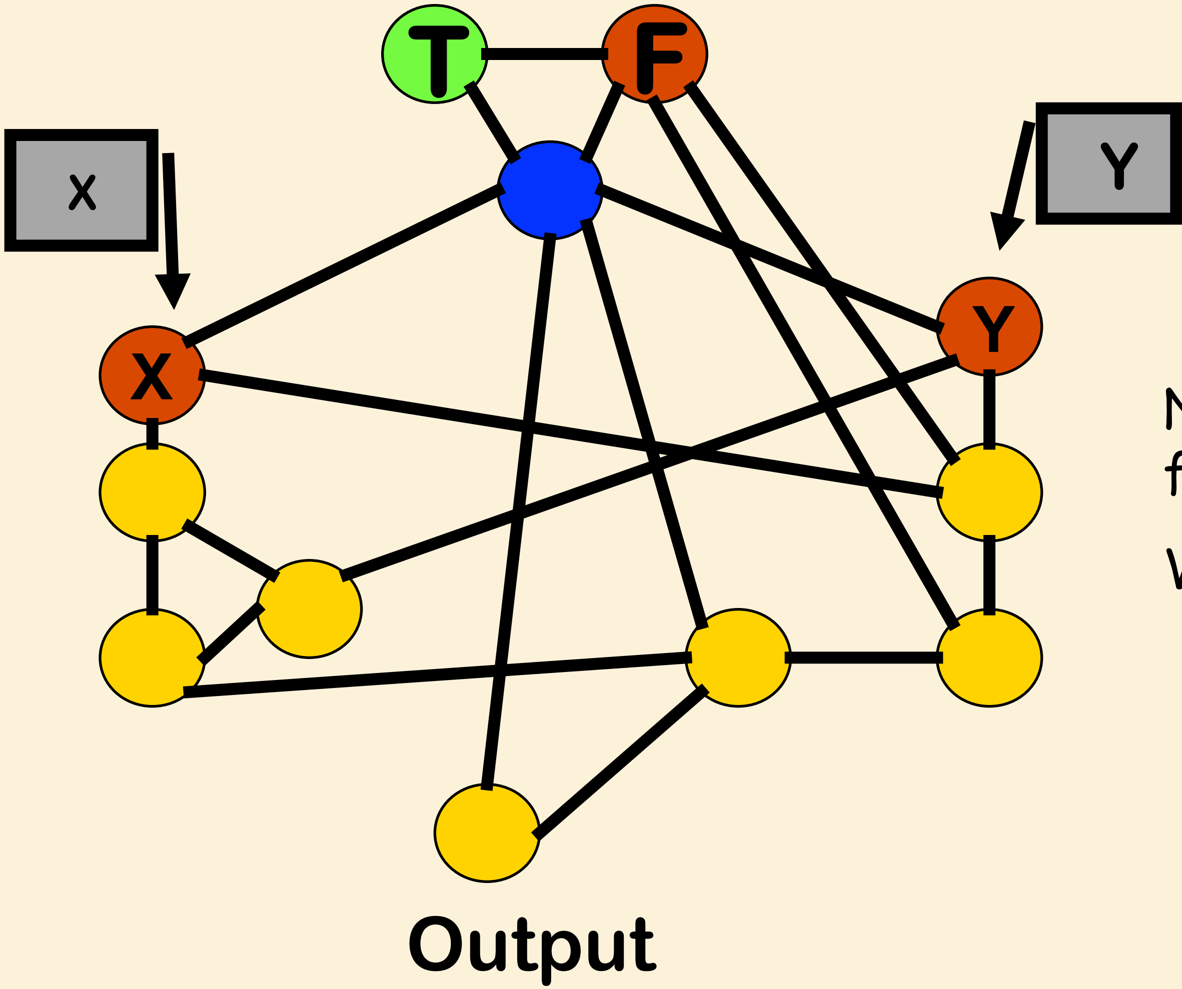
Example



Now build a truth table for (X, Y, Output).

What if X=Y=0?

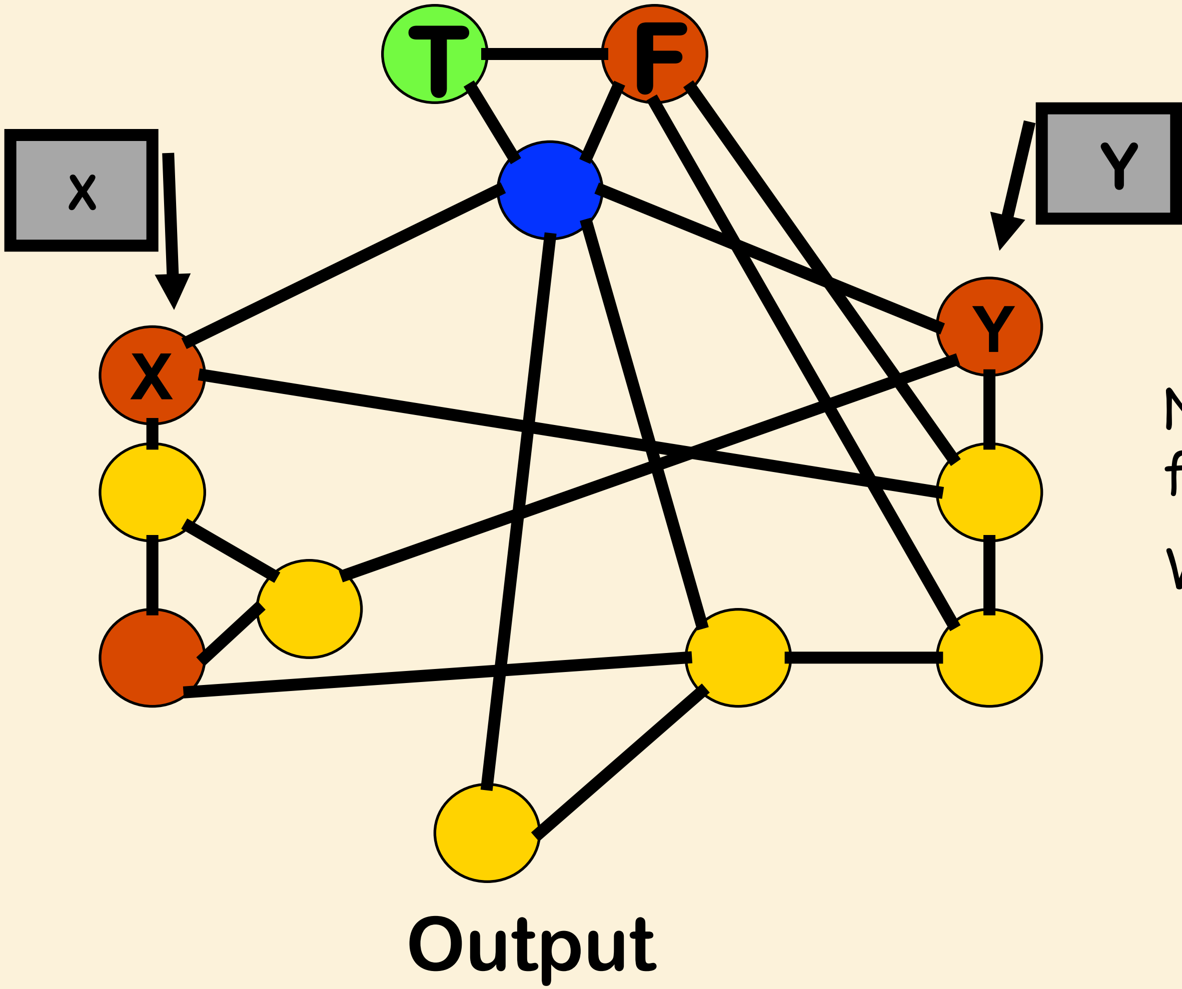
Example



Now build a truth table for (X, Y, Output).

What if X=Y=0?

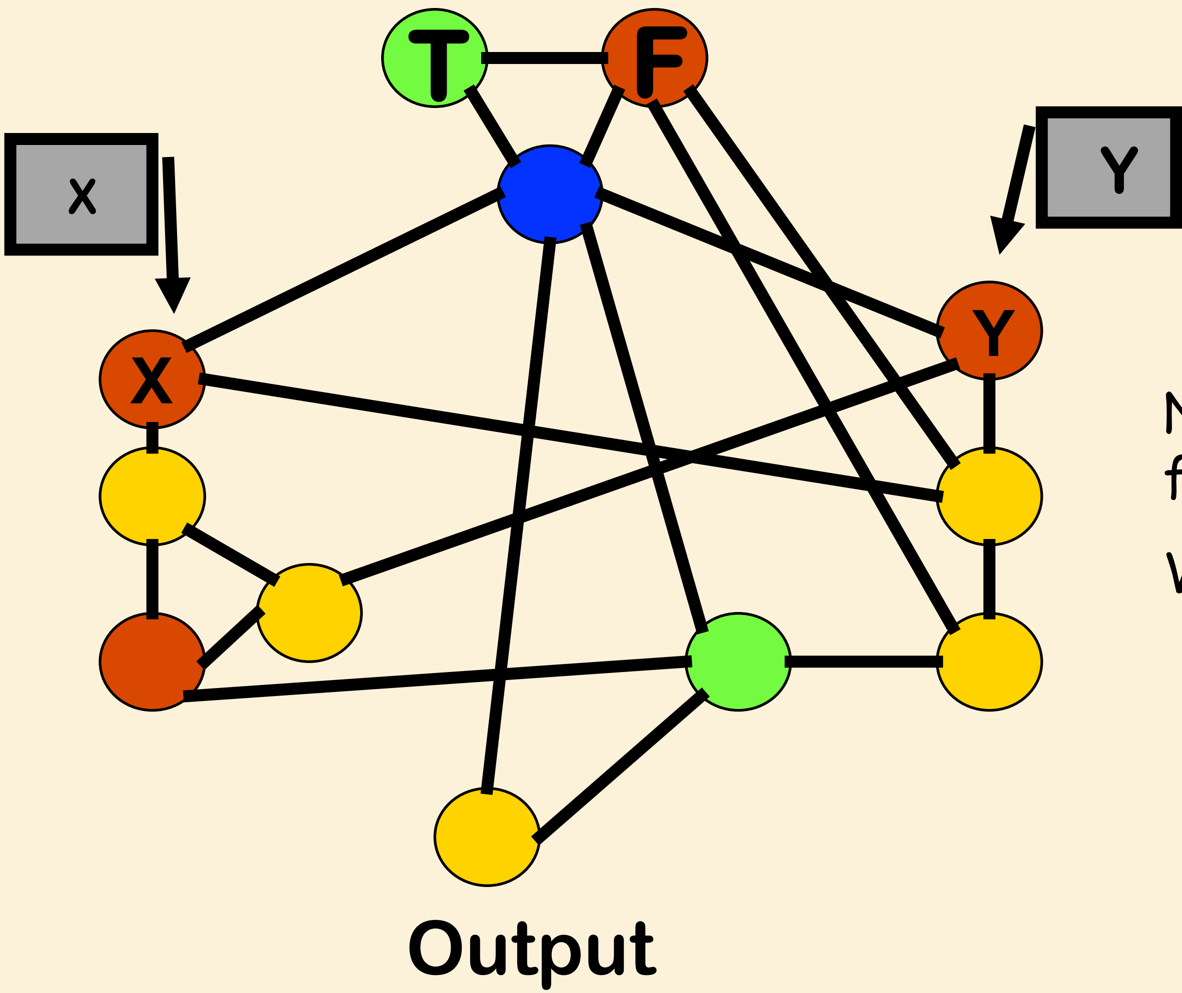
Example



Now build a truth table for (X, Y, Output).

What if X=Y=0?

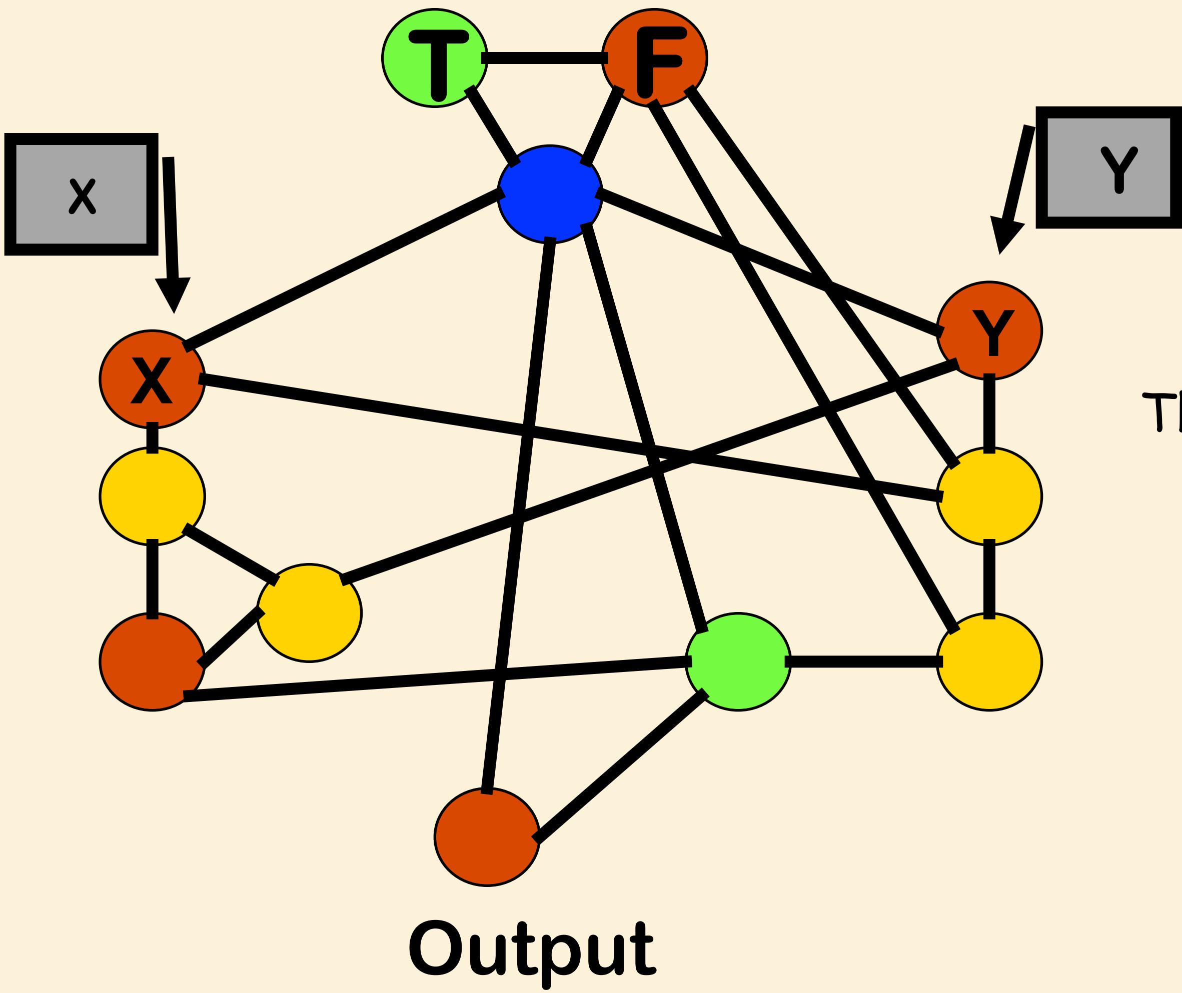
Example



Now build a truth table for (X, Y, Output).

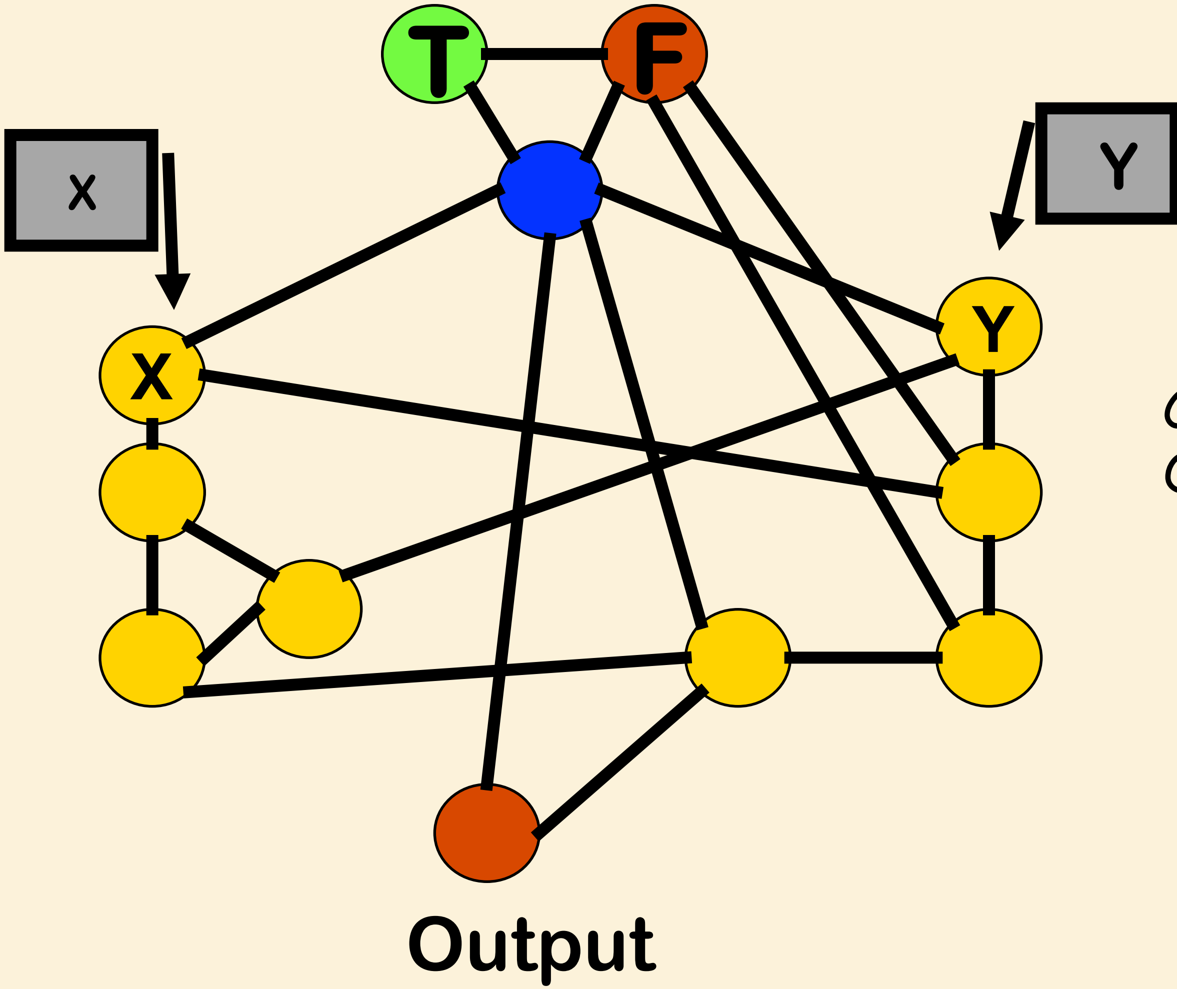
What if X=Y=0?

Example



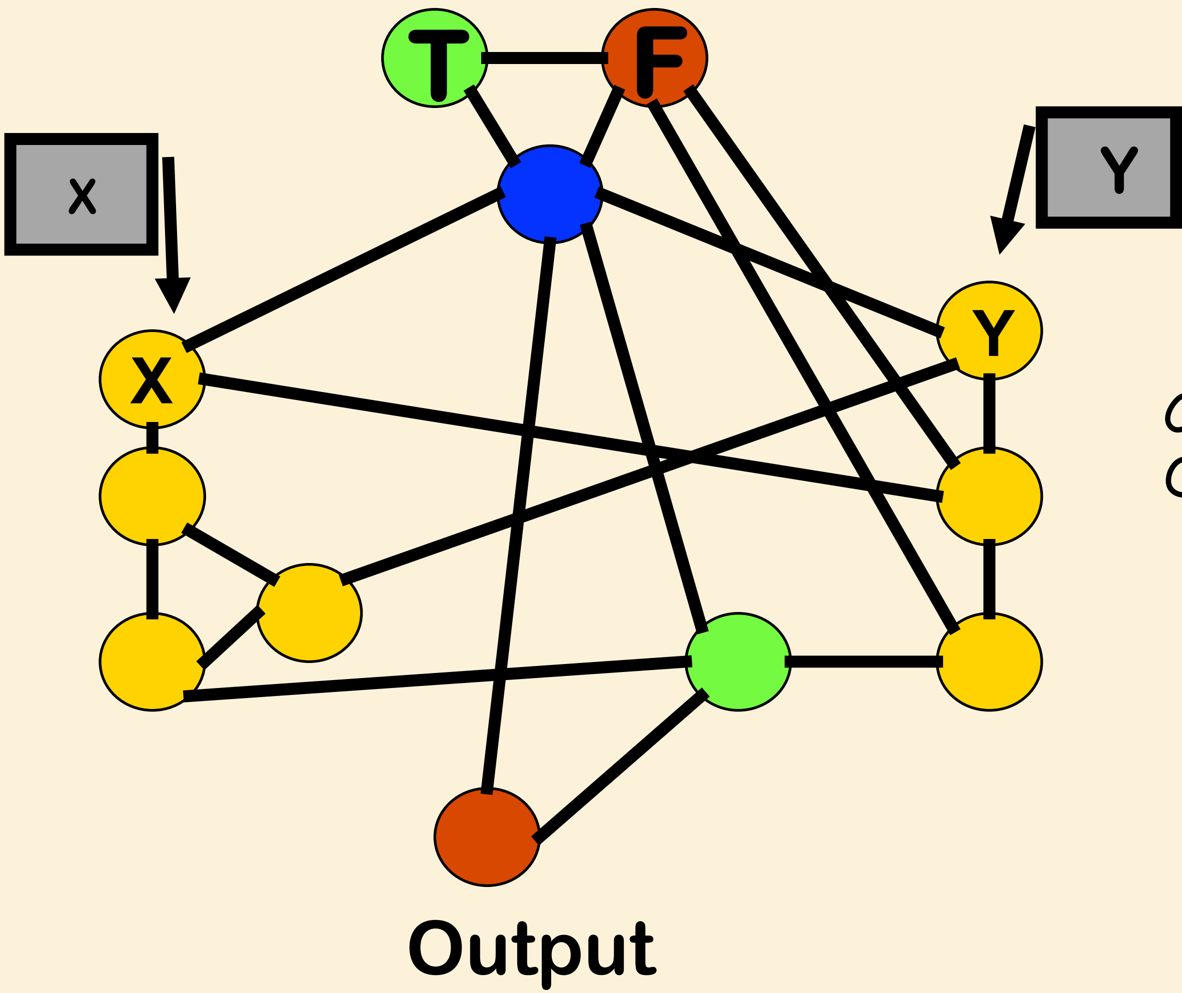
Thus (X,Y)=0→Output=0

Example



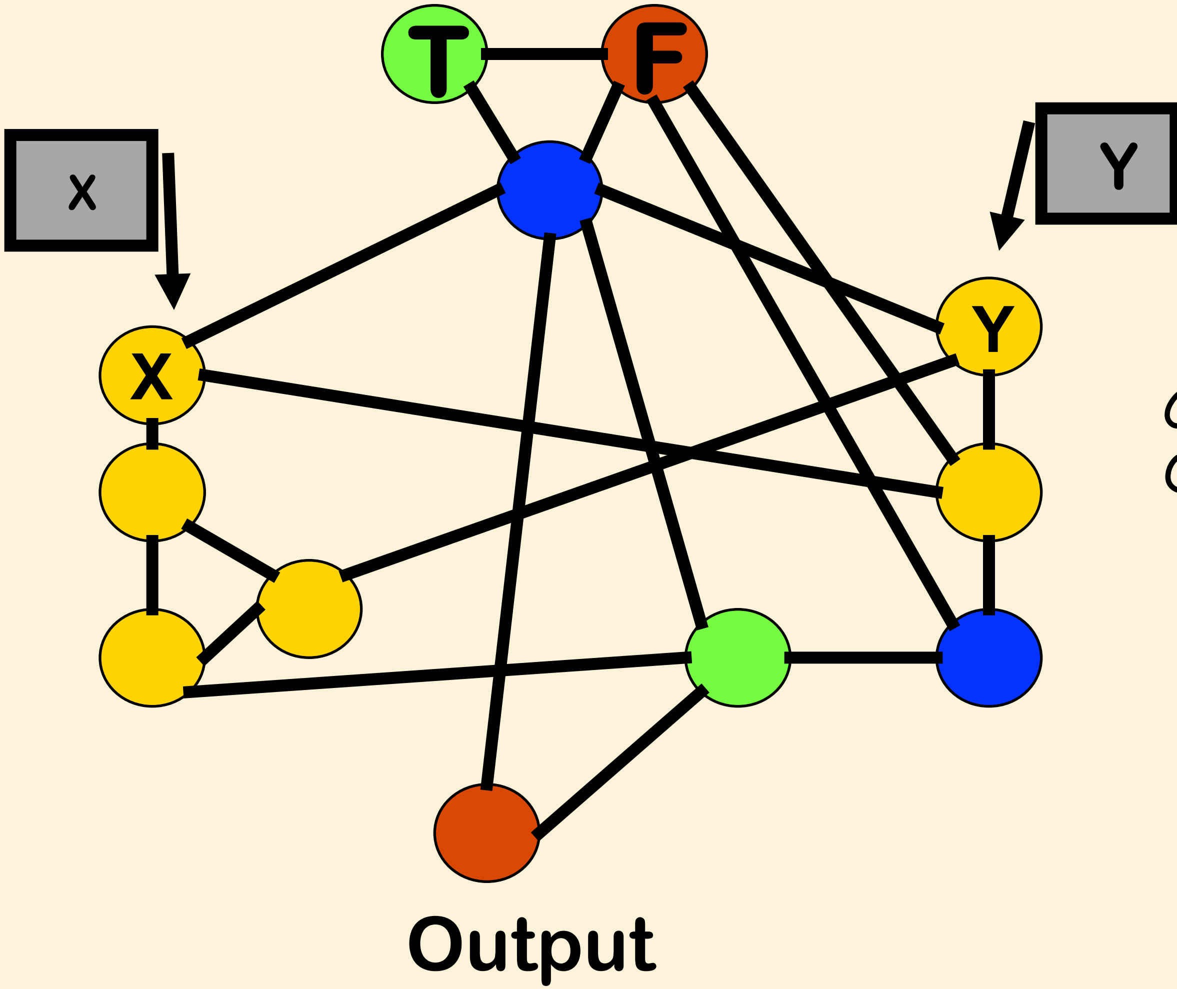
Conversely, what if Output=0?

Example



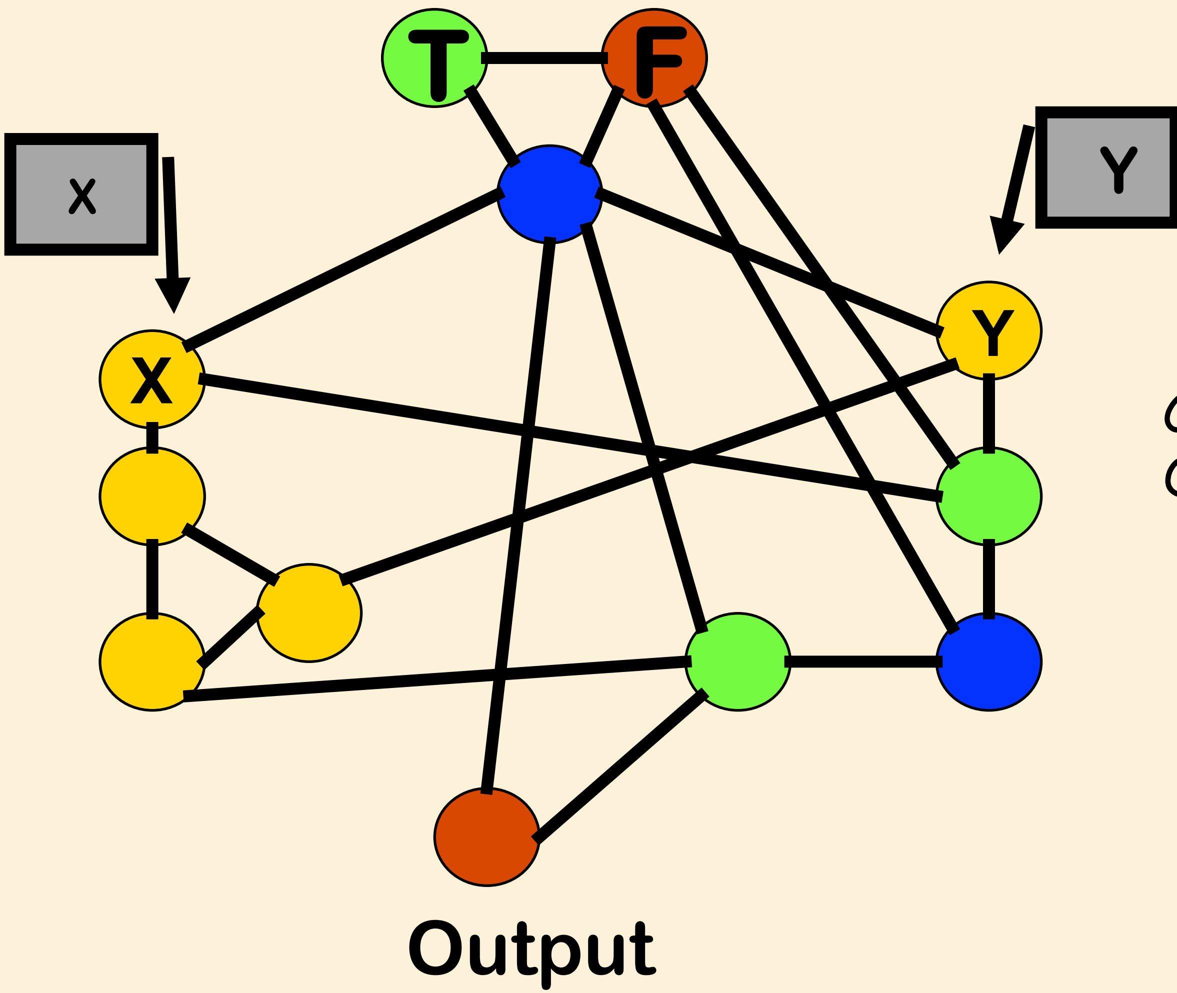
Conversely, what if Output=0?

Example



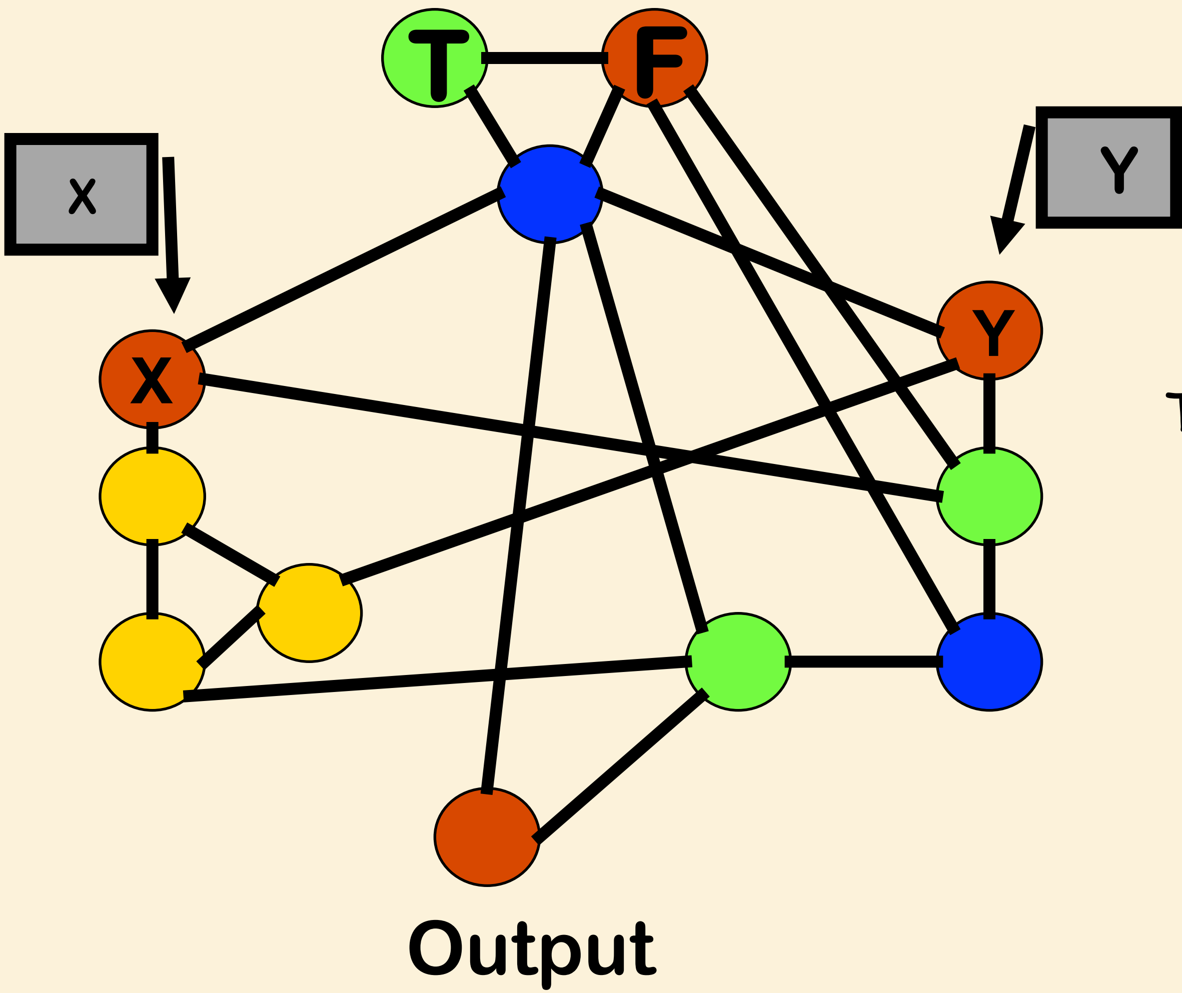
Conversely, what if Output=0?

Example



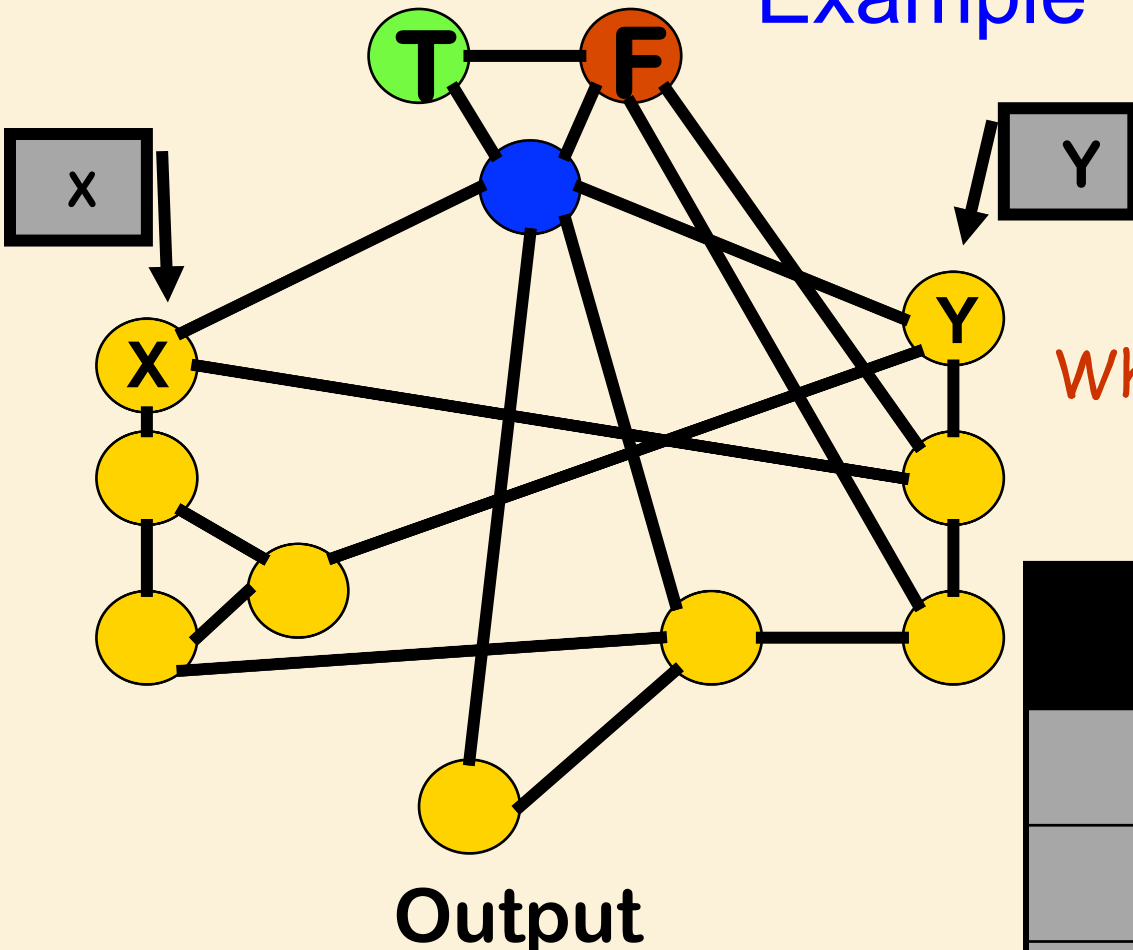
Conversely, what if
Output=0?

Example



Thus Output=0→X=Y=0.

Example

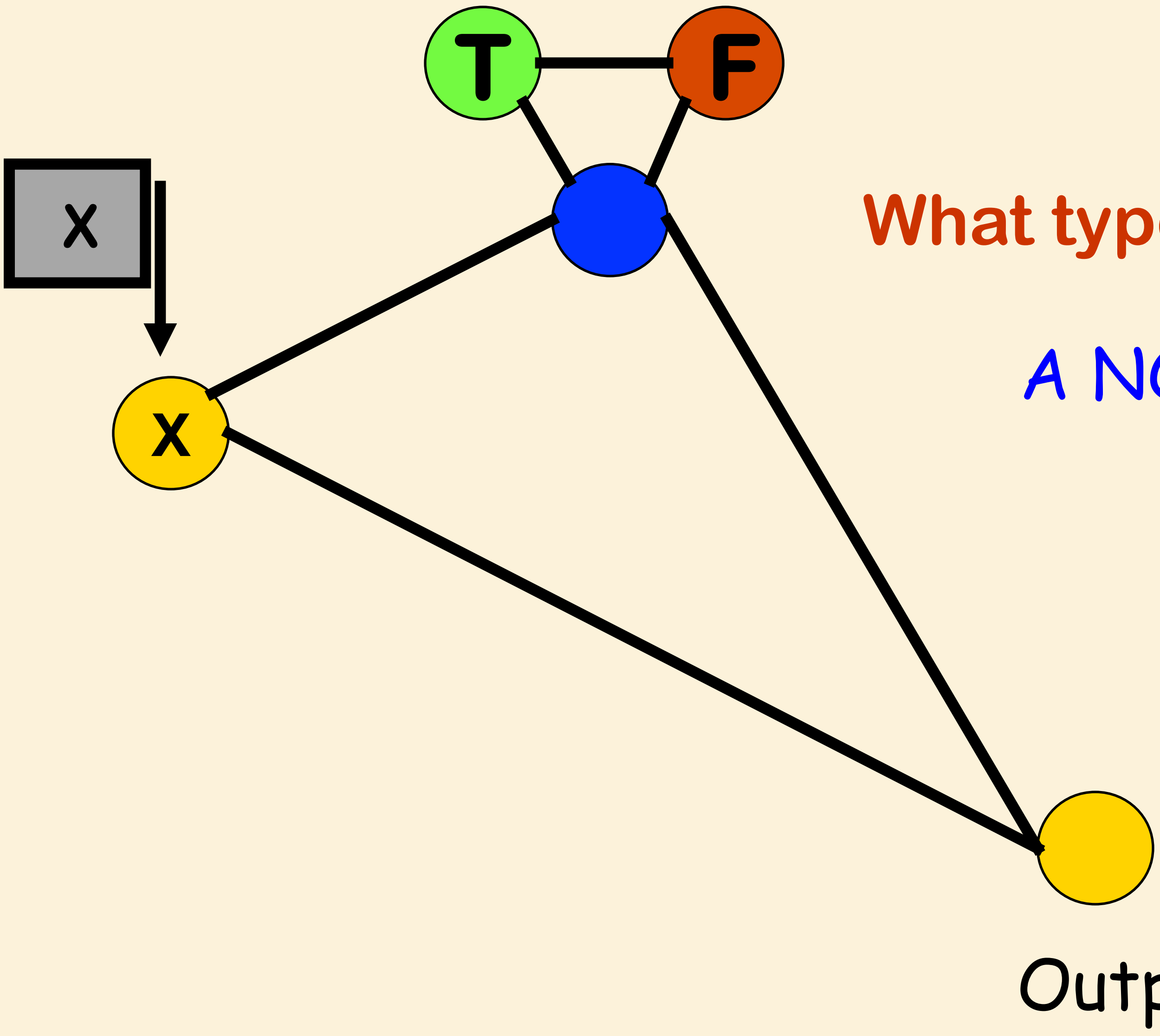


What type of gate is this?

An OR gate!

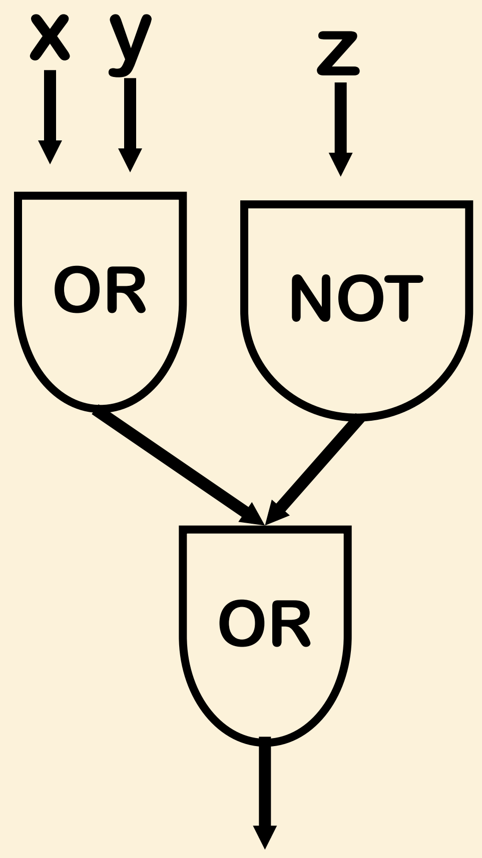
X	Y	Output
F	F	F
F	T	T
T	F	T
T	T	T

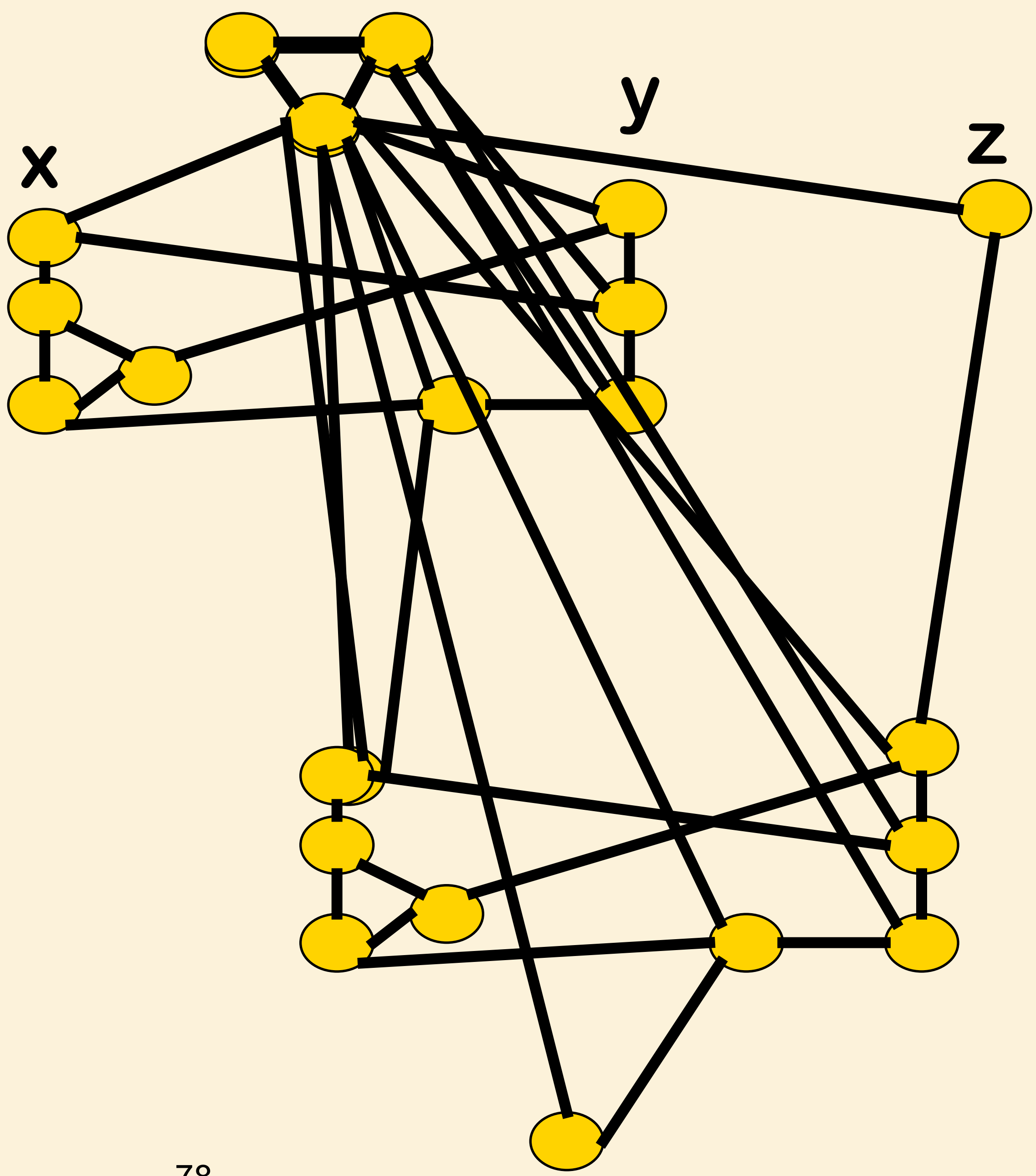
End of Final Lecture

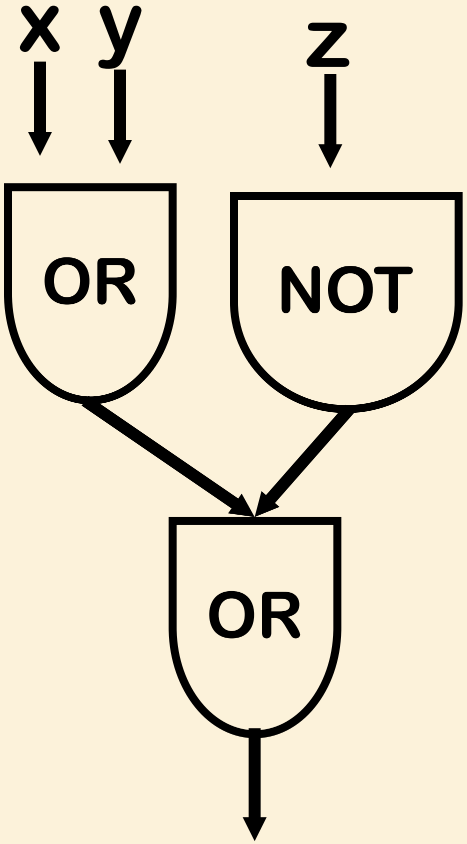


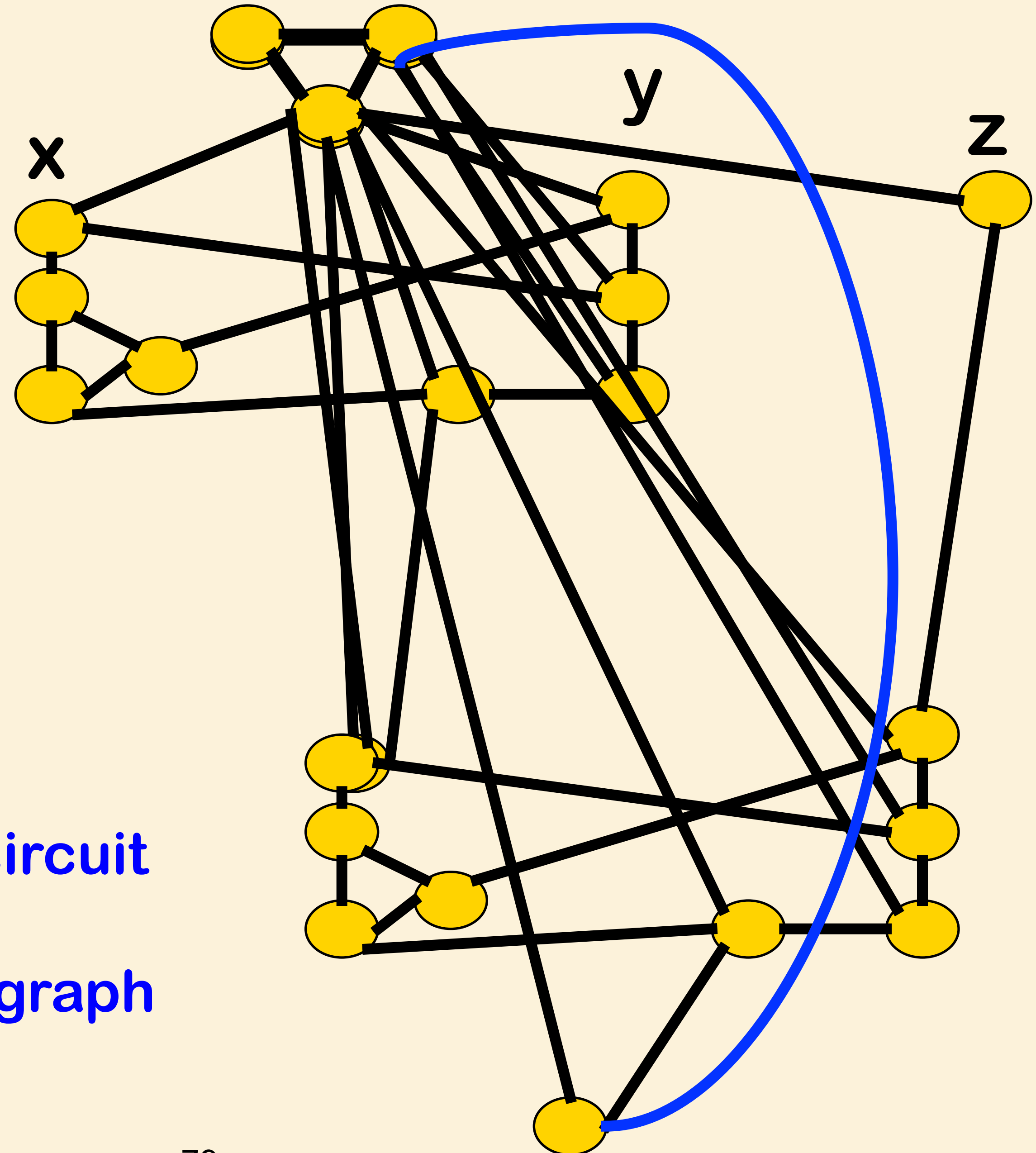
What type of gate is this?

A NOT gate!


x
y
z
OR
NOT
OR


x
y
z



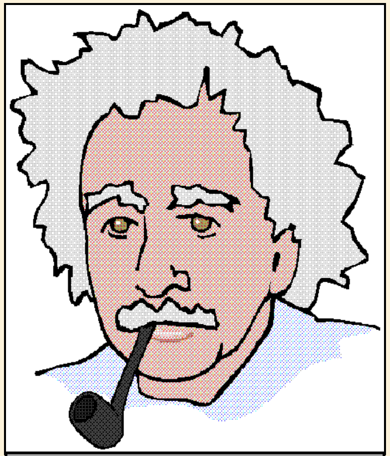


Satisfiability of this circuit
=
3-colorability of this graph

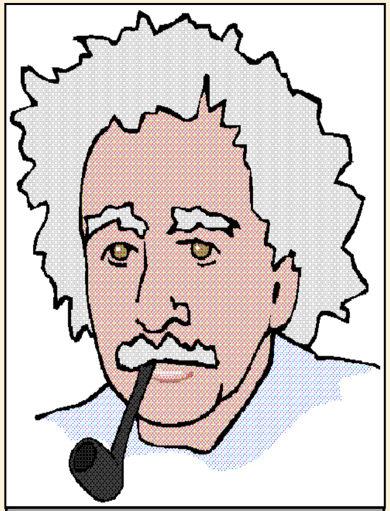
Let C be an n-input circuit.

C

Graph composed of gadgets that mimic the gates in C



BUILD:
SAT
Oracle



GIVEN:
3-color
Oracle

Formal Statement

- There is a polynomial-time function f such that:

- C is satisfiable <-> f(C) is 3 colorable

4 Problems All Equivalent

- If you can solve one quickly then you can solve them all quickly:

- Circuit-SAT
- Clique
- Independent Set
- 3 Colorability

SEARCH VERSUS DECISION.

CIRCUIT SAT DECISION VERSION.

GIVEN A CIRCUIT, IS IT SATISFIABLE?

CIRCUIT SAT SEARCH VERSION.

GIVEN A CIRCUIT, PRODUCE A SATISFYING ASSIGNMENT OR SAY THAT THERE IS NONE.

GIVEN AN ORACLE THAT SOLVES THE SEARCH VERSION, WE CAN CERTAINLY USE IT TO SOLVE THE DECISION VERSION.

MORE INTERESTING.....

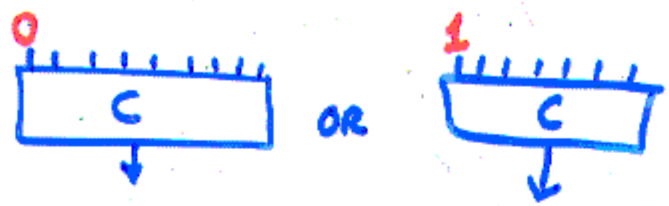
GIVEN AN ORACLE FOR THE DECISION VERSION, WE CAN QUICKLY SOLVE THE SEARCH VERSION!

GIVEN CIRCUIT C.

IF C → $\boxed{\text{DECISION ORACLE}}$ GETS

NO: THEN OUTPUT "NO ASSIGNMENT"

YES:



MUST BE SATISFIABLE. USE ORACLE TO DECIDE WHICH.

MUST BE SATISFIABLE. USE ORACLE TO DECIDE WHICH.

⋮

3-COLORING (DECISION)

GIVEN A GRAPH, IS IT 3 COLORABLE?

3-COLORING (SEARCH)

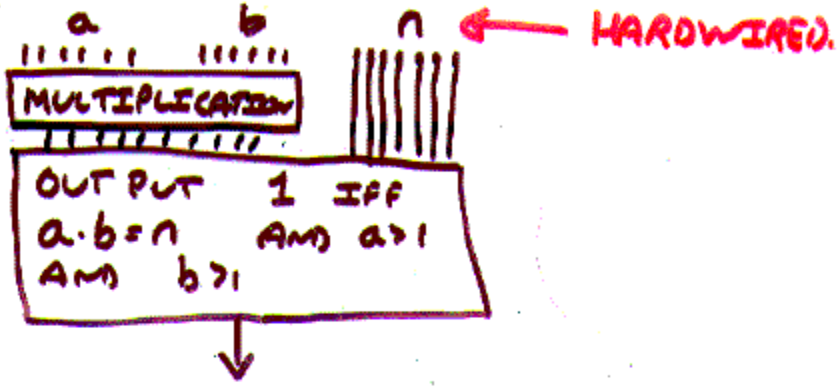
GIVEN A GRAPH, FIND A 3-COLORING IF IT EXISTS.

GIVEN AN ORACLE FOR THE DECISION VERSION OF 3-COLORING,

HOW DO WE SOLVE THE SEARCH VERSION.

USING AN ORACLE FOR CIRCUIT SAT WE CAN FACTOR A GIVEN NUMBER n.

LET C_n BE THE FOLLOWING CIRCUIT:



GIVE C_n TO THE ORACLE FOR THE SEARCH VERSION OF CIRCUIT SAT. THE ORACLE WILL PRODUCE a AND b, FACTORS OF n. (UNLESS n IS PRIME)

FACTORING

(PROGRESS REPORT)

AFTER 2000 YEARS OF RESEARCH, OUR BEST ALGORITHM CAN FACTOR A d-DIGIT NUMBER IN

$$e^{K\sqrt[3]{d \log^2 d}}$$ TIME.

---

12 TOP OF THE LINE COMPUTERS WORKING FOR 2 MONTHS CAN FACTOR 130 DIGIT NUMBERS.

CIRCUIT SAT

- NO FAST ALGORITHM IS KNOWN.
- A FAST ALGORITHM WOULD BE FAIRLY AMAZING.
- PEOPLE STRONGLY SUSPECT THAT NOTHING MUCH BETTER THAN BRUTE FORCE CAN WORK!

3-COLORING IS AS HARD AS CIRCUIT-SAT.

3-COLORING IS POLYNOMIAL TIME $\Rightarrow$ CIRCUI-SAT IS POLYNOMIAL TIME

$\Rightarrow$ FACTORING IS POLYNOMIAL TIME.

IF YOU FOLLOWED THE REASONING SO FAR YOU UNDERSTAND THAT GIVEN ANY NUMBER n I CAN MAKE A GRAPH G_n SO THAT:

I CAN GIVE SOMEONE G_n AND 3 CRAYONS AND TELL THEM TO 3-COLOR IT;

IF THEY SUCCEED, I CAN READ OFF THE FACTORS OF n ENCODED IN THE COLORS OF CERTAIN NODES!

CIRCUIT SAT IS <u>VERY</u> EXPRESSIVE.

IT CAN EXPRESS <u>ANY</u> CONSTRAINT SATISFACTION PROBLEM.

A **SOLUTION CHECKER** IS AN ALGORITHM CHECK THAT TAKES TWO INPUTS: INSTANCE AND SOLUTION. CHECK MUST RUN IN TIME POLYNOMIAL IN THE LENGTH OF INSTANCE. IT MUST ALWAYS RETURN "YES" OR "NO"

A **CONSTRAINT SATISFACTION PROBLEM** IS A SET OF THE FORM:

$$\{\text{INSTANCE} \mid \exists \text{ SOLUTION SUCH THAT CHECK(INSTANCE, SOLUTION)} = \text{"YES"}\}$$

EXAMPLE: CHECK(G, C) = "YES" IFF C IS A 3-COLORING OF G.

$$\text{3COLOR} = \{G \mid \exists C \text{ SUCH THAT CHECK}(G,C) = \text{"YES"}\}$$

e.g. CIRCUIT-SAT

3-COLOLORABLE

EVEN NUMBERS

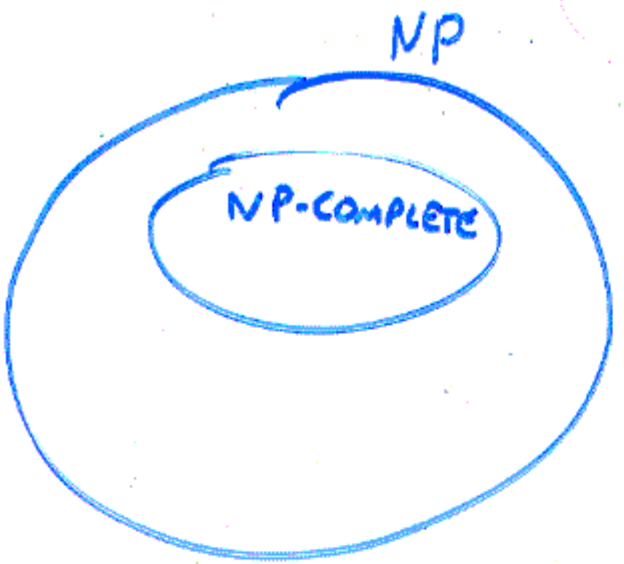
NP IS THE SET OF ALL CONSTRAINT SATISFACTION PROBLEMS.

CIRCUIT-SAT $\in$ NP

THE HARDEST PROBLEMS IN NP.

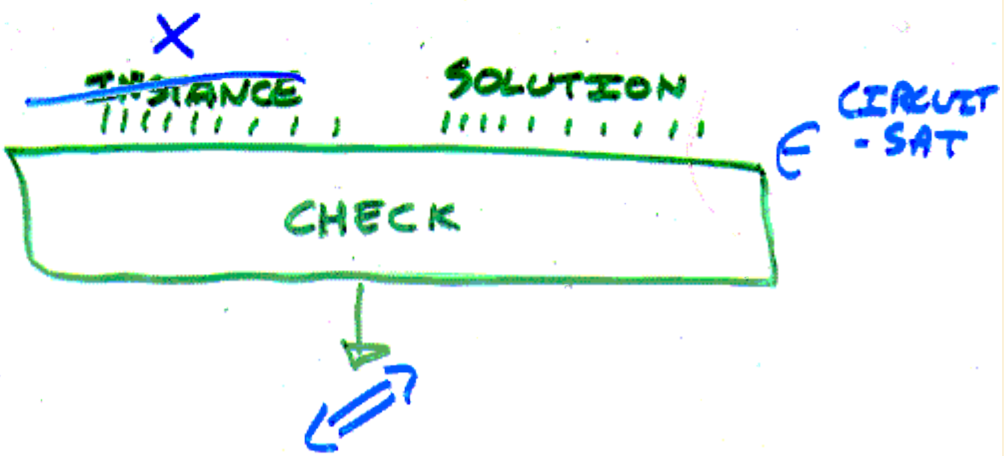
$A \in NP$ IS <u>NP-COMPLETE</u>

IF $\forall B \in NP$ B IS POLYNOMIALLY REDUCIBLE TO A



CIRCUIT-SAT IS NP-COMPLETE.

$$B = \{ \text{INSTANCE} \mid \exists \text{ SOLUTION } \text{CHECK}(\text{INSTANCE}, \text{SOLUTION}) = \text{"YES"} \}$$



$X \in B$

CIRCUIT-SAT IS NP-COMPLETE!

THAT MUST MEAN....

3-COLORABILITY IS ALSO NP-COMPLETE!

MANY FAMOUS

NP-COMPLETE PROBLEMS.

3-COLORING

TRAVELING SALESMAN

SUBSET SET

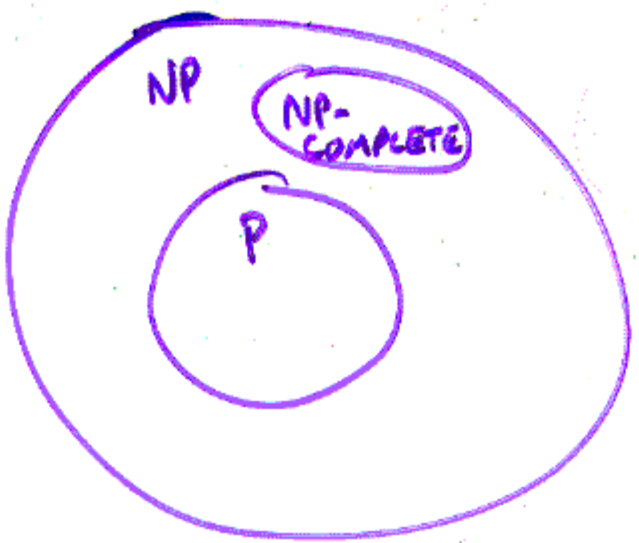
SAT

3-CNF SAT

⋮

(NOT FACTORING)

P: SET OF ALL DECISION PROBLEMS WITH POLYNOMIAL TIME ALGORITHMS.



$$P \stackrel{?}{=} NP$$

IS THERE AN NP-COMPLETE PROBLEM THAT CAN BE SOLVED IN POLYNOMIAL TIME?