Analysis of Algorithms

V. Adamchik CSCI 570 Spring 2020

Lecture 13 University of Southern California

NP-Completeness

Reading: chapter 9

P and NP complexity classes

P = set of problems that can be solved in polynomial time by a deterministic TM.

NP = set of problems that can be solved in polynomial time by a nondeterministic TM.

$P \subset NP$

NP = set of problems for which solution can be verified in polynomial time by a deterministic TM.

P

Polynomial Reduction: $Y \leq_p X$

To reduce a decision problem Y to a decision problem X (we write $Y \leq_p X$) we want a function f that maps Y to X such that:

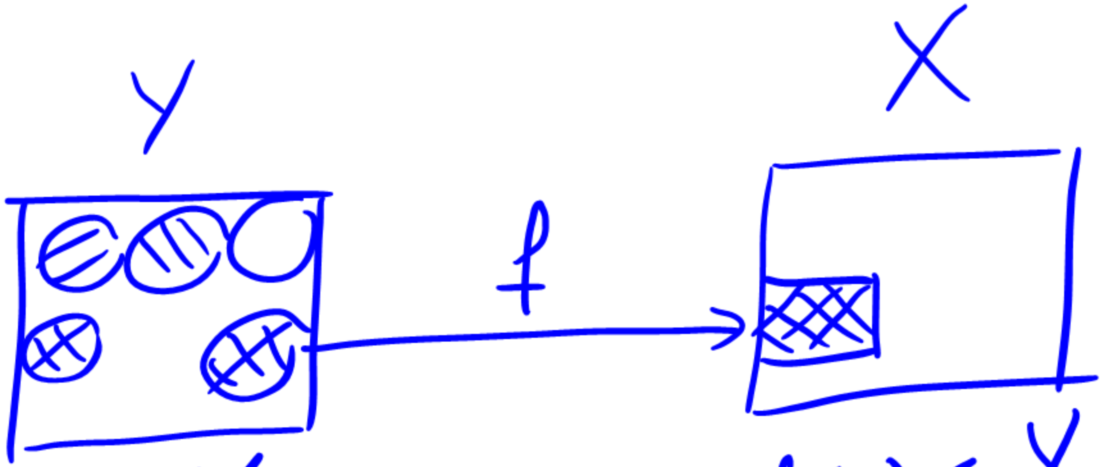
1) f is a polynomial time computable

2) $\forall y \in Y$ (y is instance of Y) is YES $\boxed{\text{if and only}}$ if $f(y) \in X$ is YES.

If we cannot solve Y, we cannot solve X.

We use this to prove NP completeness: knowing that Y is hard, we prove that X is at least as hard as Y.

Polynomial Reduction: $Y \leq_p X$



$\forall y \in Y$

$f(y) \in X$

$\cancel{p_1 = p_2} = \dots p_n$

#6

Y = partition

$\forall \; x_1, x_2, \dots, x_n \longrightarrow X$

$p_1, p_2, \dots, p_n$

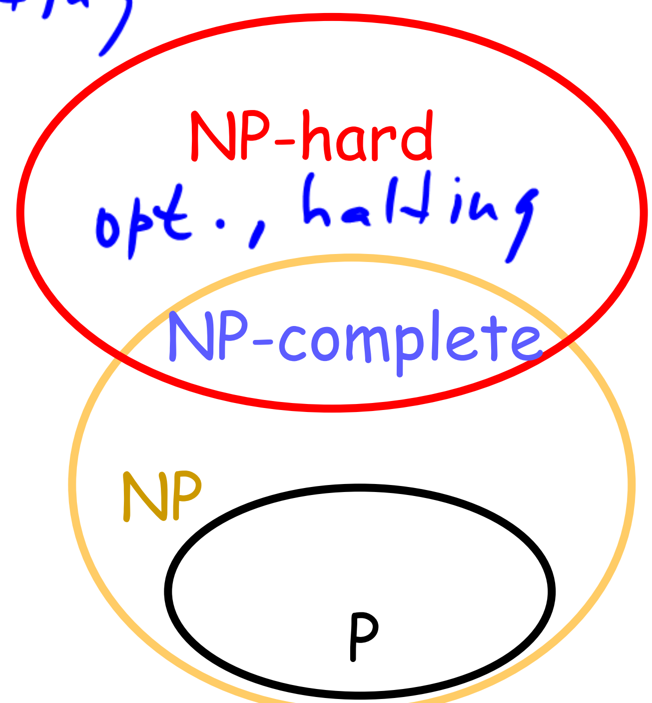
$\boxed{K \; W}$

NP-Hard and NP-Complete

X is *NP-Hard*, if $\forall Y \in NP$ and $Y \leq_p X$.

X is *NP-Complete*, if X is NP-Hard and $X \in NP$.

$SAT \leq_p$ Halting



Assuming $P \neq NP$.

NP-Completeness Proof Method

$Y \leq_p X$

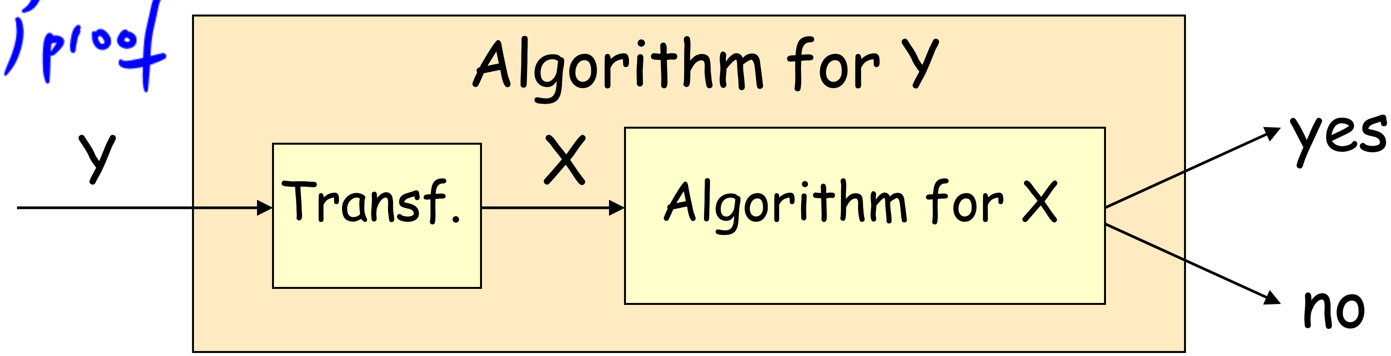
To show that X is NP-Complete:

hard

1) Show that X is in NP
2) Pick a problem Y, known to be an NP-Complete

NP

3) Prove $Y \leq_p X$ (reduce Y to X)
   a) construction, map f
   b) f is polynomial
   c) write claim
   d) proof



Boolean Satisfiability Problem (SAT)

A propositional logic formula is built from variables, operators AND (conjunction, $\wedge$), OR (disjunction, $\vee$), NOT (negation, $\neg$), and parentheses:

and

CNF $(X_1 \vee \neg X_3) \wedge (X_1 \vee \neg X_2 \vee X_4 \vee X_5) \wedge \ldots$ = True

or

A formula is said to be satisfiable if it can be made TRUE by assigning appropriate logical values (TRUE, FALSE) to its variables.

Cook-Levin

Theorem. *SAT is NP-complete.* (no reduction)

EC ⇔ HC ? think !

Euler Cycle $\in P$

Hamiltonian Cycle Problem

A Hamiltonian cycle (HC) in a graph is a cycle that visits each vertex exactly once.

$a \le_p b,\ b \le_p c \Rightarrow a \le_p c$

<u>Problem Statement:</u>

Given a directed or undirected graph G = (V,E). Find if the graph contains a Hamiltonian cycle.

$SAT \le_p HC$

We can prove it that HC problem is NP-complete by reduction from SAT, but we won't.

(TSP)

Traveling Salesman Problem

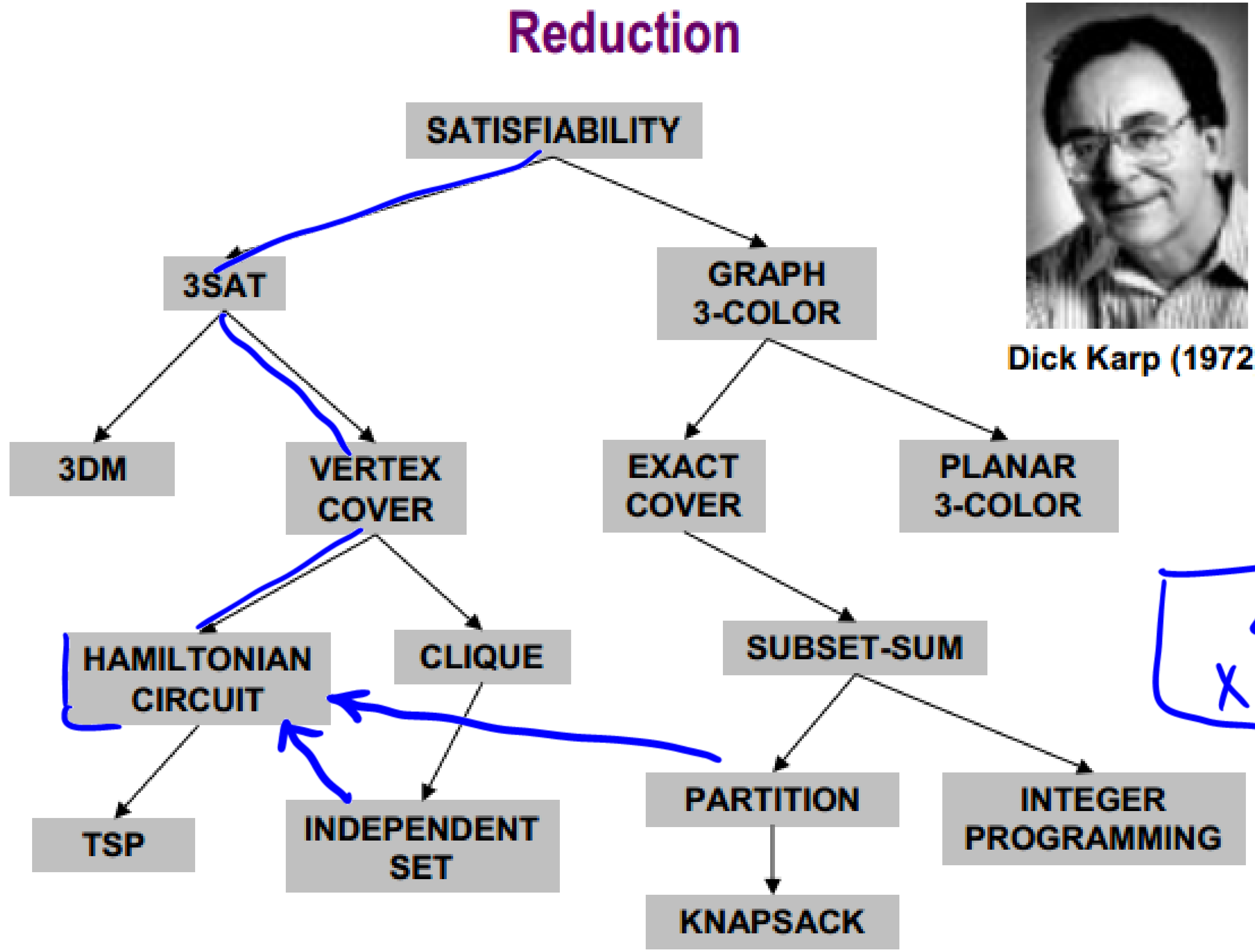
Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city and returns to the origin city?

<u>Problem Statement:</u>

Given a weighted graph G=(V,E) with positive edge costs, is there a Hamiltonian cycle that has total cost $\leq$ k?

We can prove that it is a NP-compete problem.

Reduction



Dick Karp (1972)

ILP

LP

$x_k \in \mathbb{N}$

prove!

Karp introduced the now standard methodology for proving problems to be NP-Complete.

He received a Turing Award for his work (1985).

Discussion Problem 1

Given SAT in Conjunctive Normal Form (CNF)

$$(X_1 \lor \neg X_3) \land (X_1 \lor \neg X_2 \lor X_4 \lor X_5) \land \ldots$$

with any number of clauses and any number of literals in each clause. Prove that SAT is polynomial time reducible to 3SAT.

$3SAT \le_p VC$

$SAT \in NPC$

is $3SAT \in NPC$?

$2SAT \in P$

$\boxed{SAT \le_p 3\text{-}SAT}$

$$(a \lor b \lor c \lor d) \overset{?}{=} (\underbrace{a \lor b \lor c}_{F}) \land (\underbrace{b \lor c \lor d}_{T})$$

$d = T$, $a = b = c = F$

F

extra var. $\in 3SAT$

RHS is T

$$a\vee b\vee c\vee d = \underbrace{(\underbrace{a\vee b}_{T}\vee x)}_{T}\wedge(\bar{x}\vee \underbrace{c\vee d}_{T/F})$$

$$\boxed{x=F}$$

Proof.

$a\vee b = T$

$c, d = \forall$

$c\vee d = T$

$a, b = \forall$

$$(\underbrace{a\vee b}_{T/F}\vee x)\wedge\underbrace{(\bar{x}\vee \underbrace{c\vee d}_{T})}_{T}$$

$$\boxed{x=T}$$

$a=b=c=d=F$

never occurs

$$\underbrace{(\underbrace{a\vee b}_{F}\vee x)\wedge(\bar{x}\vee \underbrace{c\vee d}_{F})}_{F}$$

$$\overset{=z}{(a \vee b \vee c \vee (d \vee e))} = (a \vee b \vee c \vee z)$$

$$= (a \vee b \vee x) \wedge (\bar{x} \vee c \vee z)$$

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

$$= (a \vee b \vee x) \wedge (\bar{x} \vee c \vee y) \wedge (\bar{y} \vee d \vee e)$$

z

NPC

$x \quad y$

$x \le_p y$

$y \le_p x$

$(a \vee b \vee c \vee d \vee e \vee f) = \dots$

is it polynomial? SAT: n-literals, m-clauses

3SAT: literals $\to (n-3)m$, clauses $\to (n-2)m$

Claim. SAT is satisfiable iff 3SAT is satis

proof. $\Rightarrow$ by construction

$\Leftarrow$ $3SAT \in SAT$

Discussion Problem 2

Assuming that finding a Hamiltonian Cycle (HC) in a graph is NP-complete, prove that finding a Hamiltonian Path is also NP-complete. HP is a path that visits each vertex exactly once and isn't required to return to its starting point.

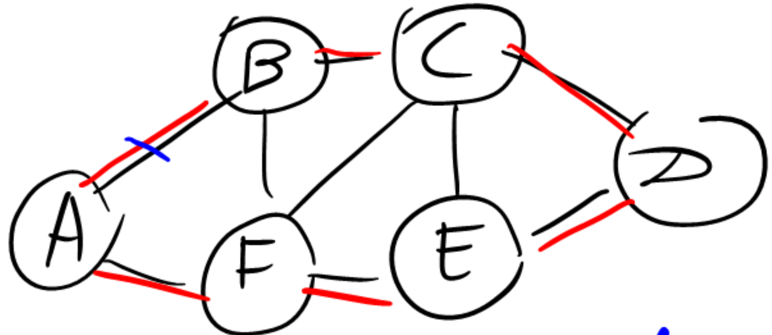
$HC \in NPC$, prove $HP \in NPC$

1) $HP \in NP$, easy

2) $HC \leq_p HP$

G

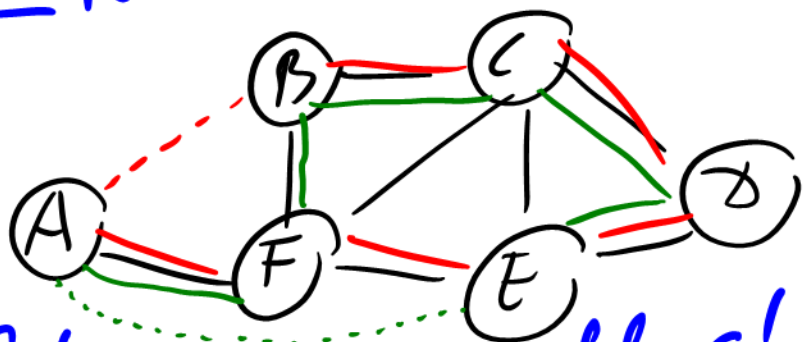
HC ∈ G



Y

G' — remove an edge from a HC

HP ∈ G'



X

<u>Claim</u>. G has a HC iff G' has a HP

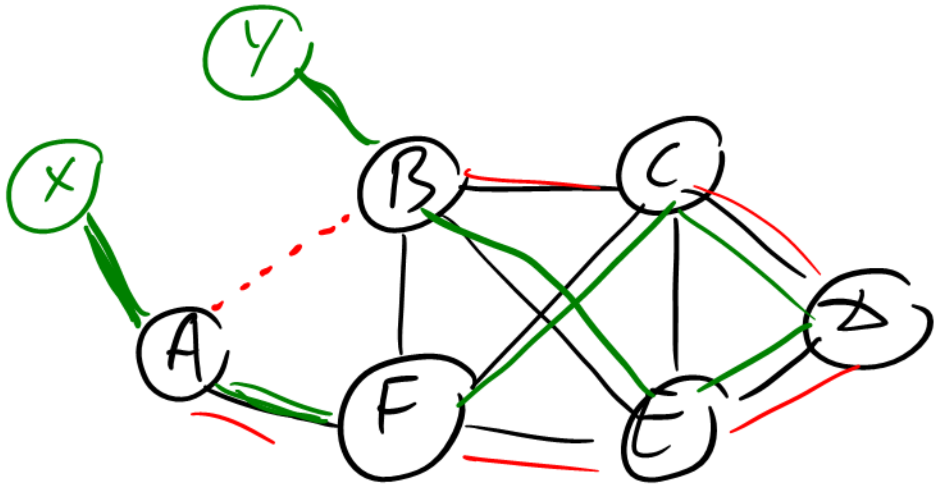
Proof. by construction

⇒ G has a HC.

⇐ G' has a HP.

wrong reduction

G'



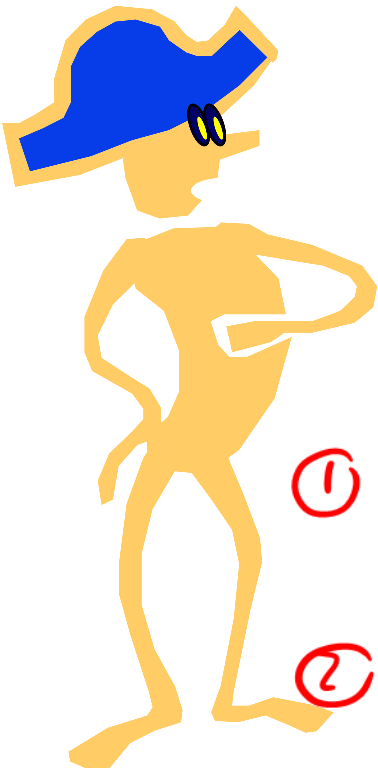
G' remove an edge on a HC in G and two new vertices connected to edge points

Claim. as before

$\Rightarrow$ by construction

$\Leftarrow$ G' has a HP.

Graph Coloring



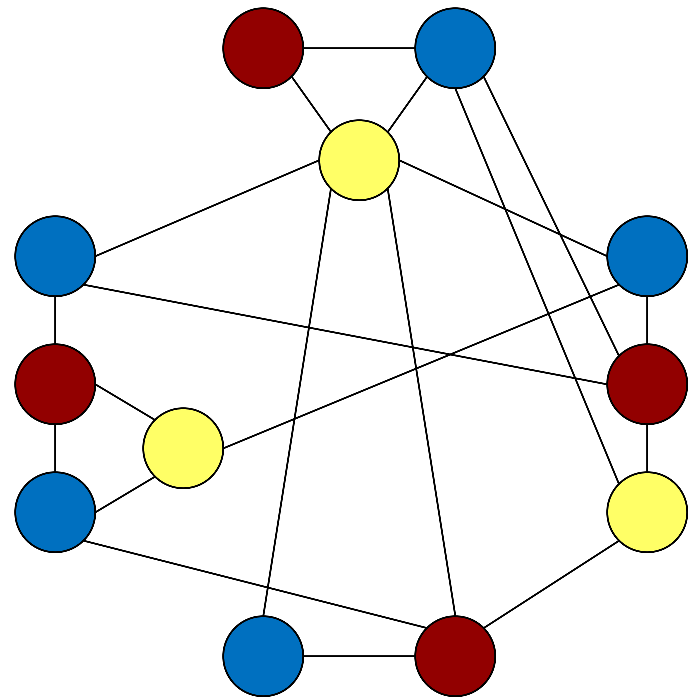
Given a graph, can you color the nodes with $\leq k$ colors such that the endpoints of every edge are colored differently?

① Planar graph Coloring $\in P$

$k = 4$

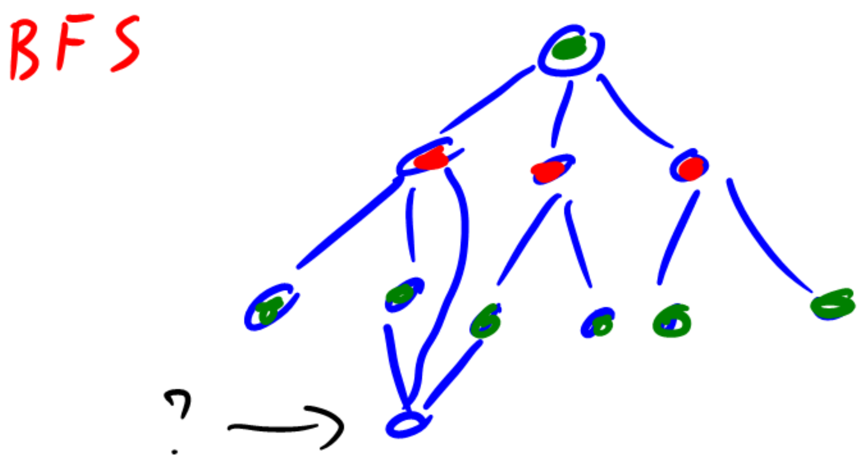
② if $k = 2$, in P

Theorem. ($k>2$)
k-Coloring is NP-complete.



Graph Coloring: k = 2

How can we test if a graph has a 2-coloring?



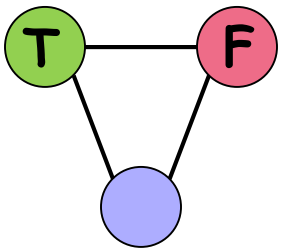
3-SAT $\leq_p$ 3-colorable

We construct a graph G that will be 3-colorable iff the 3-SAT instance is satisfiable.

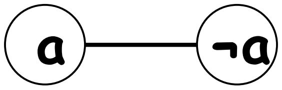
Graph G consists of the following gadgets.

only one

A truth gadget:



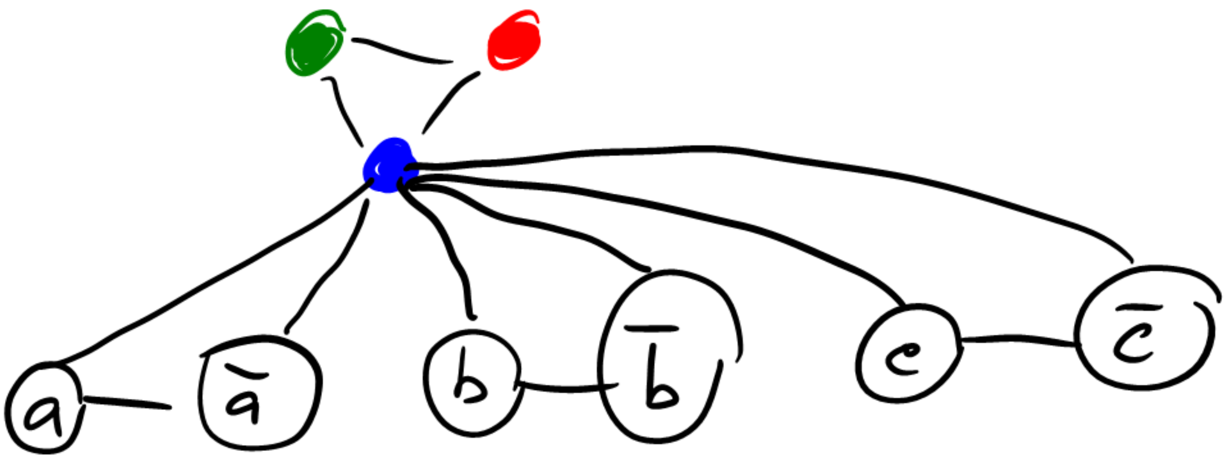
A gadget for $\boxed{\text{each}}$ variable:



3-SAT $\leq_p$ 3-colorable

Combining those gadgets together (for three literals)

a, b, c



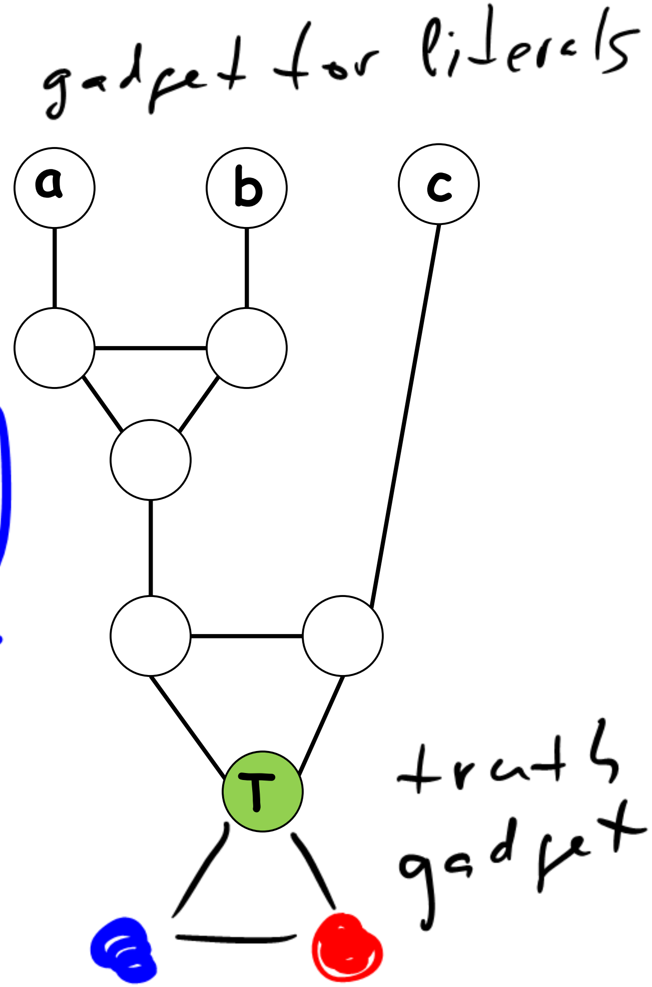
3-SAT $\leq_p$ 3-colorable

A special gadget for each clause

This gadget connects a truth gadget with variable gadgets.

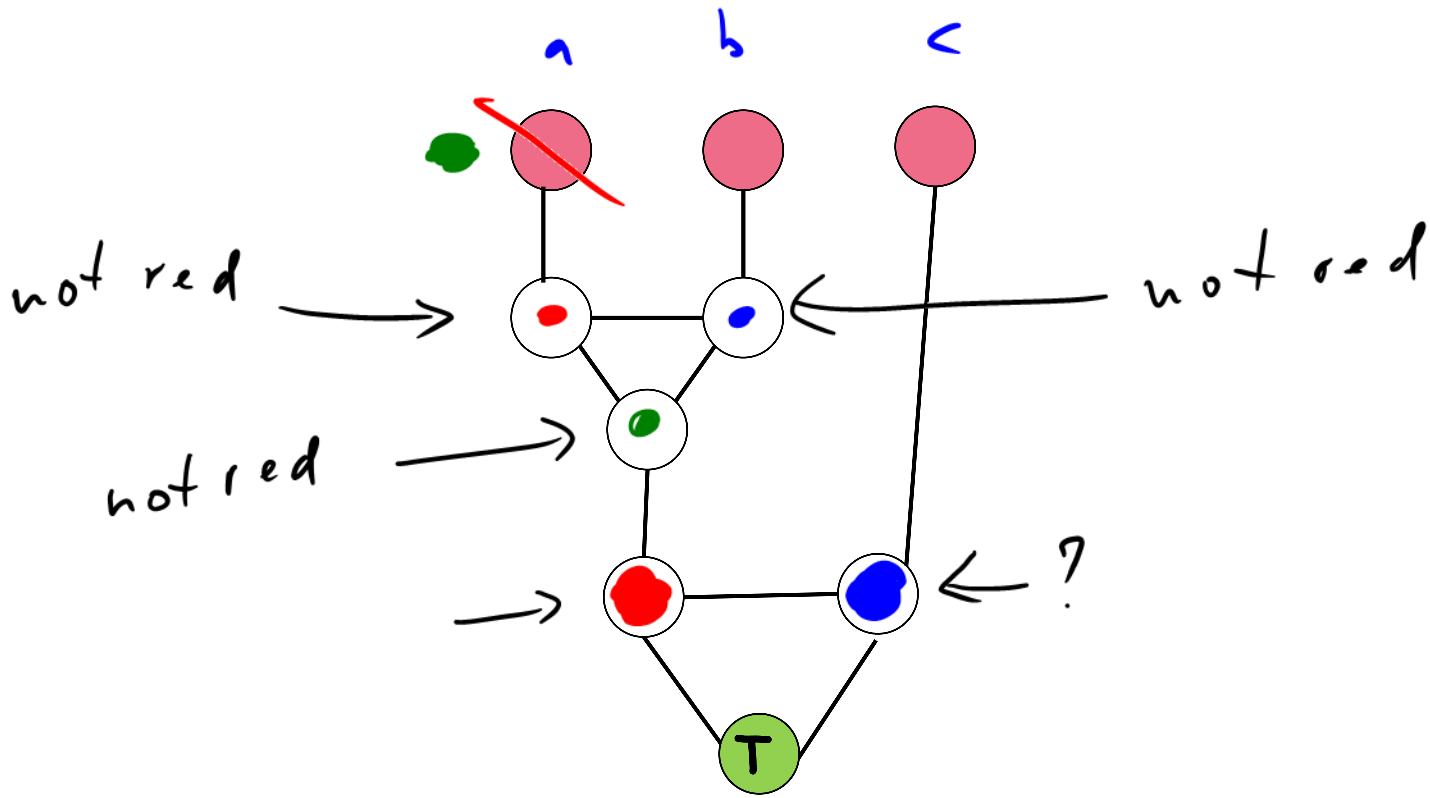
The bottom vertex is T (green) iff one of a, b, c is true.

$a \vee b \vee c = F$
if $a = b = c = F$



3-SAT $\leq_p$ 3-colorable

Suppose all a, b and c are all False (red).

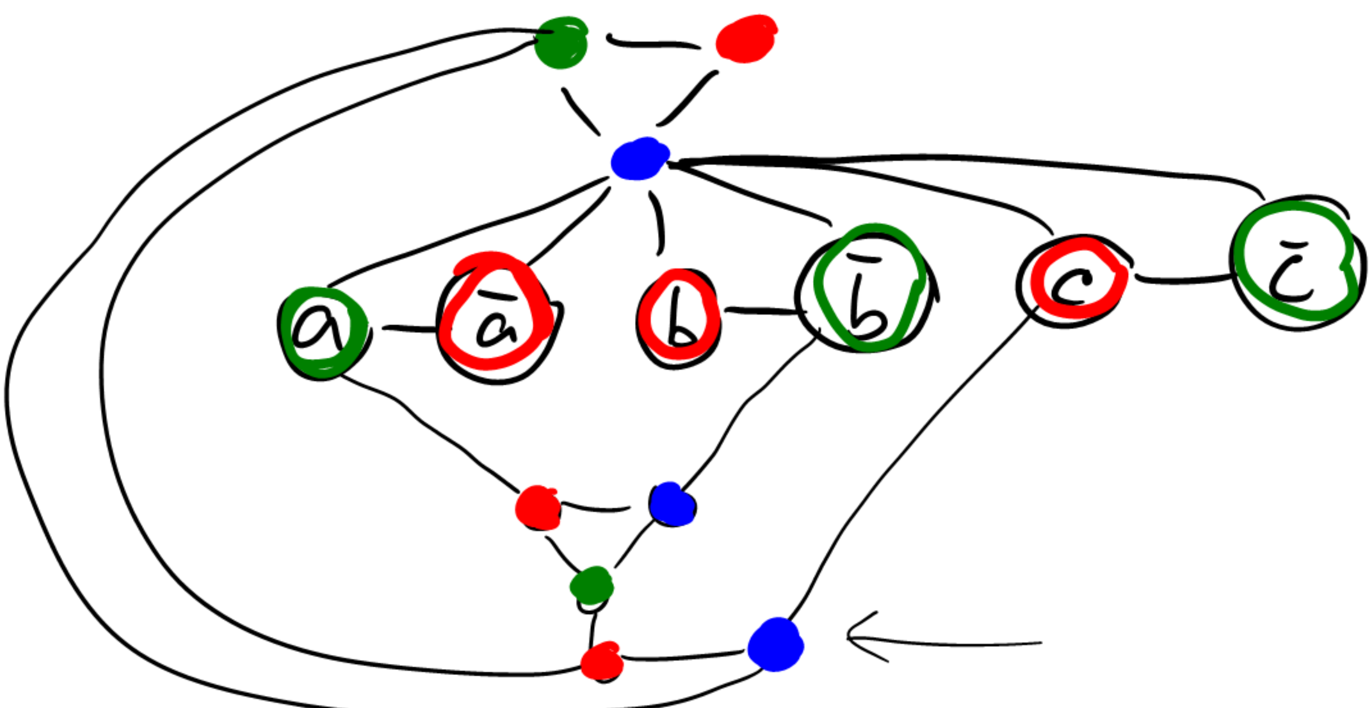


3-SAT $\leq_p$ 3-colorable

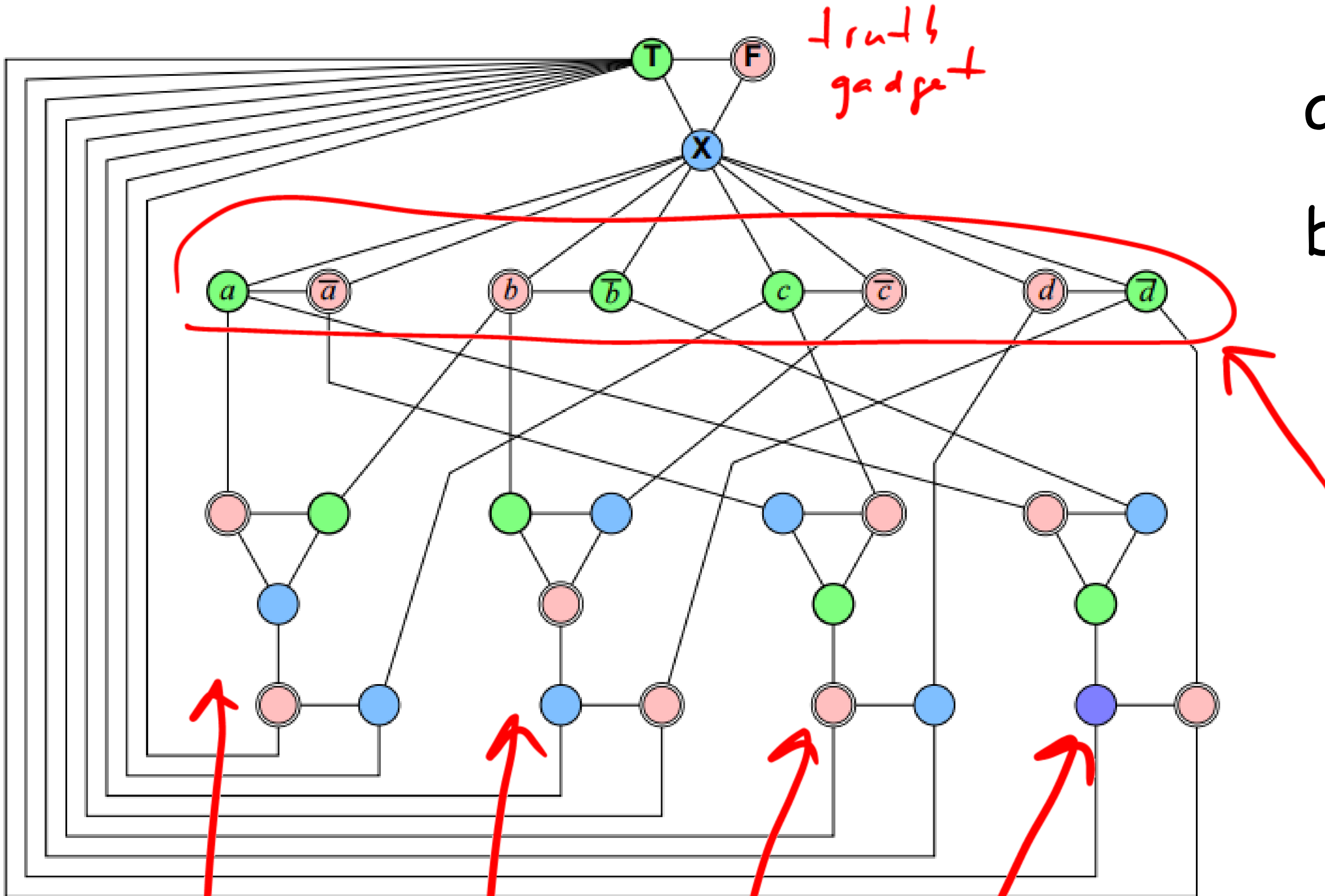
We have showed that if all the variables in a clause are false, the gadget cannot be 3-colored.

Let $a = T$, $b = c = F$

Example: $a \vee \neg b \vee c$



Example with four clauses



a=c=T

b=d=F

A 3-colorable graph derived from a satisfiable 3CNF formula.

SAT

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

3-SAT $\leq_p$ 3-colorable

<u>Claim</u>: *3-SAT instance is satisfiable if and only if G is 3-colorable.*

Proof: $\Rightarrow$) 3SAT has truth assignment

by construction

truth gadget -> color

variable gadgets -> color, T = green

clause gadgets -> coloring is forced

3-SAT $\leq_p$ 3-colorable

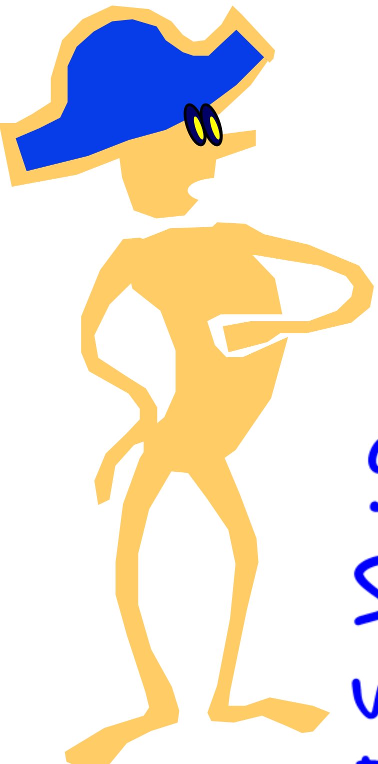
<u>Claim</u>: *3-SAT instance is satisfiable if and only if G is 3-colorable.*

Proof: $\Leftarrow$) G is 3-colorable
G is a special graph

Look at variable gadget and assign red to F, green to T.

Sudoku: $n^2 \times n^2$

$n \to \infty$



$\boxed{\text{NP-?}}$

$\boxed{\text{NP-hard?}}$

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

9-coloring

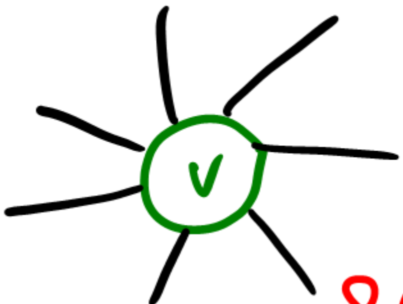
Sudoku graph:

Vertex: cell, 81

Edge: tw vertices is th same row or column or mini-grid

Sudoku Graph

how many edges? 810



degree?

$8+8+4=20$

$81 \cdot 20/2 = 810$

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

Sudoku

Constructing a Sudoku graph we have proved:

$$\text{Sudoku} \le_p 9\text{-coloring}$$

Did we prove that Sudoku $\in NPC$?

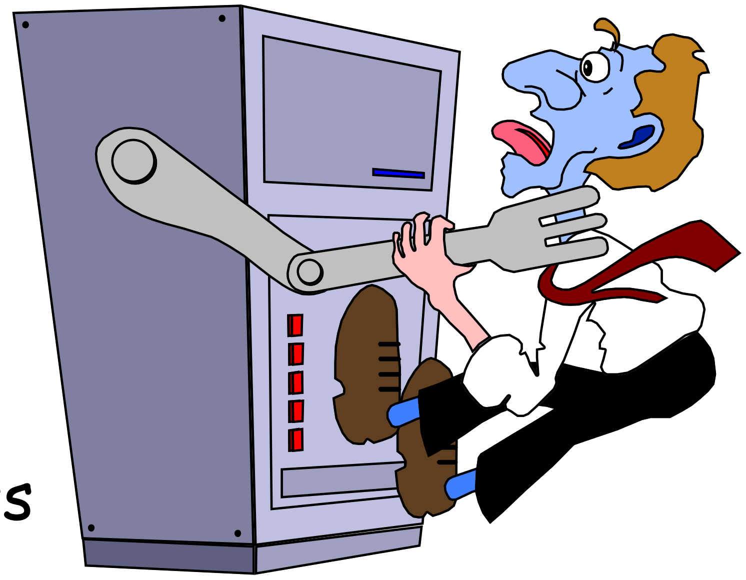
Proven Sudoku $\in NPC$.

How can you use this reduction?

We can solve Sudoku by coloring graph.

$Y \le_p SAT$, I can use SAT solver to solve problem Y.

Don't be afraid of NP-hard problems.



Many reasonable instances (of practical interest) of problems in class NP can be solved!

The largest solved $\boxed{\text{TSP}}$ an 85,900-vertex route calculated in 2006. The graph corresponds to the design of a customized computer chip created at Bell Laboratories, and the solution exhibits the shortest path for a laser to follow as it sculpts the chip.

Discussion Problem 3

The *Steiner Tree* problem is as follows. Given an undirected weighted graph $G=(V,E)$ with positive edge costs, a subset of vertices $R \subseteq V$, and a number C. Is there a tree in G that spans all vertices in R (and possibly some other in V) with a total edge cost of at most C? Prove that this problem is NP-complete by reduction from Vertex Cover.

Lenovo

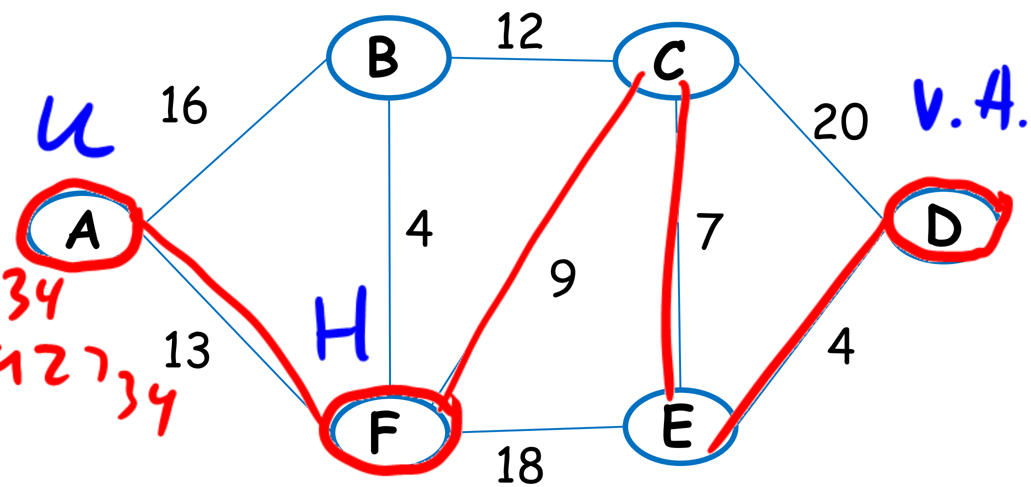
Example.

$R = \{A,F,D\}$ and $\boxed{C = 34.}$

MST ∈ P

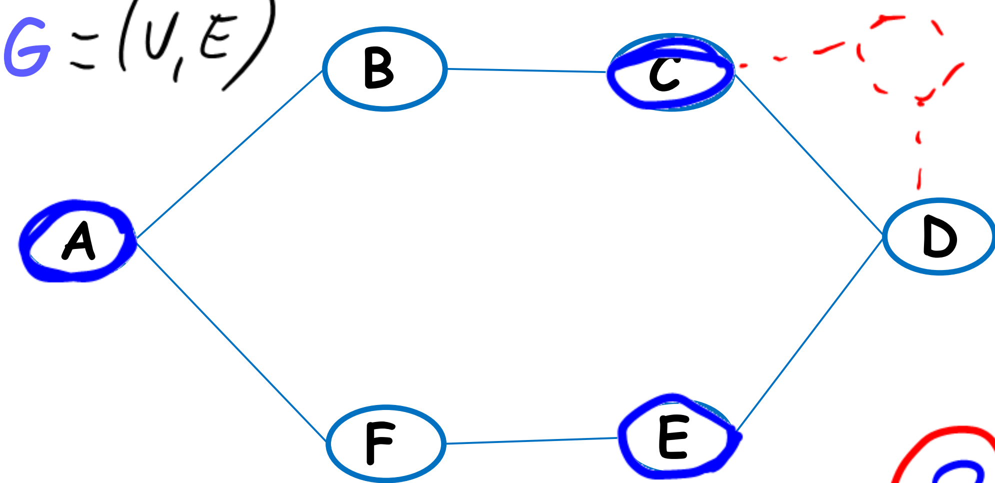
$13+18+4=35>34$

$13+9+20=42>34$



$G = (V, E)$

$VC(G) \le K$



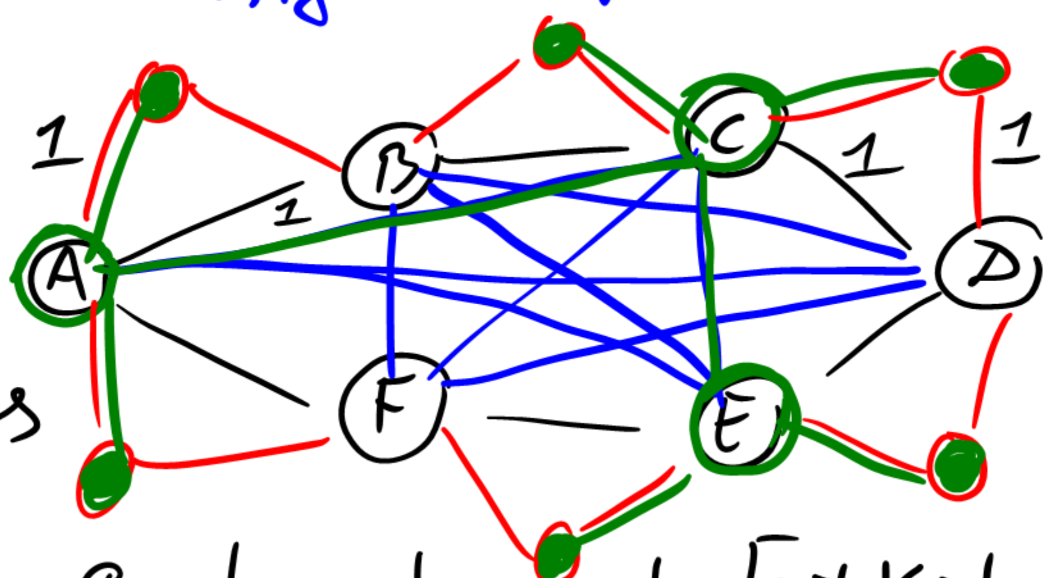
G', we have to define R and C. assign weights to each edge

G'

vertices: $V+E$

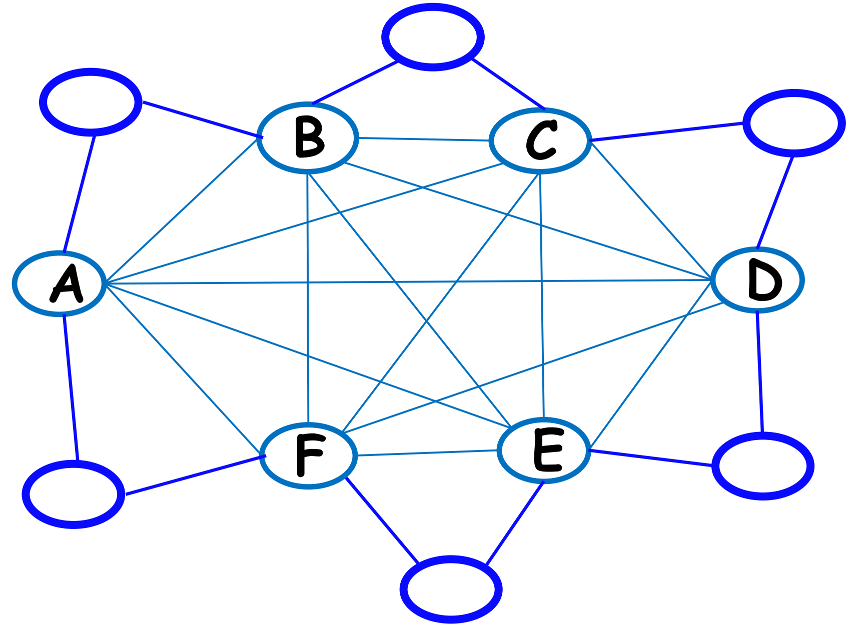
R = red vertices

steiner Tree in green.



Cost = at most $E+K-1$

Claim. G has a VC of size at most K iff G' has a Steiner tree with $R = E(G)$ and $C \leq E + K - 1$.

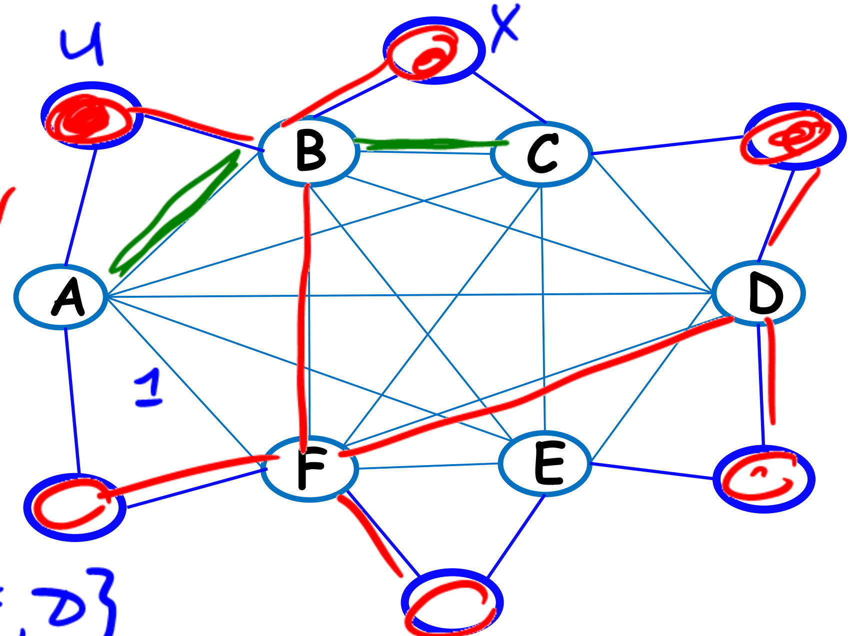


Proof.

$\Rightarrow$ G has a VC of size $\leq K$

By construction, we create a Steiner tree with $R = E(G)$ and $C \leq E + K - 1$

$\Leftarrow$) G' has a Steiner tree with R and $C \le E+K-1$



Find VC for G.

$\{B, F, D\}$

$VC(G) = ST - R$

a) prove that this is a $VC = ST - R$

b) $|VC| \le K$

$|VC| = |ST \text{ of vertices}| - R(\text{vertices})$

$= cost(ST) + 1 - E$

$\le (E+K-1) + 1 - E = K.$

Approximation Algorithms

ML

Suppose we are given an $\boxed{\text{NP-hard}}$ problem to solve.

OPT

suboptimal

Can we develop a polynomial-time algorithm that produces a “good enough" solution?

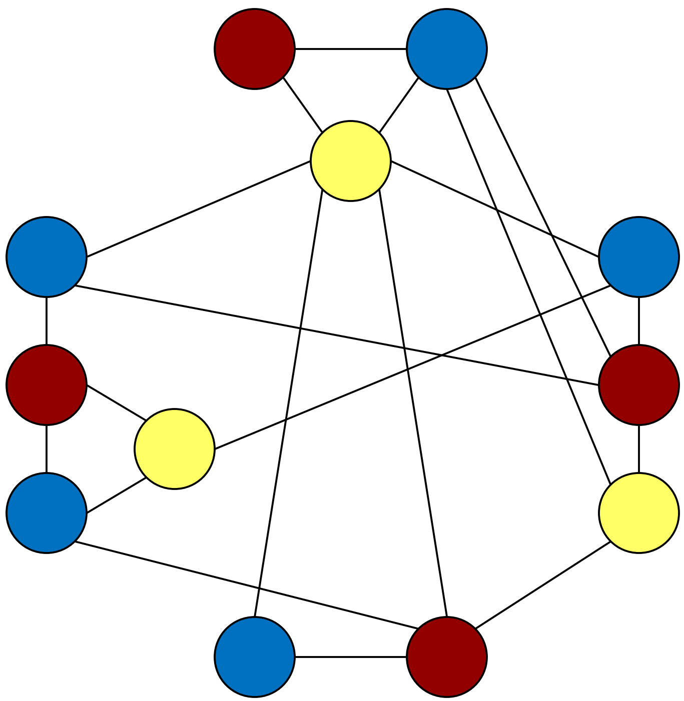
An algorithm that returns near-optimal solutions is called an *approximation algorithm*.

Graph Coloring

Given a graph G=(V,E), find the minimum number of colors required to color vertices, so no two adjacent vertices have the same color.

This is NP-hard problem.

Let us develop a solution that is close enough to the optimal solution.



Greedy Approximation

Given G=(V,E) with n vertices.

Use the integers {1,2,3, …, n} to represent colors.

Order vertices by degree in descending order.

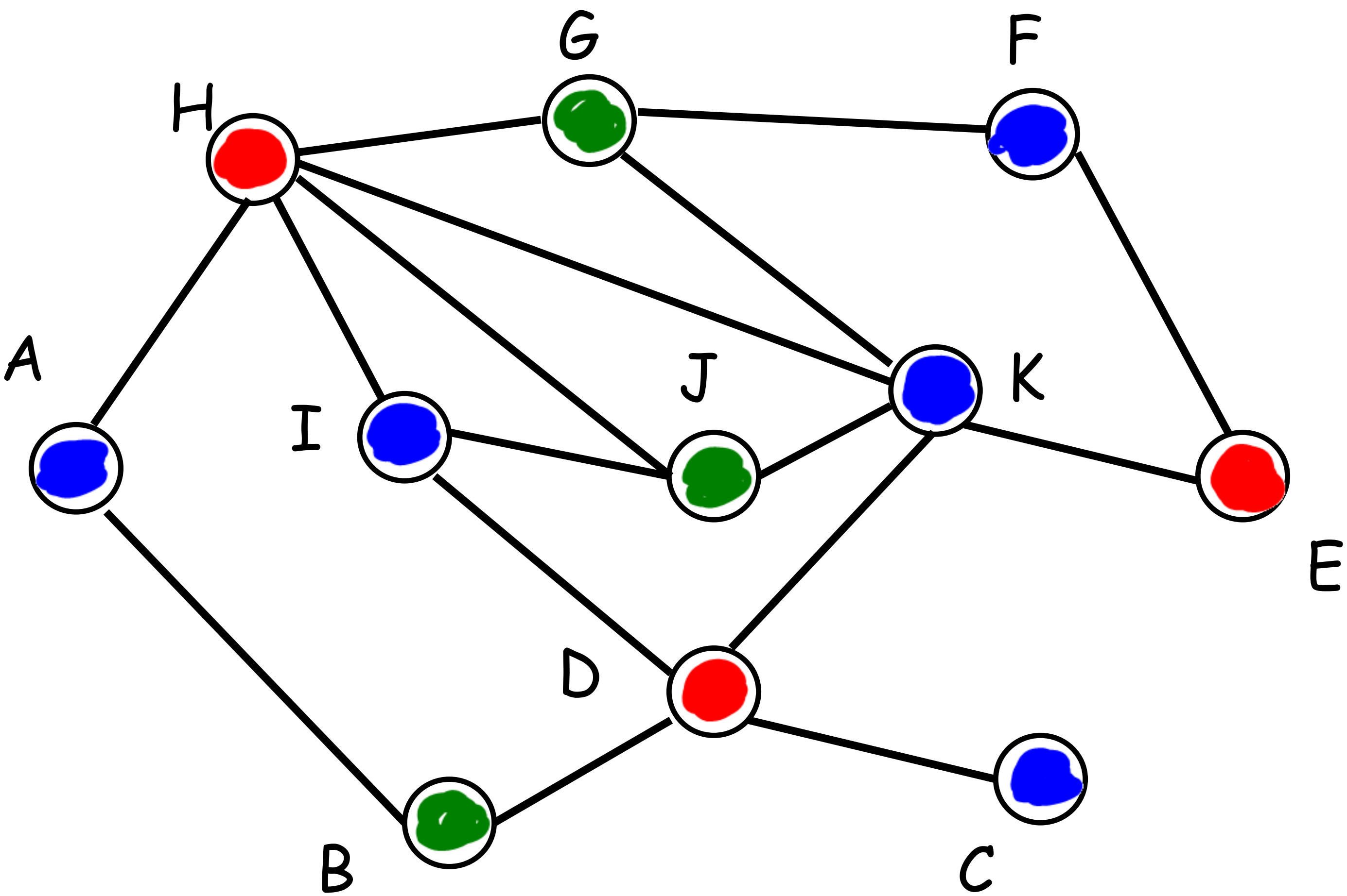
Color the first vertex (highest degree) with color 1.

Go down the vertex list and color every vertex not adjacent to it with color 1.

Remove all colored vertices from the list.

Repeat for uncolored vertices with color 2.

Example

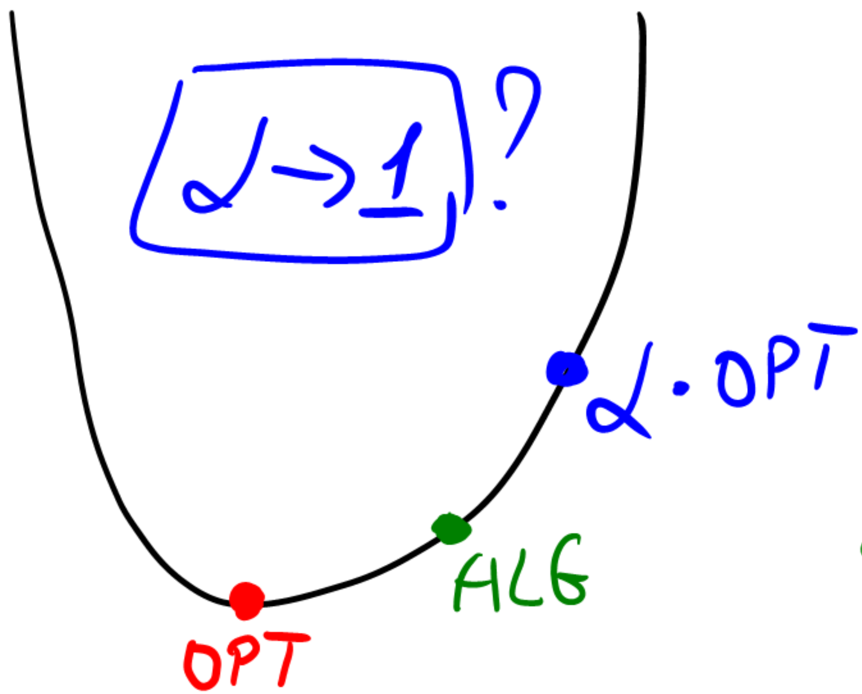


Order: ~~H~~, ~~K~~, ~~D~~, G, ~~I~~, J, ~~A~~, B, ~~E~~, ~~F~~, C

Formal Definition

Let P be a minimization problem, and I be an instance of P. Let ALG(I) be a solution returned by an algorithm, and let OPT(I) be the optimal solution.

Then ALG(I) is said to be a α-*approximation* algorithm for some $\alpha > 1$, if for $ALG(I) \leq \alpha \cdot OPT(I)$.



OPT = 4 bags

$\alpha = 2$

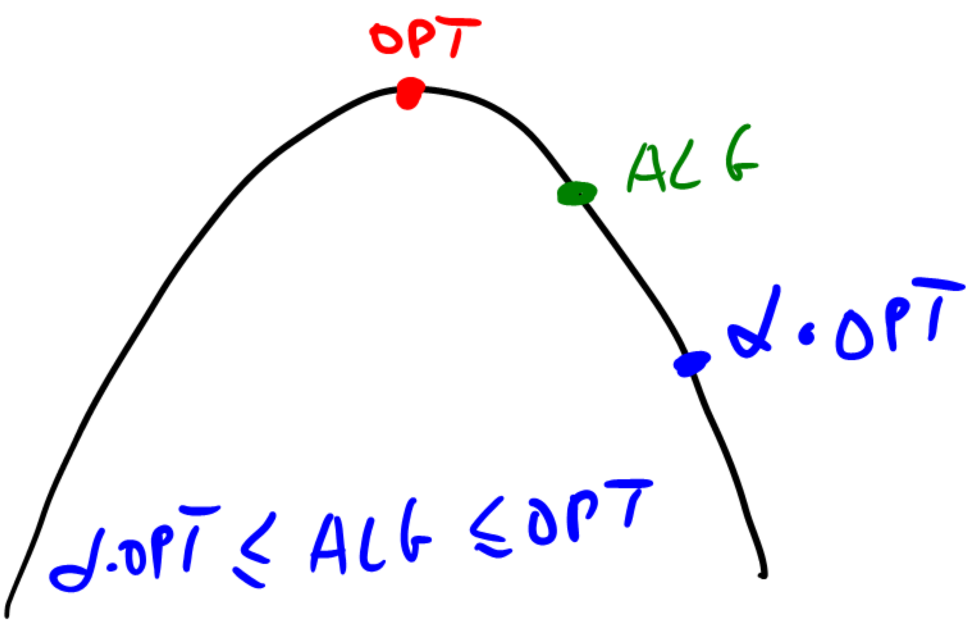
α-appr ≤ 8 bags

$OPT \leq ALG \leq \alpha \cdot OPT$

Maximization Problem

Let P be a <u>maximization</u> problem, and I be an instance of P.
Let ALG(I) be a solution returned by an algorithm, and let OPT(I) be the optimal solution.

Then ALG(I) is said to be a α *-approximation* algorithm for some $0 < \alpha < 1$, if for $ALG(I) \geq \alpha \cdot OPT(I)$.



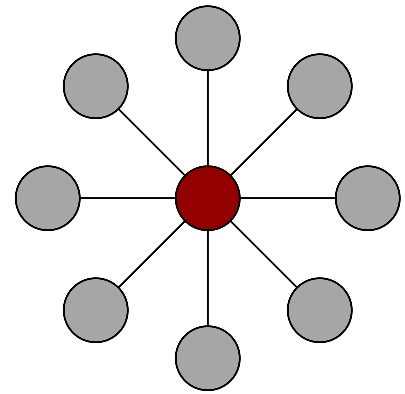
$\alpha \in \mathbb{R}$

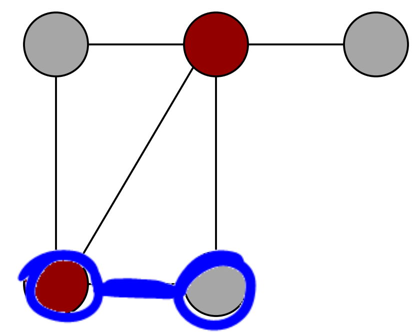
$\alpha = f(n)$

Vertex cover $\in NPH$

Given G=(V,E), find the smallest $S \subset V$ s.t. every edge is incident on a vertex in S.

Let us design a ~~greedy~~ random algorithm for vertex cover.





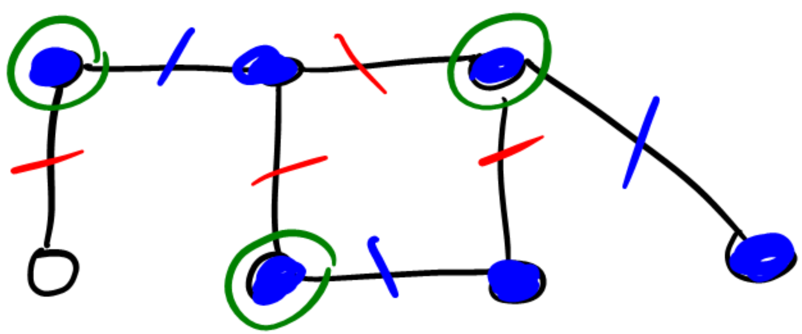
Example

α - approx.

to make α smaller

is it possible $\alpha < 2$

$\alpha = 1.9$



VC = 6

$\alpha = 2$

OPT = 3

Claim. $|ALG| \le 2 \cdot |OPT|$

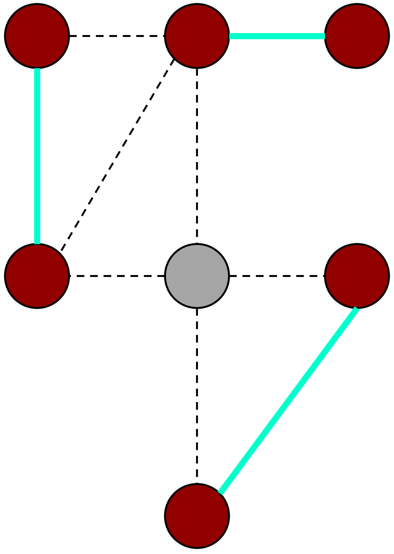
Proof. $|ALG| = |\text{matching}| \cdot 2$

$\le 2 \cdot |OPT|$

Vertex cover

Lemma. Let M be a matching in G, and S be a vertex cover, then

$$|S| \geq |M|$$



2-Approximation Vertex Cover

Approx-VC(G):
M - maximal matching on G
S - take both endpoints of edges in M
Return S

Theorem. Let OPT(G) be the size of the optimal vertex cover and S = ApproxVC(G). Then $|S| \leq 2 \cdot OPT(G)$.

Proof.

Can we do better than 2-Approximation?

$|OPT(K_{n,n})| = ? = n$

$|\text{Maximal Matching}| = ? = n$

$ALG(K_{n,n}) = 2n$

$\alpha = 2$

The End!

$K_{n,n}$