CS 381 – FALL 2019

Week 14.2, Wed, Nov 20

Homework 7 Due: November 26th at 11:59PM on Gradescope

Q1b Typo: if there is a directed from v_ℓ to $v_{\bar{\ell}}$ AND from $v_{\bar{\ell}}$ to v_ℓ then the 2-SAT instance is not satisfiable.

iClicker

Jane is excited! She thinks she has proved that P=NP. In particular, she claims to have proved that 2-SAT is NP-Complete by showing that $2SAT \leq_p 3SAT$. She has a reduction f which maps 2-SAT instances (Φ_{2SAT}) to 3-SAT instances (Φ_{3SAT}) in polynomial time. She proved that if Φ_{2SAT} is satisfiable then $\Phi_{3SAT} = f(\Phi_{2SAT})$ is satisfiable. What mistakes (if any) did Jane make?

A. Proving that 2-SAT is NP-Complete would not imply P=NP.
   - 2SAT is in P (see homework 7). This would imply P=NP.

B. Jane still needs to show that if Φ_{3SAT} is satisfiable then Φ_{2SAT} is satisfiable to conclude that $2SAT \leq_p 3SAT$.
   - Otherwise $f(\Phi_{2SAT})$ could ignore the input Φ_{2SAT} and *always* output a satisfiable 3SAT formula

C. The reduction is in the wrong direction to conclude that P=NP.
   - Jane would need to prove that $3SAT \leq_p 2SAT$

D. Claims B and C are both true.

E. Jane's logic is sound! She can claim the $1,000,000 prize!

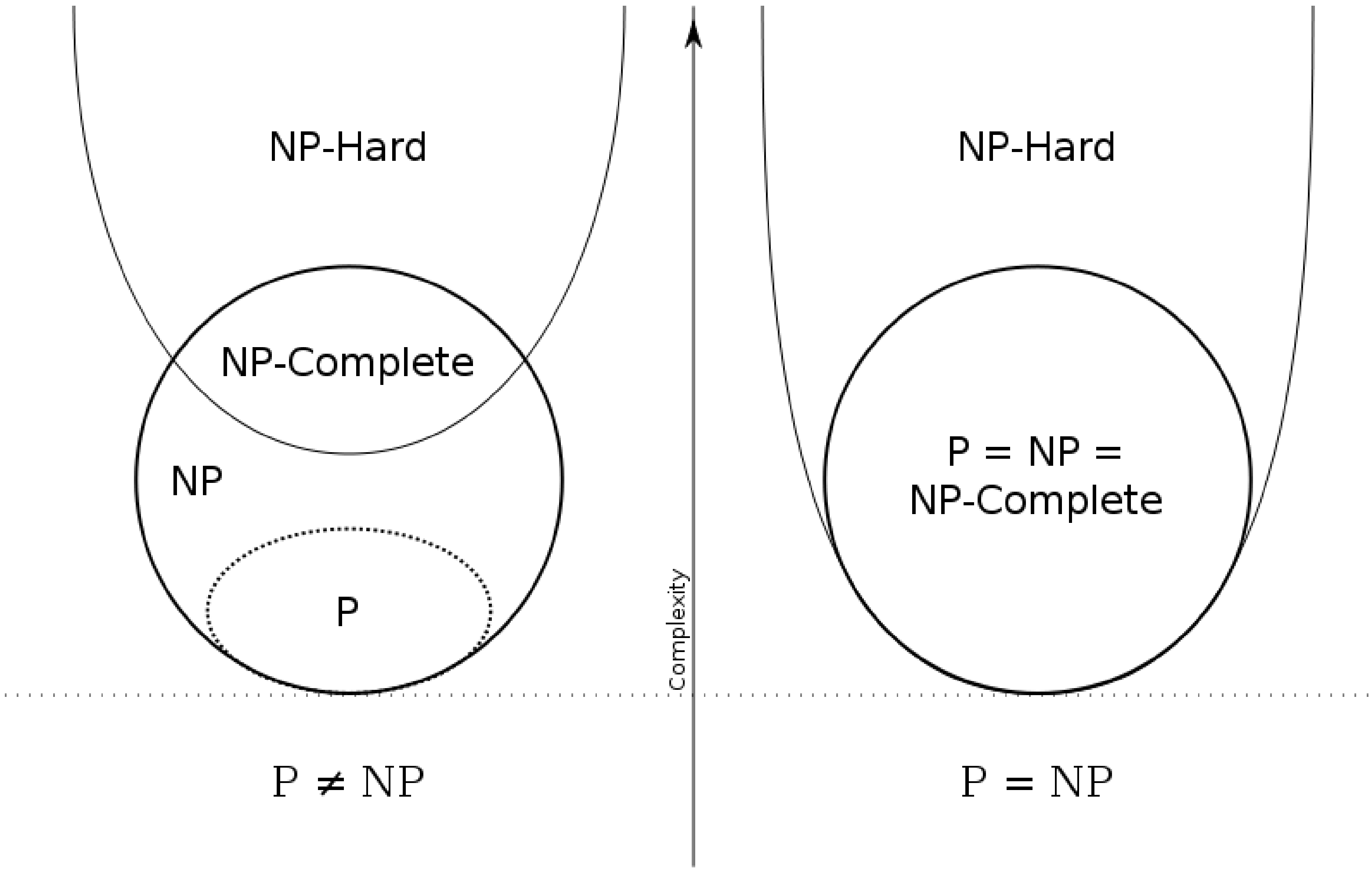
NP-hard vs NP-Complete

A problem *A* is NP-hard if and only if a polynomial-time algorithm for *A* implies a polynomial-time algorithm for every problem in NP.

- NP-hard problems are at least *as hard as* NP-complete problems
- NP-hard includes the optimization version of decision versions
- An NP-hard problem may not be in NP (have no polynomial time verification)

NP-complete

A problem is NP-complete if it is NP-hard and it is in NP


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

Some NP-Complete Problems

Six basic genres of NP-complete problems and paradigmatic examples.

- Packing problems: SET-PACKING, INDEPENDENT SET.
- Covering problems: SET-COVER, VERTEX-COVER.
- Constraint satisfaction problems: SAT, 3-SAT.
- Sequencing problems: HAMILTONIAN-CYCLE, TSP.
- Partitioning problems: 3D-MATCHING, 3-COLOR.
- Numerical problems: SUBSET-SUM, KNAPSACK.

Practice. Most NP problems are either known to be in P or NP-complete.

Notable exceptions. Factoring, graph isomorphism, Nash equilibrium.

Thm [Ladner 75]: If $P \neq NP$ then there are some "NP-intermediate" decision problems $X \in NP$ i.e., $X \notin P$ and X is not NP-Complete.

SUBSET SUM

SUBSET SUM:

Instance: n integers $x_1, \dots, x_n$ and a separate integer k

Question: Does there exist a subset $S \subseteq \{1, \dots, n\}$ s.t. $\sum_{i \in S} x_i = k$

Example 1: (YES Instance)

Instance: $x_1 = 4, x_2 = 5, x_3 = 8, x_4 = 9$ and k=14

Witness: $S = \{2,4\} \rightarrow \sum_{i \in S} x_i = 5 + 9 = 14 = k$

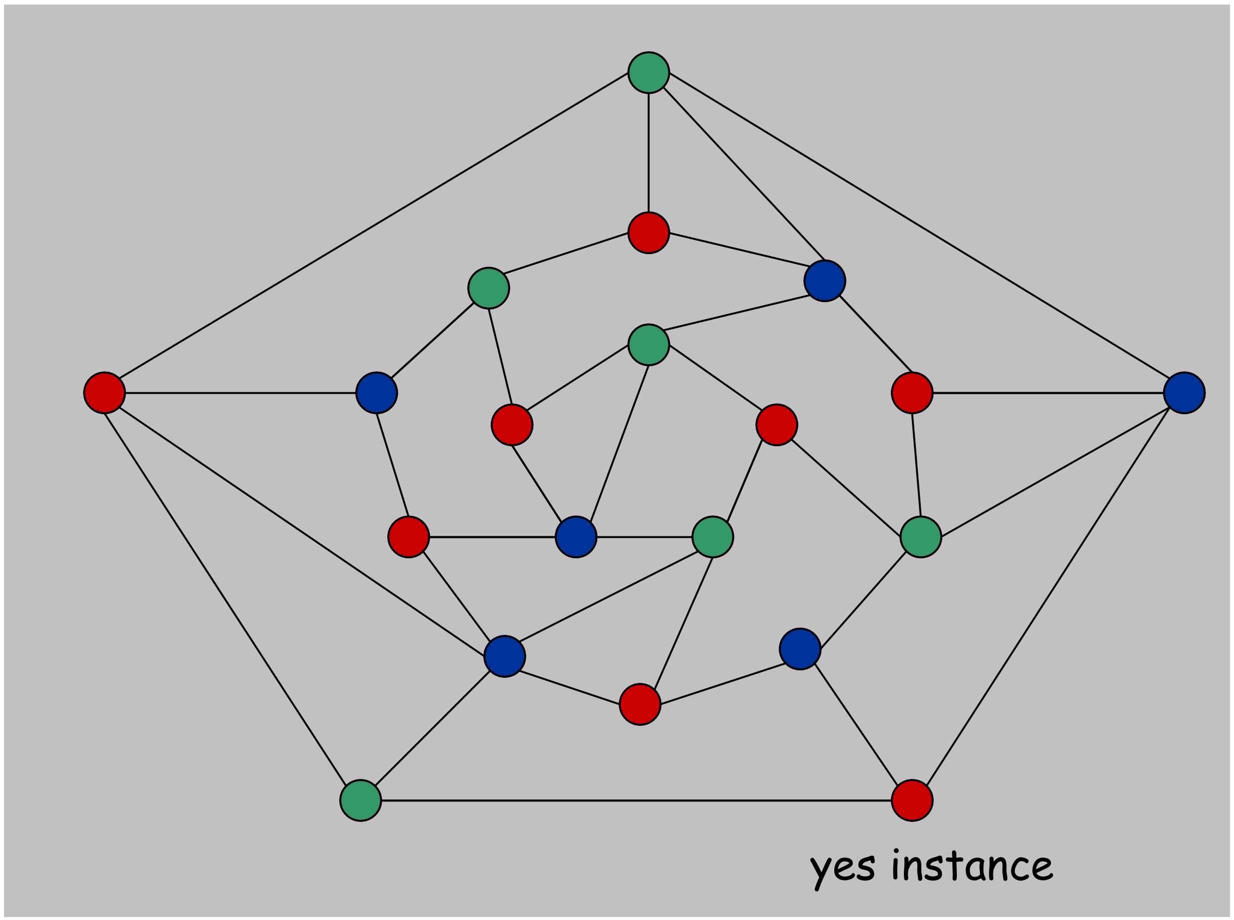
Example 2: (NO Instance)

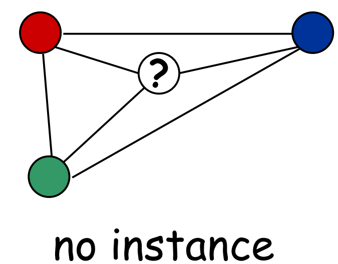
Instance: $x_1 = 4, x_2 = 5, x_3 = 8, x_4 = 9$ and k=15

FACT: SUBSET SUM is NP-Complete

3-Colorability

3-COLOR: Given an undirected graph G does there exists a way to color the nodes red, green, and blue so that no adjacent nodes have the same color?





Register Allocation

Register allocation. Assign program variables to machine register so that no more than k registers are used and no two program variables that are needed at the same time are assigned to the same register.

Interference graph. Nodes are program variables names, edge between u and v if there exists an operation where both u and v are "live" at the same time.

Observation. [Chaitin 1982] Can solve register allocation problem iff interference graph is k-colorable.

Fact. 3-COLOR $\leq_P$ k-REGISTER-ALLOCATION for any constant $k \geq 3$.

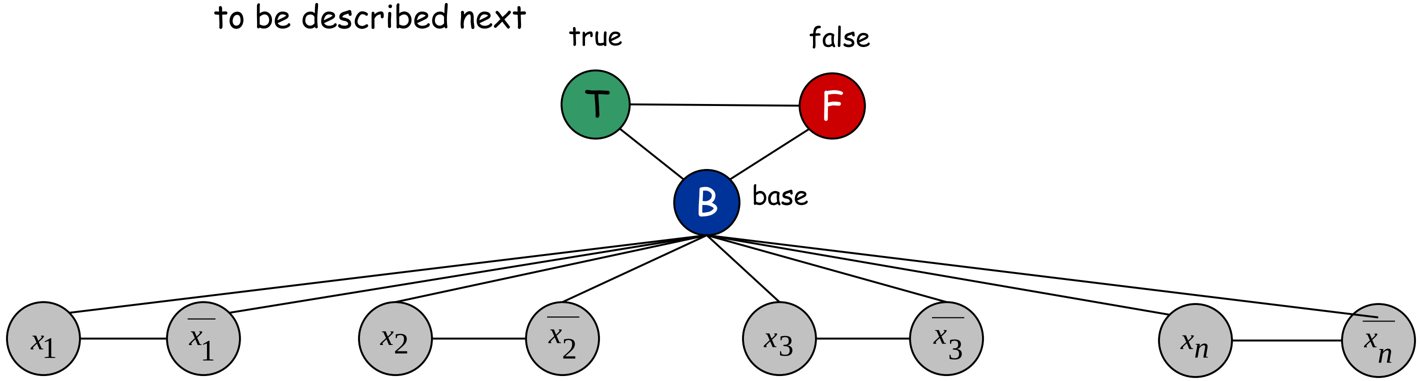
Claim. 3-SAT $\le_P$ 3-COLOR.

Pf. Given 3-SAT instance Φ, we construct an instance G_Φ of 3-COLOR that is 3-colorable iff Φ is satisfiable.

Construction (G_Φ).

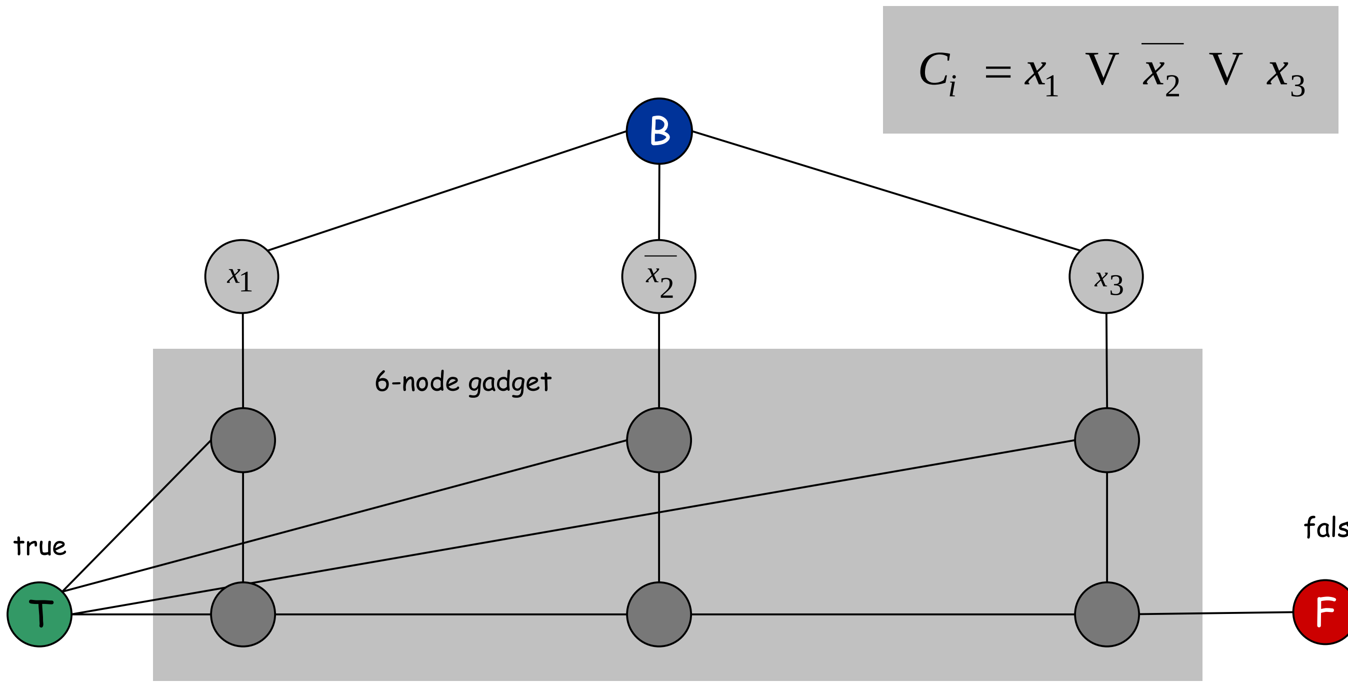
i. For each literal, create a node.
ii. Create 3 new nodes T, F, B; connect them in a triangle, and connect each literal to B.
iii. Connect each literal to its negation.
iv. For each clause, add gadget of 6 nodes and 13 edges.

↑ to be described next



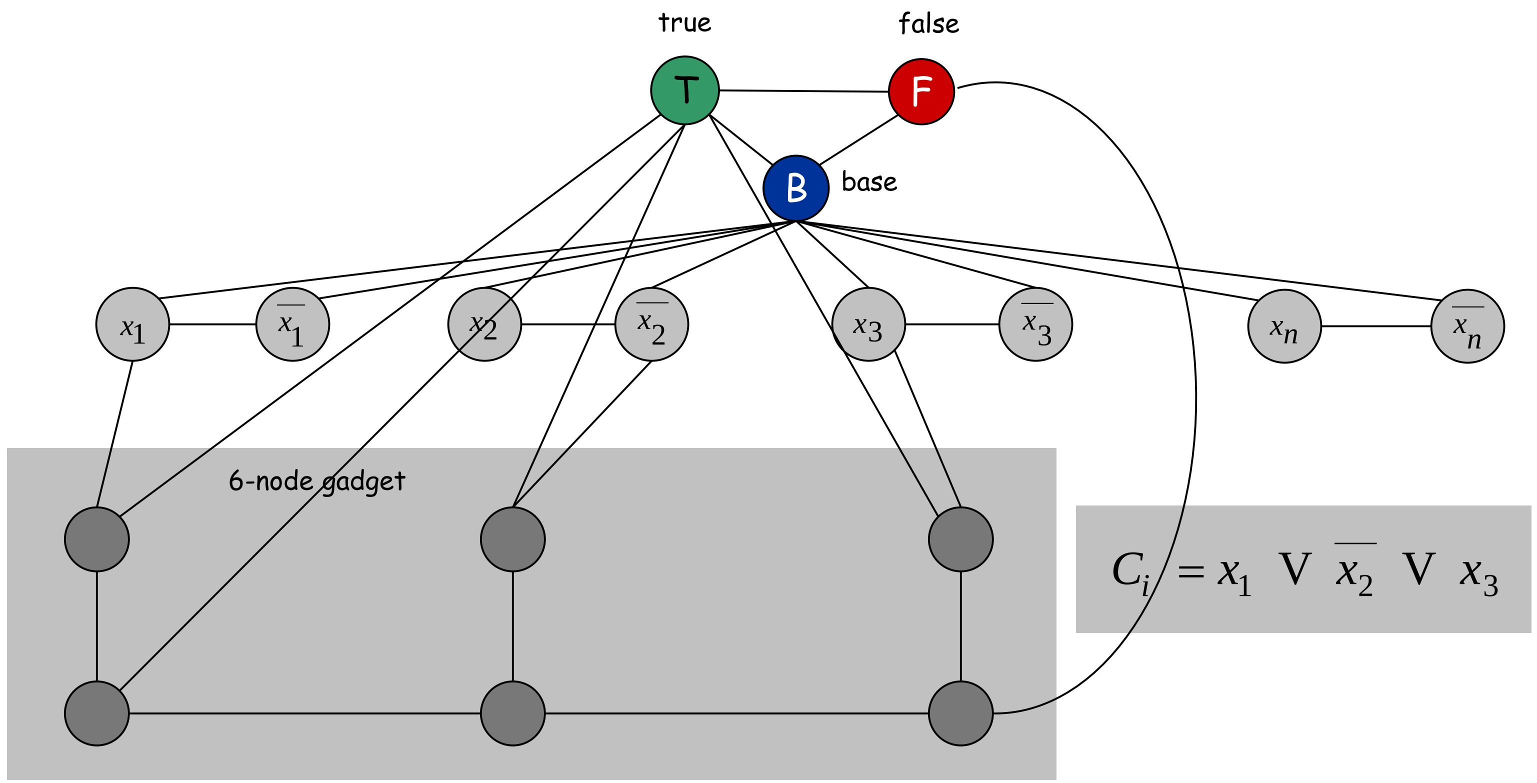
Clause Gadget

For each clause, add gadget of 6 nodes and 13 edges.



Clause Gadget

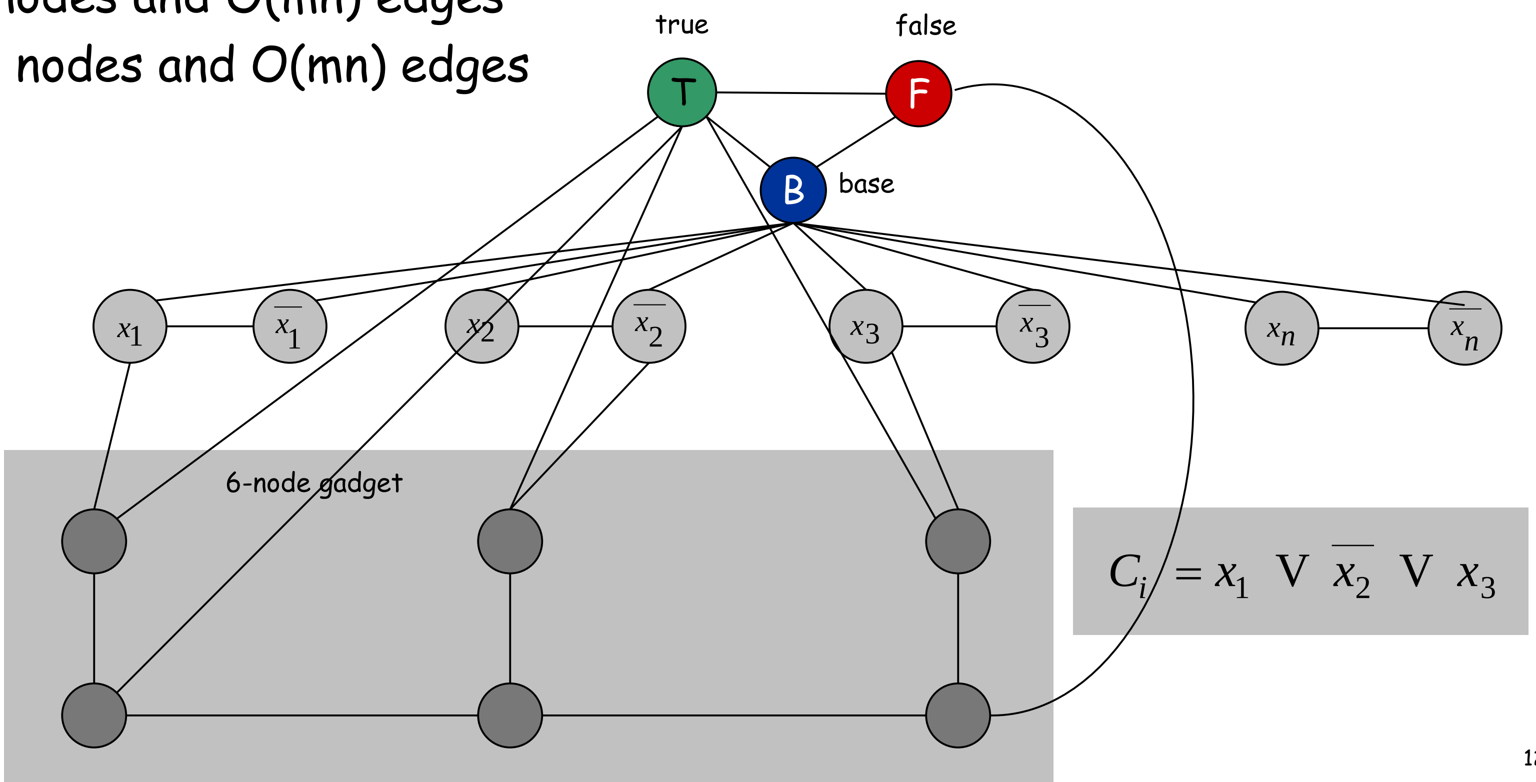
For each clause, add gadget of 6 nodes and 13 edges.



iClicker

Suppose Φ has n variables and m clauses. How many nodes/edges does G_Φ have?

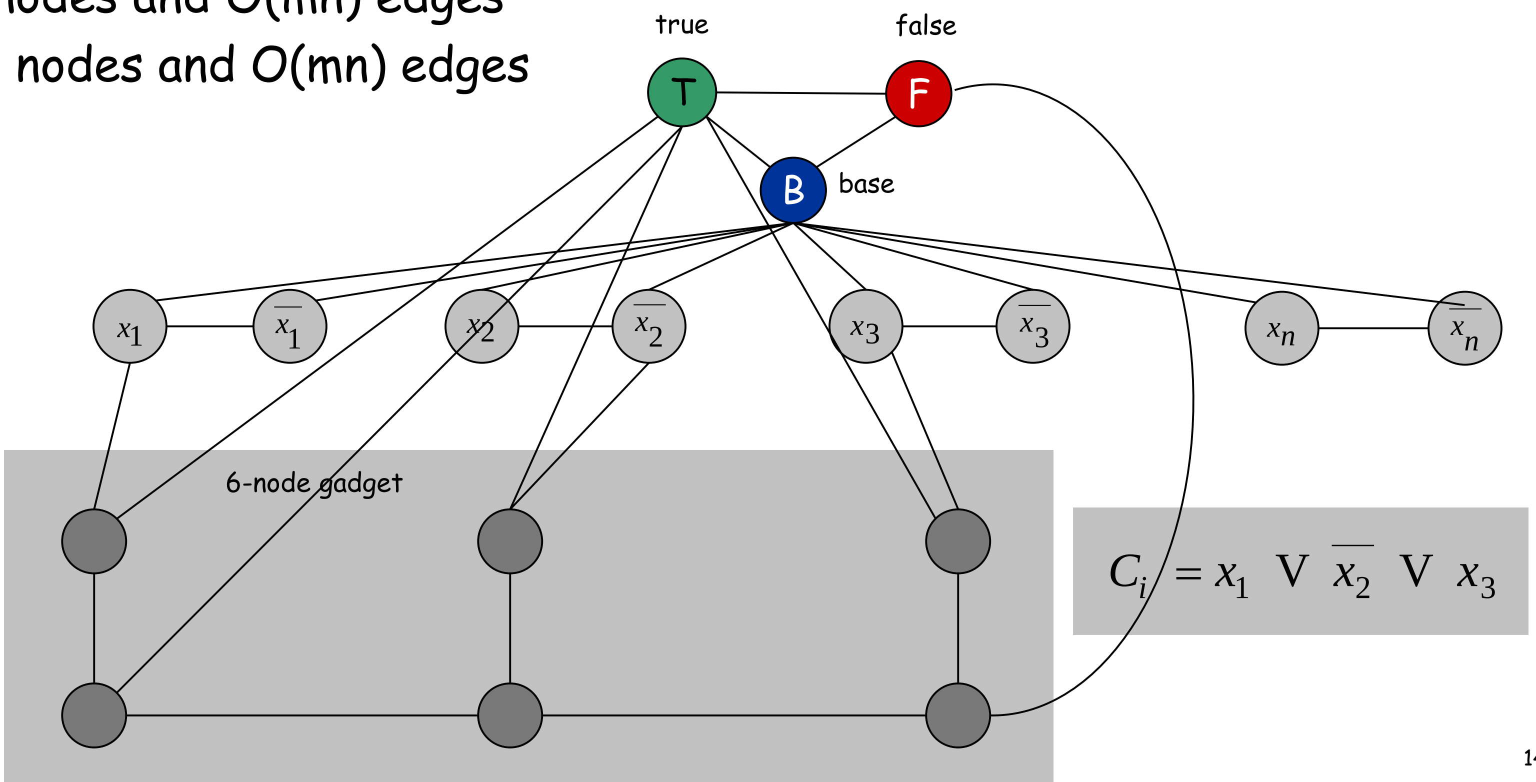
A. O(n) nodes, O(m) edges
B. O(m) nodes, O(n) edges
C. O(m+n) nodes and O(m+n) edges
D. O(m) nodes and O(mn) edges
E. O(mn) nodes and O(mn) edges



iClicker

Suppose Φ has n variables and m clauses. How many nodes/edges does G_Φ have?

A. O(n) nodes, O(m) edges
B. O(m) nodes, O(n) edges
C. O(m+n) nodes and O(m+n) edges
D. O(m) nodes and O(mn) edges
E. O(mn) nodes and O(mn) edges

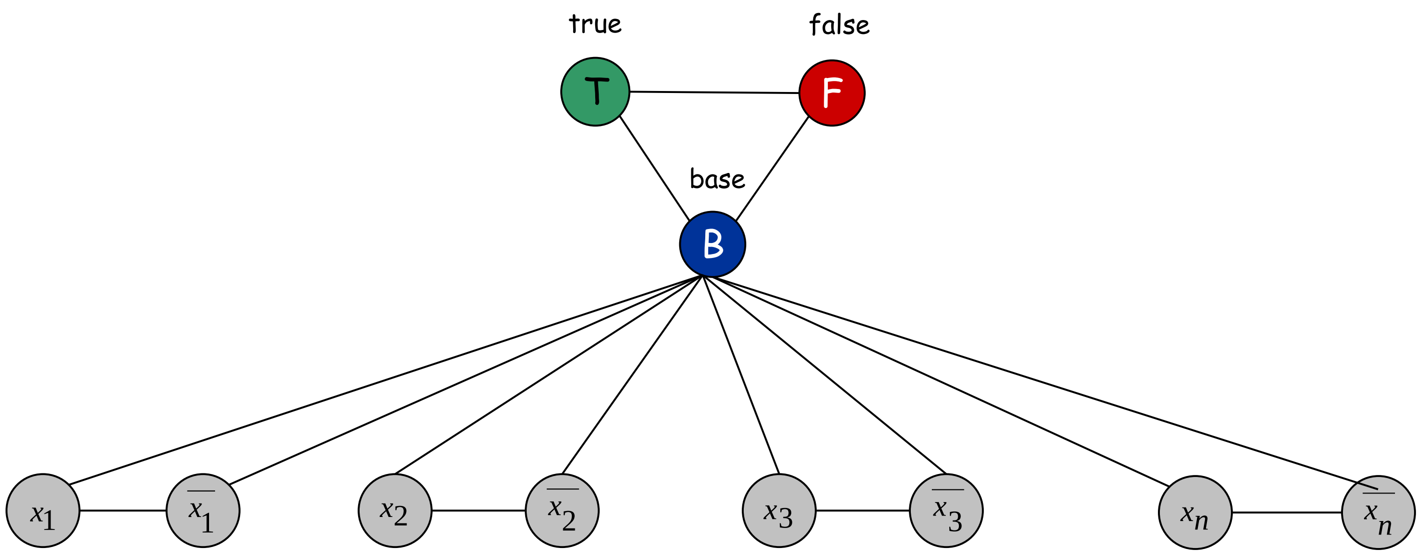


3-Colorability

Claim. Graph G_Φ is 3-colorable iff Φ is satisfiable.

Pf. $\Rightarrow$ Suppose graph G_Φ is 3-colorable.

- Consider assignment that sets all T literals (nodes have same color as T) to true.
- (ii) triangle ensures each literal is T or F (cannot be same color as base).
- (iii) edge between x_i and $\overline{x_i}$ ensures a literal and its negation are have opposite colors.

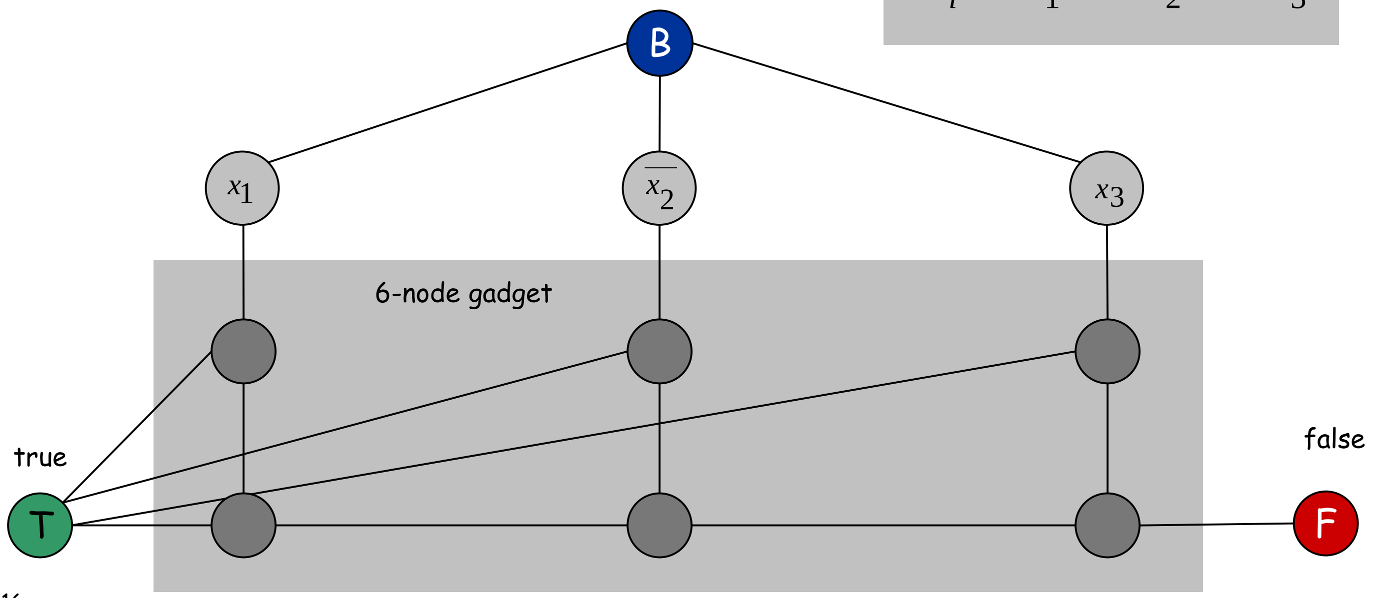


Claim. Graph is 3-colorable iff Φ is satisfiable.

Pf. $\Rightarrow$ Suppose graph is 3-colorable.

- Consider assignment that sets all T literals to true.
- (ii) ensures each literal is T or F.
- (iii) ensures a literal and its negation are opposites.
- (iv) clause gadget ensures at least one literal in each clause is T.

$$C_i = x_1 \ \mathrm{V} \ \overline{x_2} \ \mathrm{V} \ x_3$$



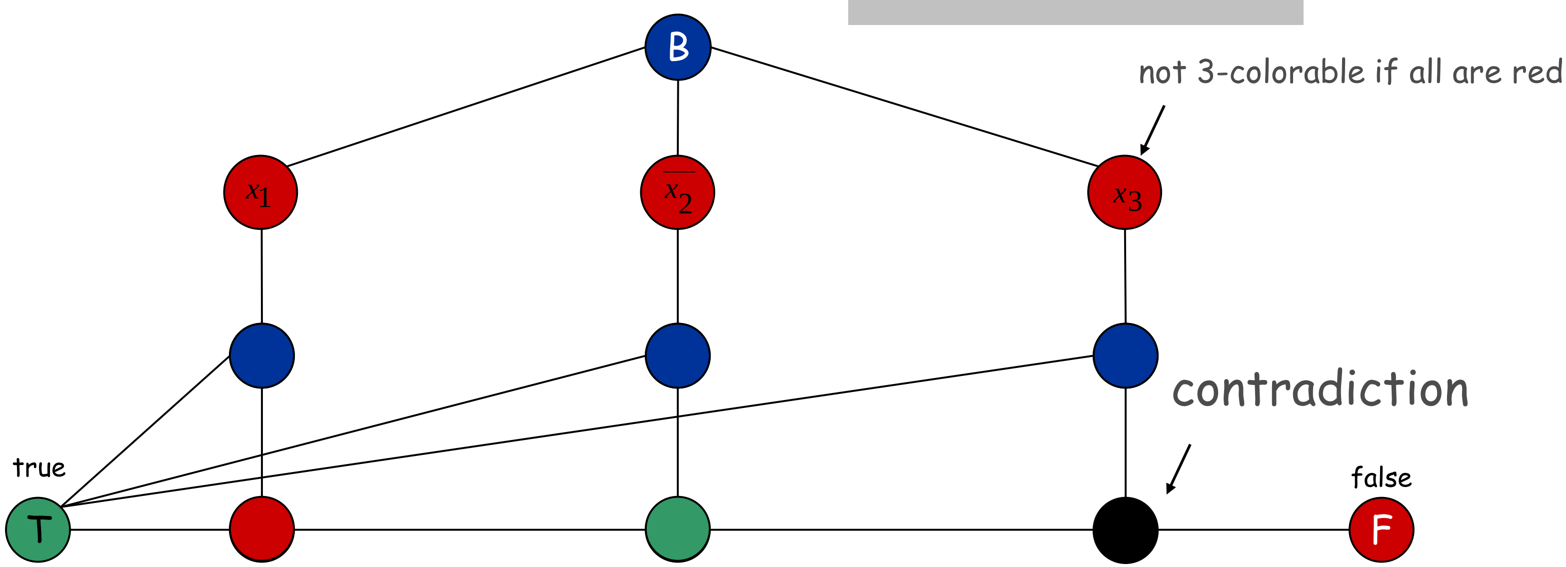
3-Colorability

Claim. Graph is 3-colorable iff Φ is satisfiable.

Pf. $\Rightarrow$ Suppose graph is 3-colorable.

- Consider assignment that sets all T literals to true.
- (ii) ensures each literal is T or F.
- (iii) ensures a literal and its negation are opposites.
- (iv) ensures at least one literal in each clause is T.

$$C_i = x_1 \ \text{V} \ \overline{x_2} \ \text{V} \ x_3$$

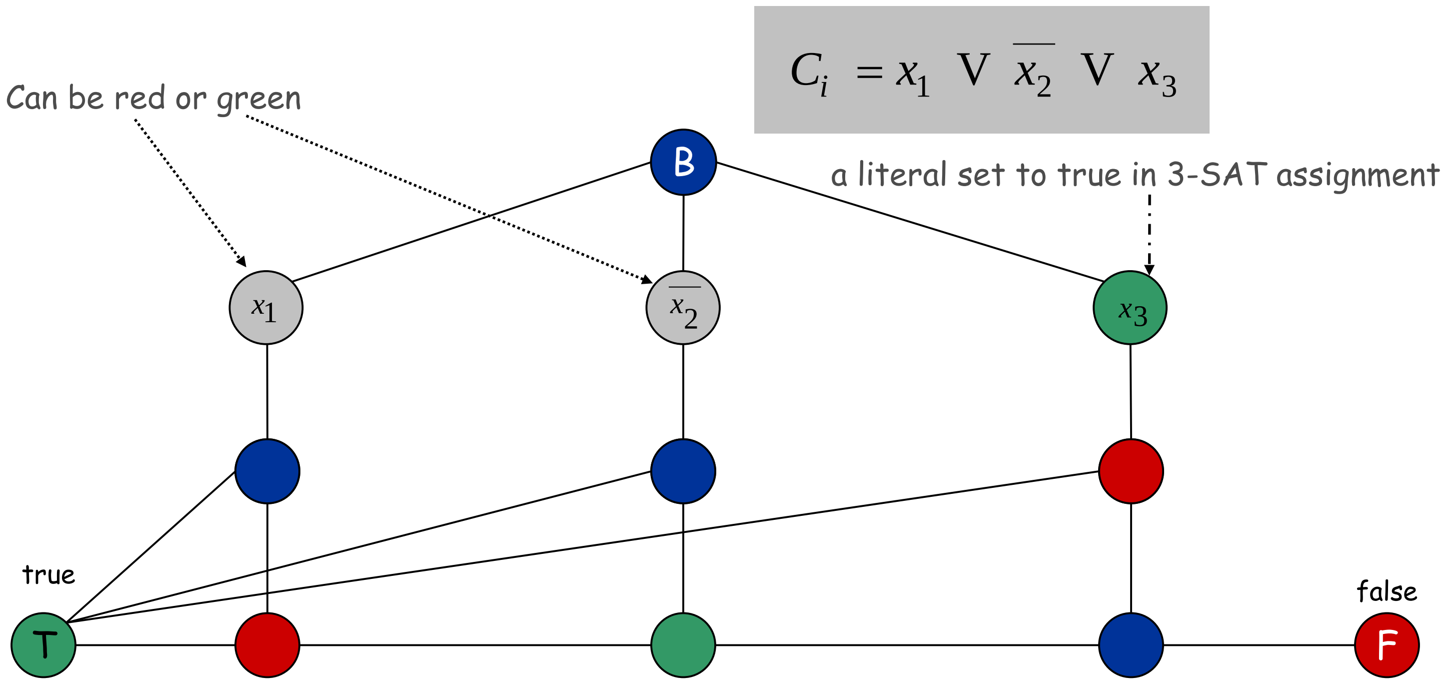


3-Colorability

Claim. Graph is 3-colorable iff Φ is satisfiable.

Pf. $\Leftarrow$ Suppose 3-SAT formula Φ is satisfiable.

- Color all true literals T.
- Color node below green node F, and node below that B.
- Color remaining middle row nodes B.
- Color remaining bottom nodes T or F as forced. ▪



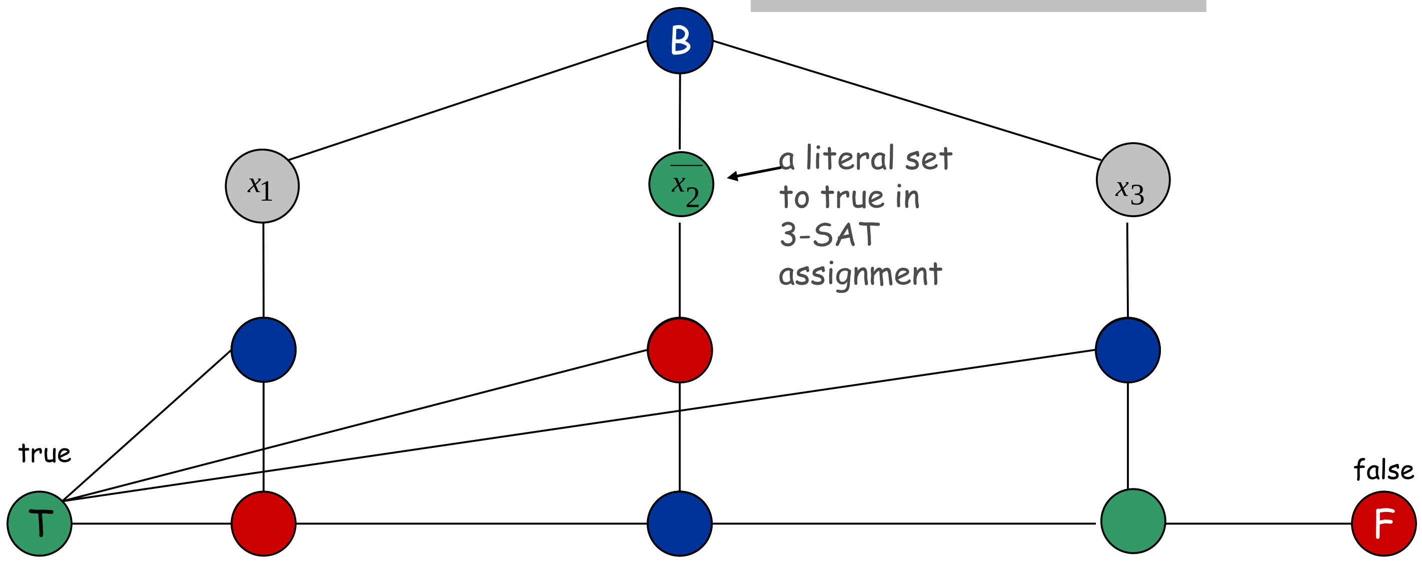
3-Colorability

Claim. Graph is 3-colorable iff Φ is satisfiable.

Pf. $\Leftarrow$ Suppose 3-SAT formula Φ is satisfiable.

- Color all true literals T.
- Color node below green node F, and node below that B.
- Color remaining middle row nodes B.
- Color remaining bottom nodes T or F as forced. ▪

$$C_i = x_1 \vee \overline{x_2} \vee x_3$$



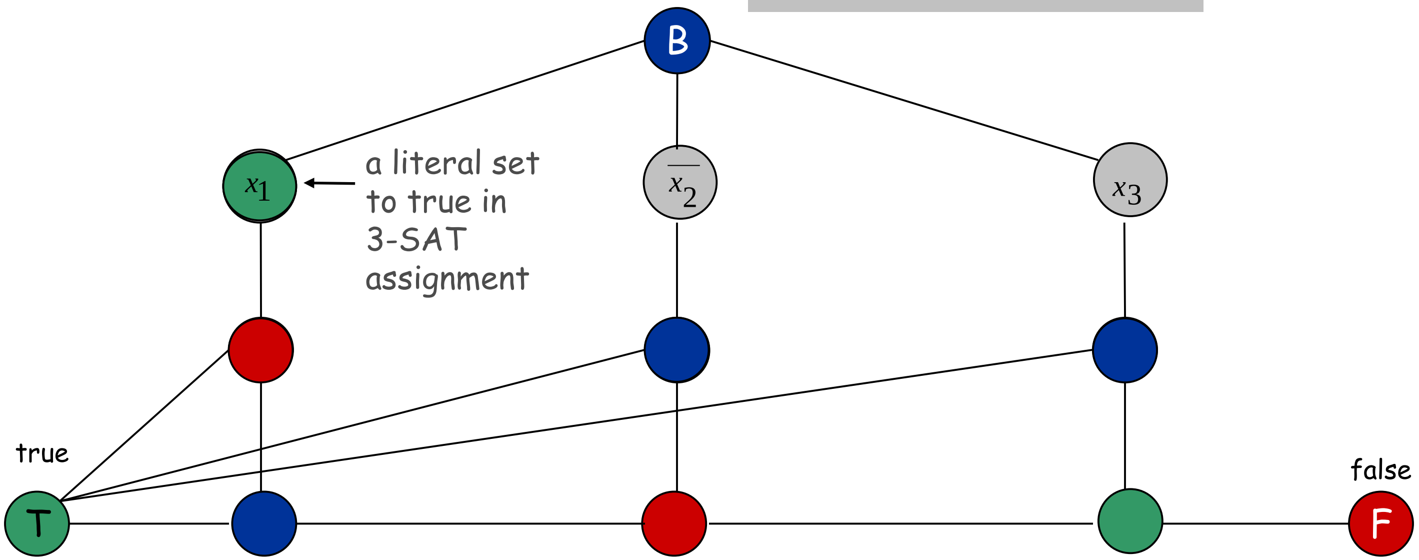
3-Colorability

Claim. Graph is 3-colorable iff Φ is satisfiable.

Pf. $\Leftarrow$ Suppose 3-SAT formula Φ is satisfiable.

- Color all true literals T.
- Color node below green node F, and node below that B.
- Color remaining middle row nodes B.
- Color remaining bottom nodes T or F as forced. ▪

$$C_i = x_1 \vee \overline{x_2} \vee x_3$$



3-Colorability

Claim. 3-SAT $\leq_P$ 3-COLOR.

This shows 3-COLOR is NP-Hard.

Still need to show 3-COLOR is in NP to conclude problem is NP-Complete

Witness: a coloring c(v) in {red, green, blue} for each node v

Certifier: C(G,c)
- For each edge (u,v) check that $c(u) \neq c(v)$
- check that each color c(v) is red, green, blue
- If all checks pass then output ACCEPT; otherwise REJECT

Claim 1: if G is three colorable then there exists a witness c tha the certifier accepts i.e., C(G,c)=ACCEPT.

Claim 2: if G is not three colorable then C(G,c)=REJECT for all witnesses t

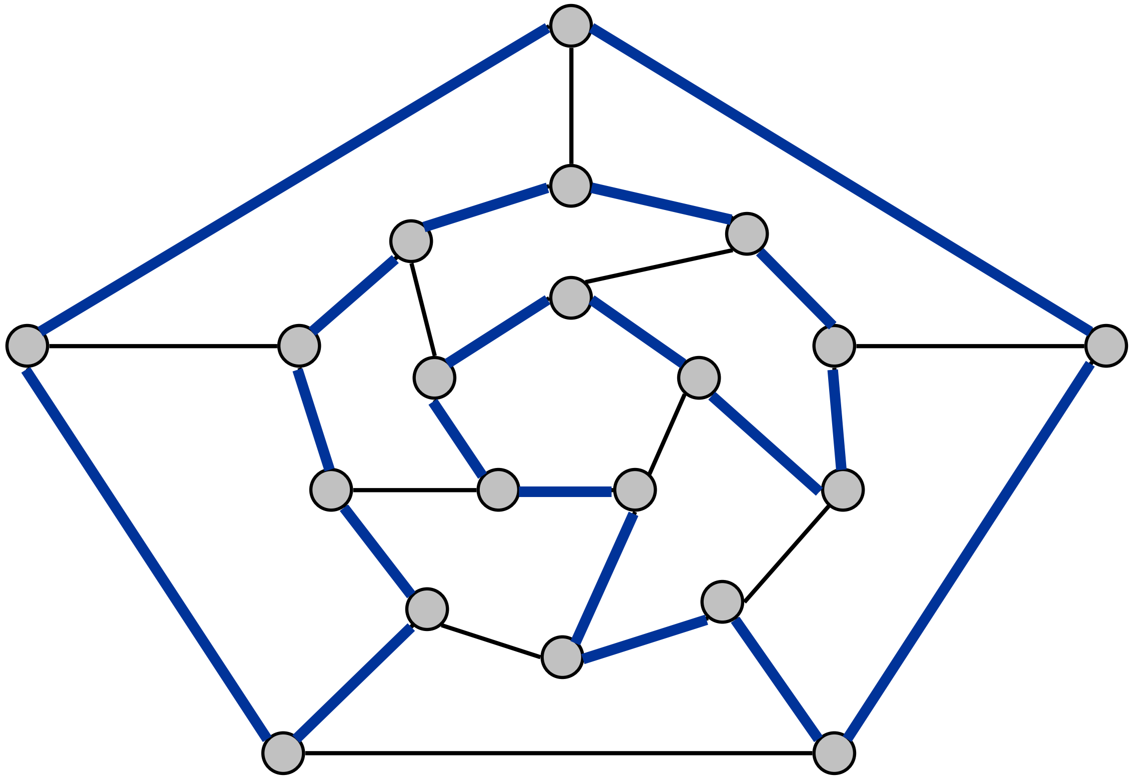
8.5 Sequencing Problems

Basic genres.

- Packing problems: SET-PACKING, INDEPENDENT SET.
- Covering problems: SET-COVER, VERTEX-COVER.
- Constraint satisfaction problems: SAT, 3-SAT.
- Sequencing problems: HAMILTONIAN-CYCLE, TSP.
- Partitioning problems: 3D-MATCHING, 3-COLOR.
- Numerical problems: SUBSET-SUM, KNAPSACK.

Hamiltonian Cycle

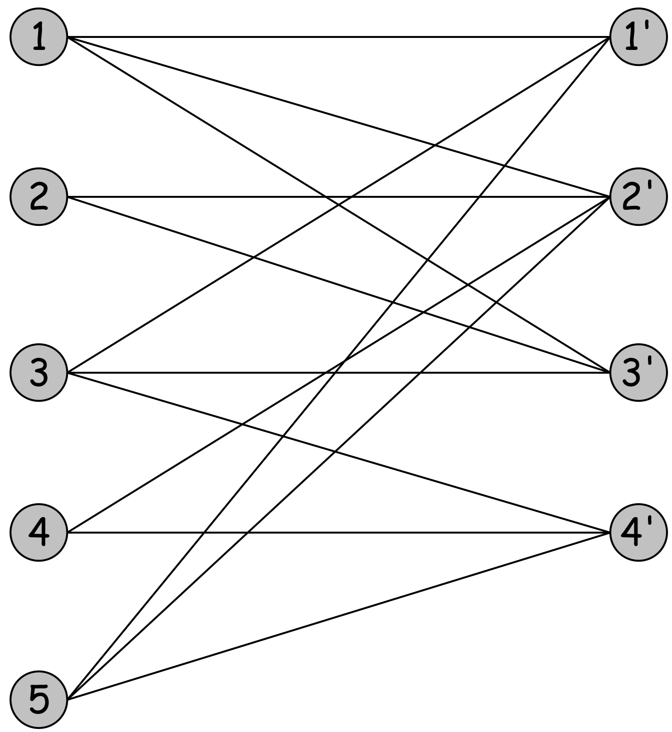
HAM-CYCLE: given an undirected graph G = (V, E), does there exist a simple cycle Γ that contains every node in V.



YES: vertices and faces of a dodecahedron.

Hamiltonian Cycle

HAM-CYCLE: given an undirected graph G = (V, E), does there exist a simple cycle Γ that contains every node in V.



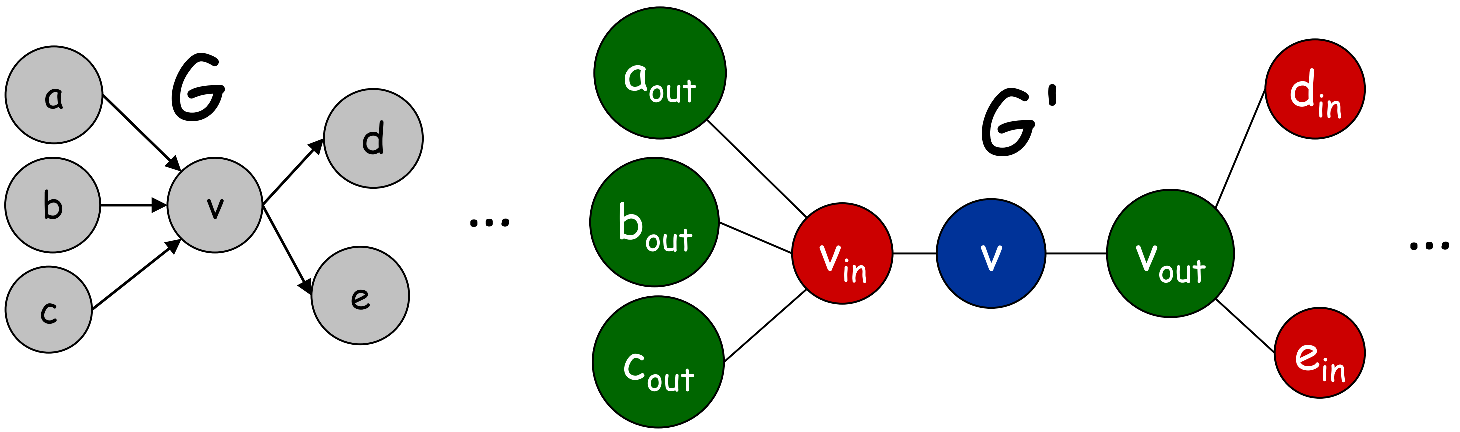
NO: bipartite graph with odd number of nodes.

Directed Hamiltonian Cycle

DIR-HAM-CYCLE: given a digraph $G = (V, E)$, does there exists a simple directed cycle Γ that contains every node in V?

Claim. DIR-HAM-CYCLE $\leq_P$ HAM-CYCLE.

Pf. Given a directed graph $G = (V, E)$, construct an undirected graph G' with 3n nodes.

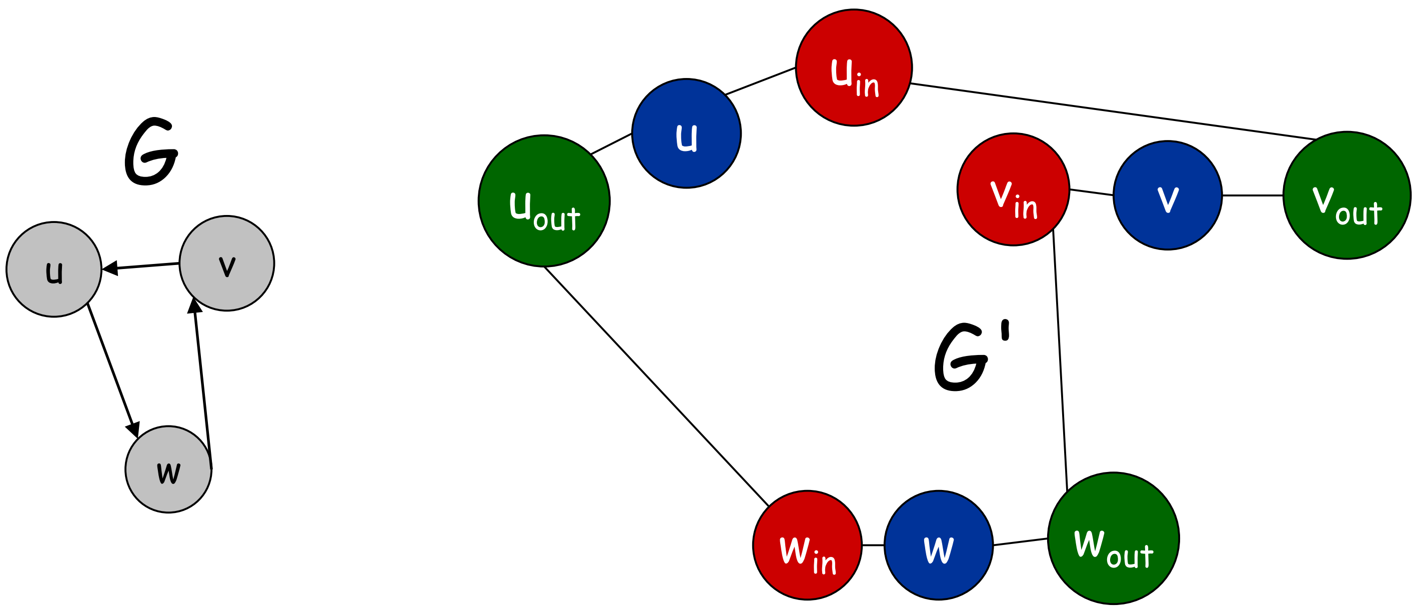


Directed Hamiltonian Cycle

Claim. G has a Hamiltonian cycle iff G' does.

Pf. $\Rightarrow$

- Suppose G has a directed Hamiltonian cycle Γ (e.g., (u,w,v)).
- Then G' has an undirected Hamiltonian cycle (same order).
  - For each node v in directed path cycle replace v with v_{in},v,v_{out}



Directed Hamiltonian Cycle

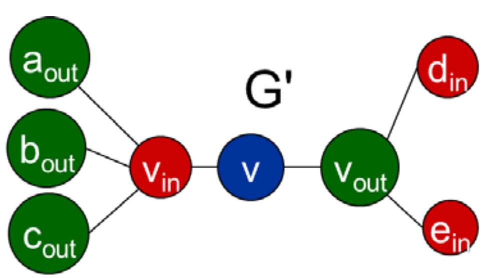
Claim. G has a Hamiltonian cycle iff G' does.

Pf. $\Rightarrow$

- Suppose G has a directed Hamiltonian cycle Γ.
- Then G' has an undirected Hamiltonian cycle (same order).
  - For each node v in directed path cycle replace v with v_{in},v,v_{out}

Pf. $\Leftarrow$

- Suppose G' has an undirected Hamiltonian cycle Γ'.
- Γ' must visit nodes in G' using one of following two orders:
  …, B, G, R, B, G, R, B, G, R, B, …
  …, B, R, G, B, R, G, B, R, G, B, …
- Blue nodes in Γ' make up directed Hamiltonian cycle Γ in G, or reverse of one. ▪



More Hard Computational Problems

Aerospace engineering: optimal mesh partitioning for finite elements.
Biology: protein folding.
Chemical engineering: heat exchanger network synthesis.
Civil engineering: equilibrium of urban traffic flow.
Economics: computation of arbitrage in financial markets with friction.
Electrical engineering: VLSI layout.
Environmental engineering: optimal placement of contaminant sensors.
Financial engineering: find minimum risk portfolio of given return.
Game theory: find Nash equilibrium that maximizes social welfare.
Genomics: phylogeny reconstruction.
Mechanical engineering: structure of turbulence in sheared flows.
Medicine: reconstructing 3-D shape from biplane angiocardiogram.
Operations research: optimal resource allocation.
Physics: partition function of 3-D Ising model in statistical mechanics.
Politics: Shapley-Shubik voting power.
Pop culture: Minesweeper consistency.
Statistics: optimal experimental design.

Numerical Problems

Basic genres.

- Packing problems: SET-PACKING, INDEPENDENT SET.
- Covering problems: SET-COVER, VERTEX-COVER.
- Constraint satisfaction problems: SAT, 3-SAT.
- Sequencing problems: HAMILTONIAN-CYCLE, TSP.
- Partitioning problems: 3D-MATCHING, 3-COLOR.
- Numerical problems: SUBSET-SUM, KNAPSACK.

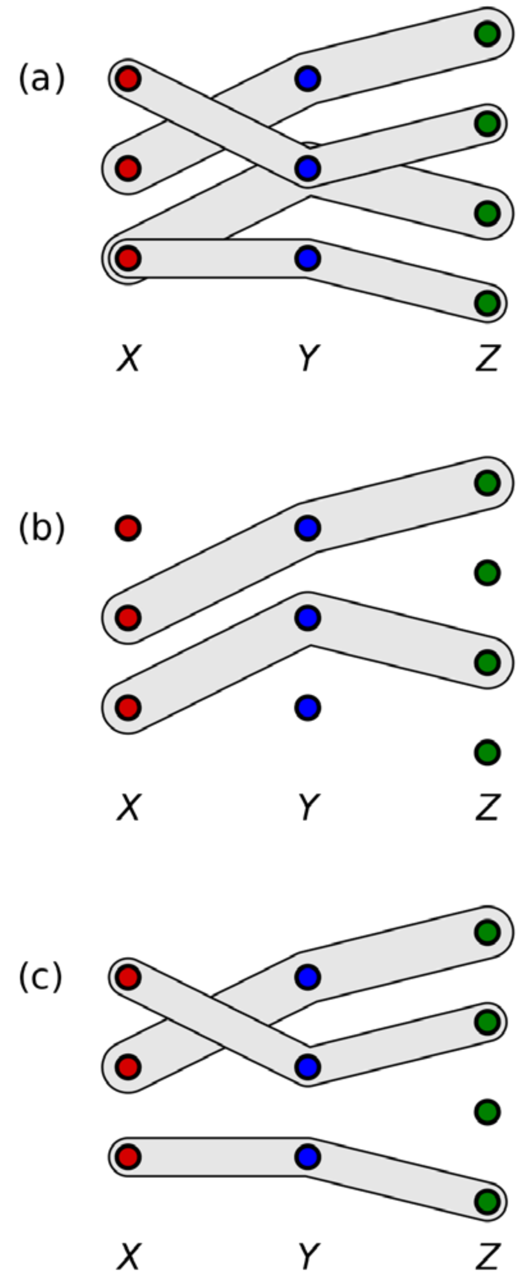
3D-Matching

Input: X ∪ Y ∪ Z, subset T ⊆ $X \times Y \times Z$ of triples (hyperedges) and an integer k.

Output: YES if there is a 3Dmatching of size k, NO otherwise

Matching: M ⊆ T.
- Each node v in X ∪ Y ∪ Z is contained in *at most* one triple in M

Fact: 3D-Matching is NP-Complete



Subset Sum (proof from book)

Construction. Let $X \cup Y \cup Z$ be a instance of 3D-MATCHING with triplet set T. Let $n = |X| = |Y| = |Z|$ and $m = |T|$.

- Let $X = \{ x_1, x_2, x_3\ x_4 \}$, $Y = \{ y_1, y_2, y_3, y_4 \}$, $Z = \{ z_1, z_2, z_3, z_4 \}$
- For each triplet $t = (x_i, y_j, z_k) \in T$, create an integer w_t with $3n$ digits that has a 1 in positions i, $n+j$, and $2n+k$.

use base m+1

Claim. 3D-matching iff some subset sums to W = 111,…, 111.

Triplet t_i			x_1	x_2	x_3	x_4	y_1	y_2	y_3	y_4	z_1	z_2	z_3	z_4	w_i
x_1	y_2	z_3	1	0	0	0	0	1	0	0	0	0	1	0	100,001,000,010
x_2	y_4	z_2	0	1	0	0	0	0	0	1	0	1	0	0	10,000,010,100
x_1	y_1	z_1	1	0	0	0	1	0	0	0	1	0	0	0	100,010,001,000
x_2	y_2	z_4	0	1	0	0	0	1	0	0	0	0	0	1	10,001,000,001
x_4	y_3	z_4	0	0	0	1	0	0	1	0	0	0	0	1	100,100,001
x_3	y_1	z_2	0	0	1	0	1	0	0	0	0	1	0	0	1,010,000,100
x_3	y_1	z_3	0	0	1	0	1	0	0	0	0	0	1	0	1,010,000,010
x_3	y_1	z_1	0	0	1	0	1	0	0	0	1	0	0	0	1,010,001,000
x_4	y_4	z_4	0	0	0	1	0	0	0	1	0	0	0	1	100,010,001
															111,111,111,111

Partition

SUBSET-SUM. Given natural numbers $w_1, \ldots, w_n$ and an integer W, is there a subset that adds up to exactly W?

PARTITION. Given natural numbers $v_1, \ldots, v_m$, can they be partitioned into two subsets that add up to the same value?

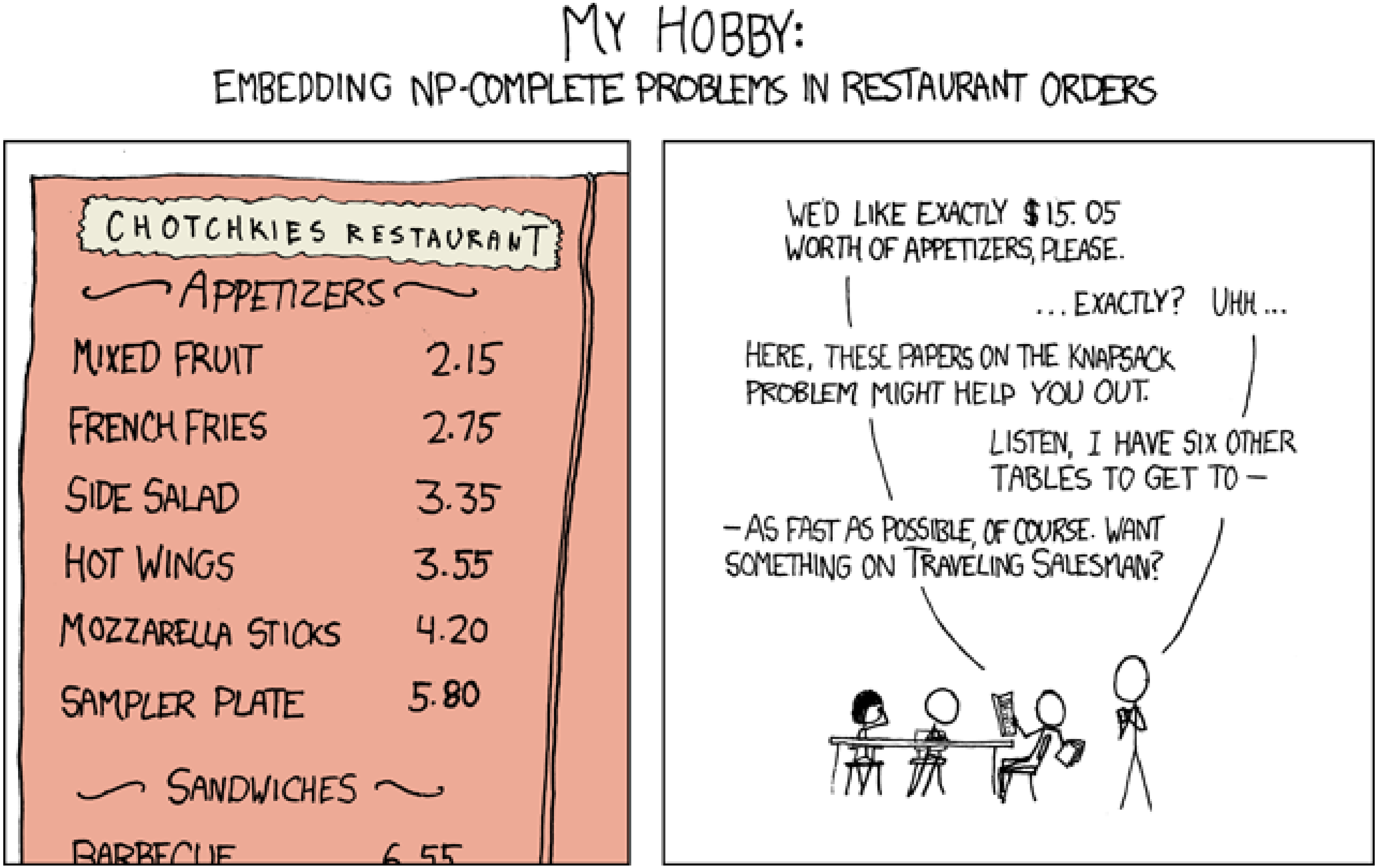
↖ $\frac{1}{2} \Sigma_i v_i$

Claim. SUBSET-SUM $\leq_P$ PARTITION.

Pf. Let $W, w_1, \ldots, w_n$ be an instance of SUBSET-SUM.

- Create instance of PARTITION with $m = n+2$ elements.
  - $v_1 = w_1, v_2 = w_2, \ldots, v_n = w_n, \quad v_{n+1} = 2 \Sigma_i w_i - W, \quad v_{n+2} = \Sigma_i w_i + W$

- There exists a subset that sums to W iff there exists a partition since two new elements cannot be in the same partition. ▪



Randall Munro
http://xkcd.com/c287.html