CS4234
Optimiz(s)ation Algorithms

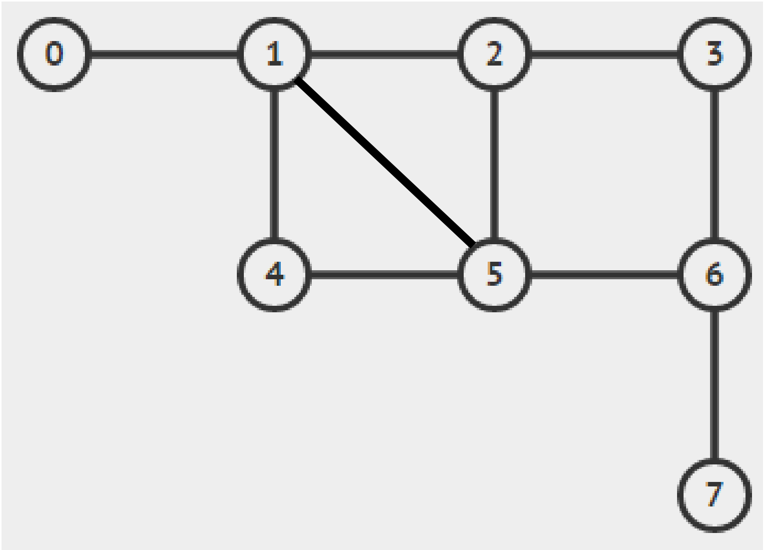
L1 – `Min-Vertex-Cover`

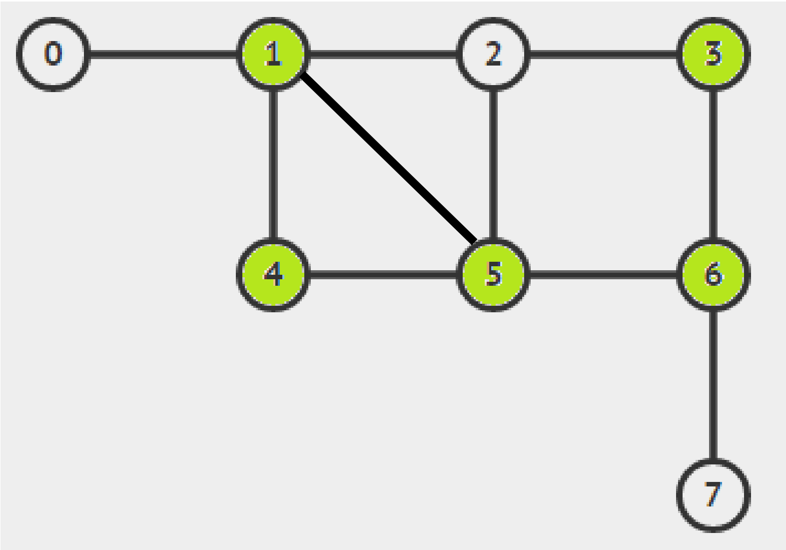
[**https://visualgo.net/en/mvc**](https://visualgo.net/en/mvc)

Vertex Cover

Definition:

– A vertex cover for a graph $G = (V, E)$ is a set $S \subseteq V$ such that for every edge $e = (u, v) \in E$, either $u \in S$ or $v \in S$



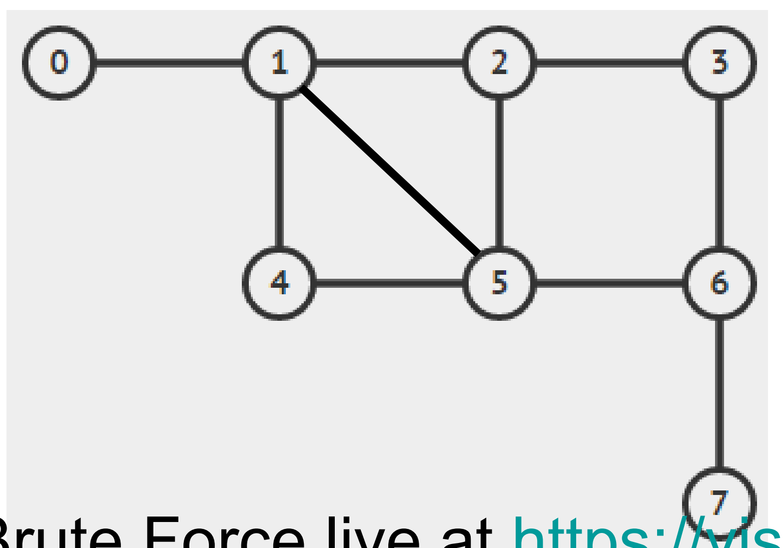


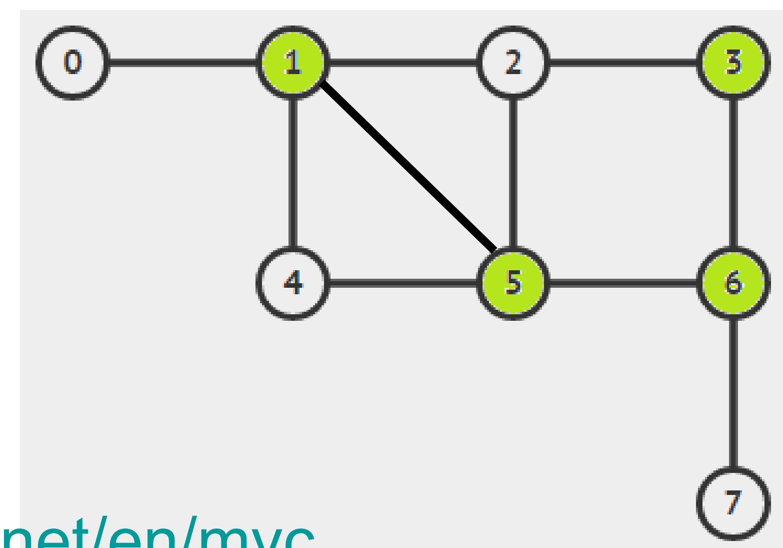
Min-Vertex-Cover (MVC)

Definition:

- Given a graph G = (V, E), find a **minimum-sized** set S that is a vertex cover for G

Analogy: A certain coffee brand in Singapore





Try Brute Force live at https://visualgo.net/en/mvc

`Vertex-Cover` is NP-Complete

The **decision** version

- Given a graph G = (V, E) and a parameter k, does there exist a `Vertex-Cover` of G of size k (vertices)?

Proof:

- `Vertex-Cover` is in NP
- `Vertex-Cover` is NP-hard
  - `Clique` $\leq_p$ `Vertex-Cover`
- See revision slides from CS3230 (copied here)

1. **VERTEX-COVER ∈ NP**

Input: An undirected graph **G = (V, E)** and an integer **k**

Certificate: A subset **V'** of size **k**

The **O(V+E)** verification algorithm checks:

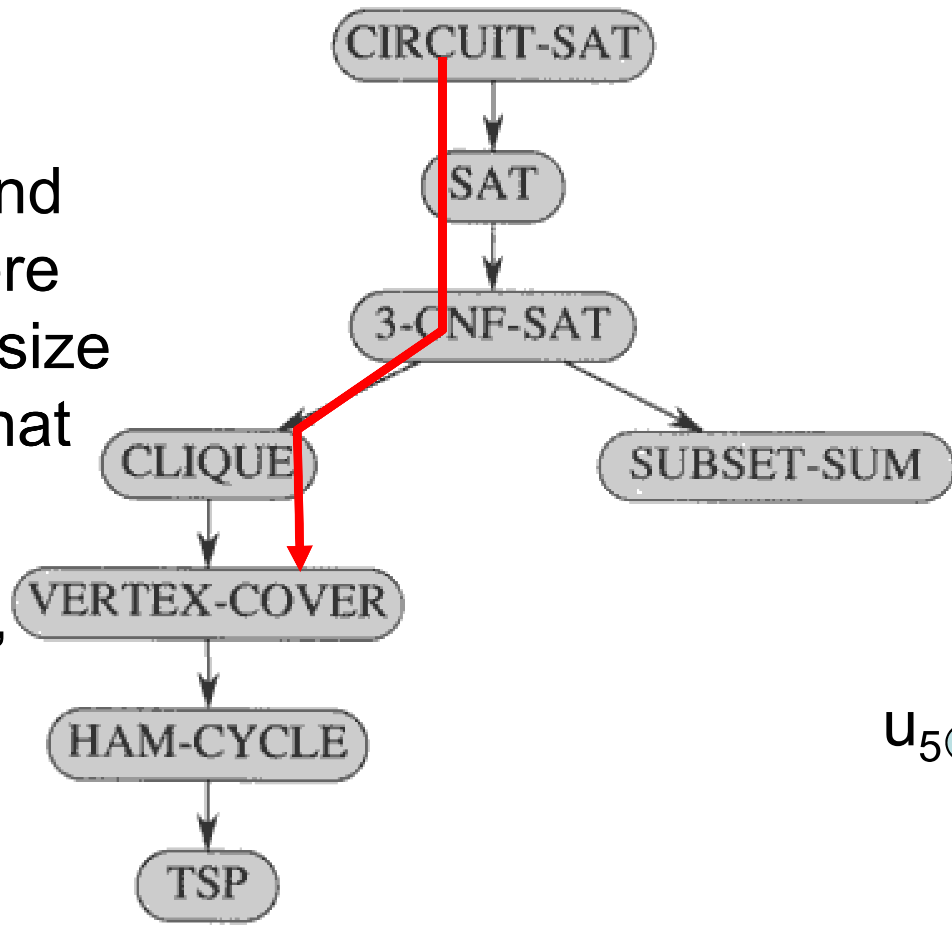
- if **|V'| = k** and insert those vertices into a ___________ (**O(1)** per that data structure insertion, so **O(V)** overall)
- Then, it scans all edge **(u, v)** ∈ **E** to check if at least one of **u** and **v** belongs to **V'** (**O(1)** per that data structure check, so **O(E)** overall)

Therefore, **Vertex Cover is in NP**

Agenda for showing more NP-complete problems

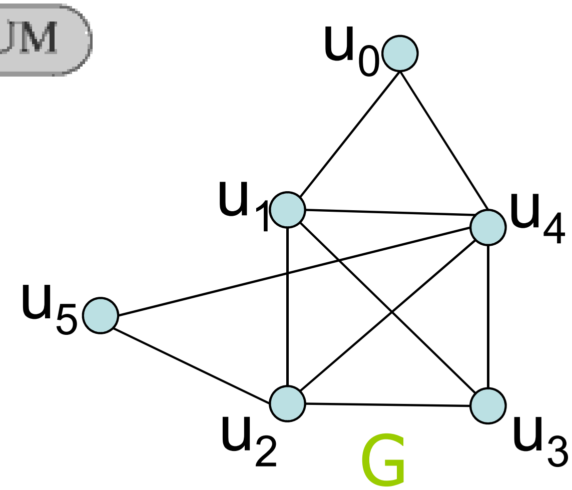
Using polynomial time reduction,
we can obtain **more** NP-complete problems

CLIQUE: Given a graph **G = (V, E)** and an integer **k**, is there a subset **C** ⊆ **V** of size **k** (vertices) such that **C** is a clique in **G**?



For now, "assume" that CLIQUE has been proven to be NP-hard

Eyeball: Is there a clique of size k = 4 below?



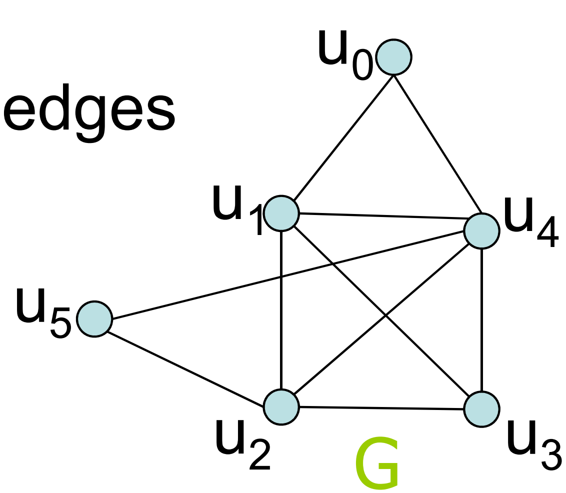
2. CLIQUE$\leq_p$VERTEX-COVER (1)

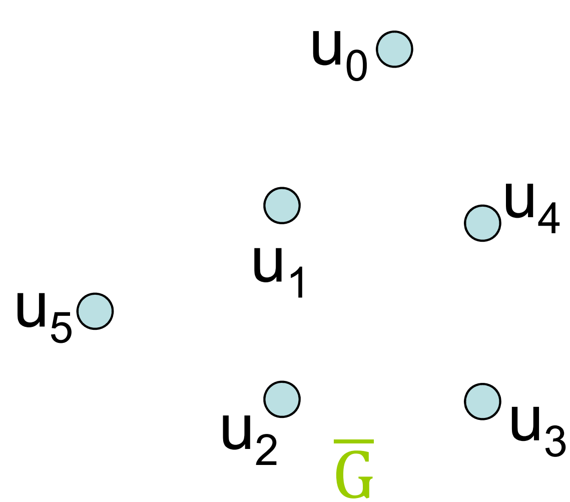
Given an undirected graph **G = (V, E)** and **k** (note: **n = |V|**), we construct a graph $\bar{G}$ **= (V,** $\bar{E}$**)** where **(u, v)** $\in$ $\bar{E}$ iff **(u, v)** $\notin$ **E**

Claim: G has a size-**k** clique iff $\bar{G}$ has a size-**(n-k)** vertex cover

Exercise: Draw the complement graph $\bar{G}$ **in O(V²)**

Graph **G**
6 vertices,
10 edges
K_6 has 15 edges



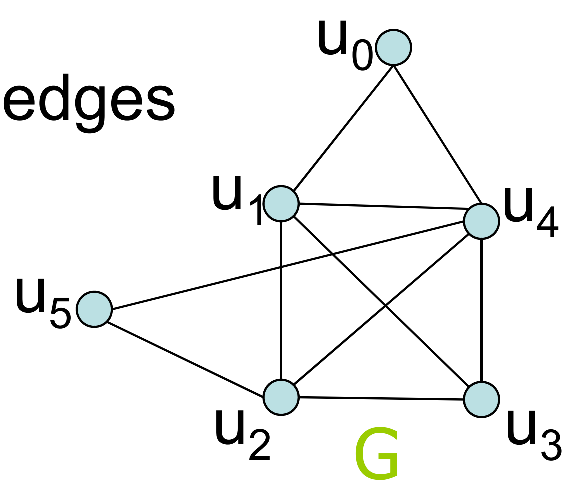


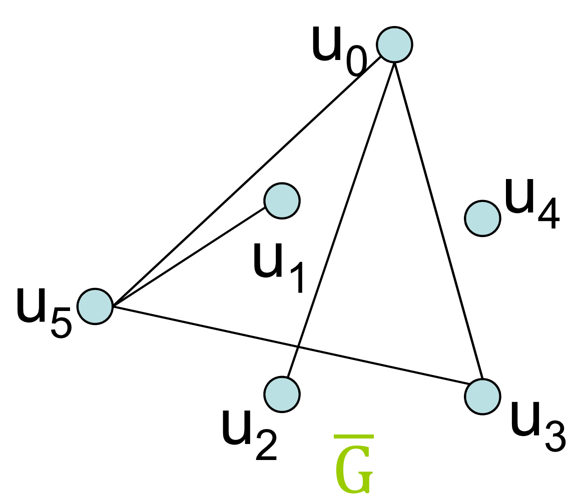
2. CLIQUE$\leq_p$VERTEX-COVER (2)

Given an undirected graph **G = (V, E)** and **k** (note: **n = |V|**), we construct a graph $\bar{G}$ **= (V, $\bar{E}$)** where **(u, v)** ∈ $\bar{E}$ iff **(u, v)** ∉ **E**

Claim: G has a size-**k** clique iff $\bar{G}$ has a size-**(n-k)** vertex cover

Graph **G**
6 vertices,
10 edges
K_6 has 15 edges





Complement graph $\bar{G}$
6 vertices,
5 edges

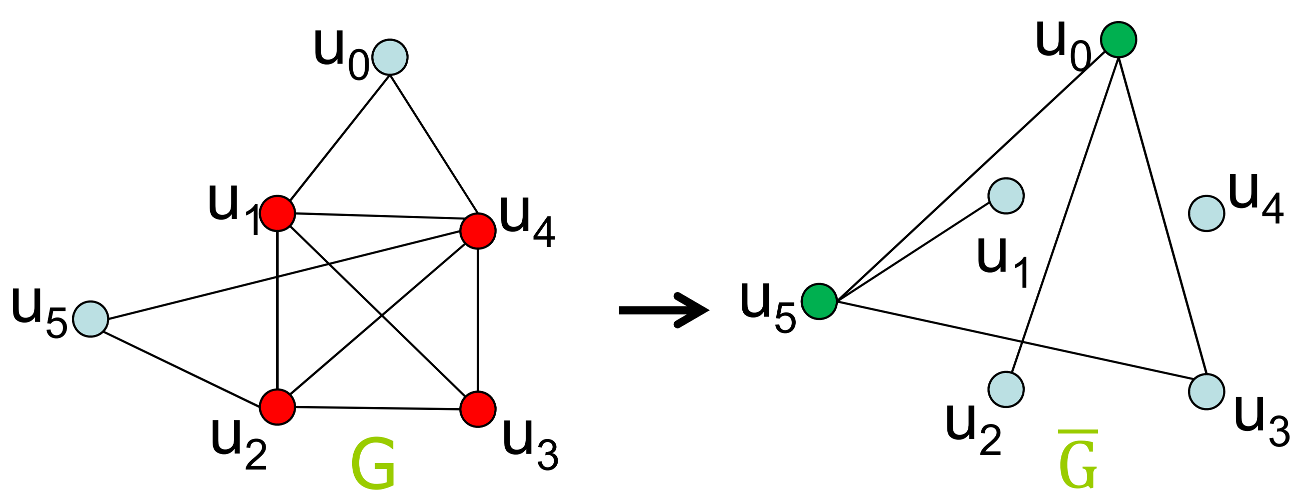
G has a size-**k** clique → $\overline{G}$ has a size-**(n-k)** vertex cover

Suppose V' ⊆ V is a size-**k** clique of G,
e.g., V' = {u_1, u_2, u_3, u_4} is a size-**4** clique of G

- For any arbitrary edge (u, v) ∈ $\bar{E}$ in the complement graph $\overline{G}$, then (u, v) ∉ E
  - Which implies that at least one of u or v does not belong to a clique V' as **every pair** of vertices in V' are connected by an edge in E and thus won't be in $\bar{E}$

Hence, at least one of u and v belongs to V-V' (of size **n-k**),
e.g., V-V' = {u_0, u_5} is a size-**2** vertex cover of $\overline{G}$

- Since edge (u, v) ∈ $\bar{E}$ was chosen arbitrarily, every edge (u, v) ∈ $\bar{E}$ is covered by a vertex in V-V', so V-V' (of size **n-k**) is a VC of $\overline{G}$

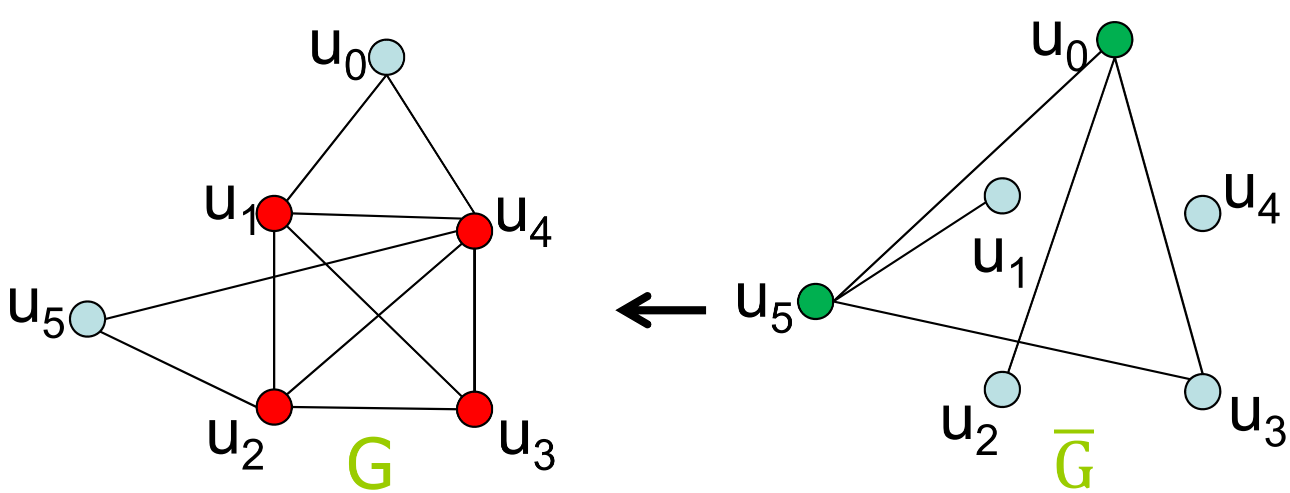


G has a size-**k** clique ← $\overline{G}$ has a size-**(n-k)** vertex cover

Conversely, suppose U ⊆ V is the size-(**n-k**) vertex cover of $\overline{G}$,
e.g., $U = \{u_0, u_5\}$ is a size-**2** vertex cover of $\overline{G}$

- By definition of vertex cover, for all u, v ∈ V,
  if (u, v) ∈ $\bar{E}$, then at least one of u and v belong to U
- The contrapositive of this statement is for all u, v ∈ V and
  both u and v do not belong to U → (u, v) ∉ $\bar{E}$ → (u, v) ∈ E

Hence, V-U is a clique and V-U has size = **n-(n-k) = k**,
e.g., $V\text{-}U = \{u_1, u_2, u_3, u_4\}$ is a size-**4** clique of G



VERTEX-COVER is NP-Complete

We have shown that:

1. VERTEX-COVER is in NP
2. VERTEX-COVER is NP-Hard

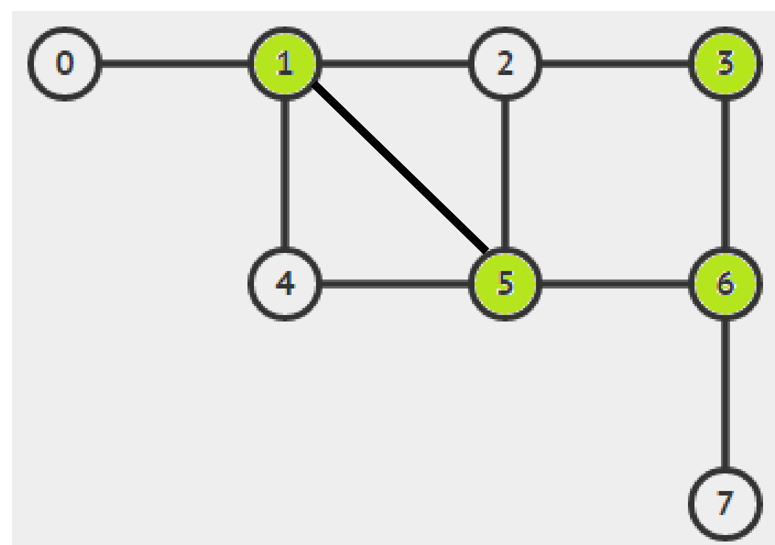
Therefore, VERTEX-COVER (**the decision problem**) is NP-Complete

MIN-VERTEX-COVER is NP-hard (1)

Clearly, **if** we could efficiently find a MIN-VERTEX-COVER (MVC, **an optimization problem**), then we could also efficiently answer the **decision version** of VERTEX-COVER (VC), i.e., VERTEX-COVER $\leq_p$ MIN-VERTEX-COVER

Decide: Can we have VC with **k** vertices of this graph?
➔ Just run MVC optimization algorithm on that graph output yes if the ans ≤ **k**
or no otherwise

- An **O(1)** (polynomial) reduction



MIN-VERTEX-COVER is NP-hard (2)

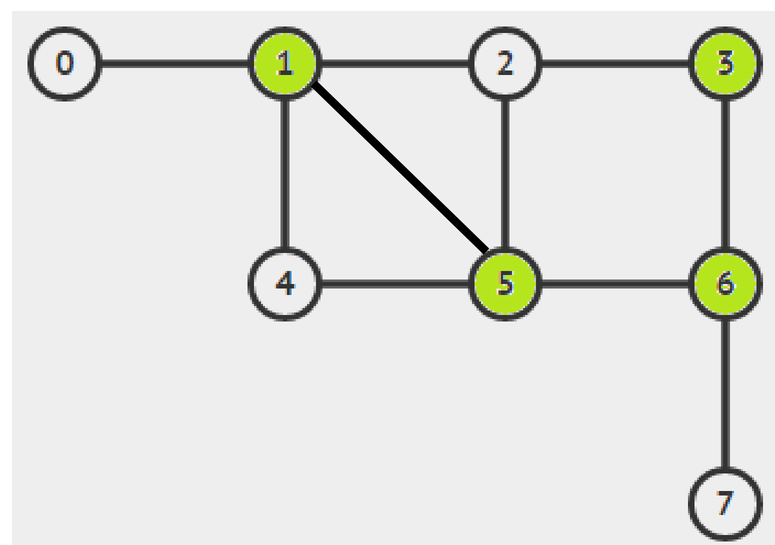
Hence finding a MIN-VERTEX-COVER (MVC) is at least as hard as the decision version, and hence we term it **NP-hard**

MVC is **not NP-complete**

- We do not have polynomial time verifier to show that MVC is in NP

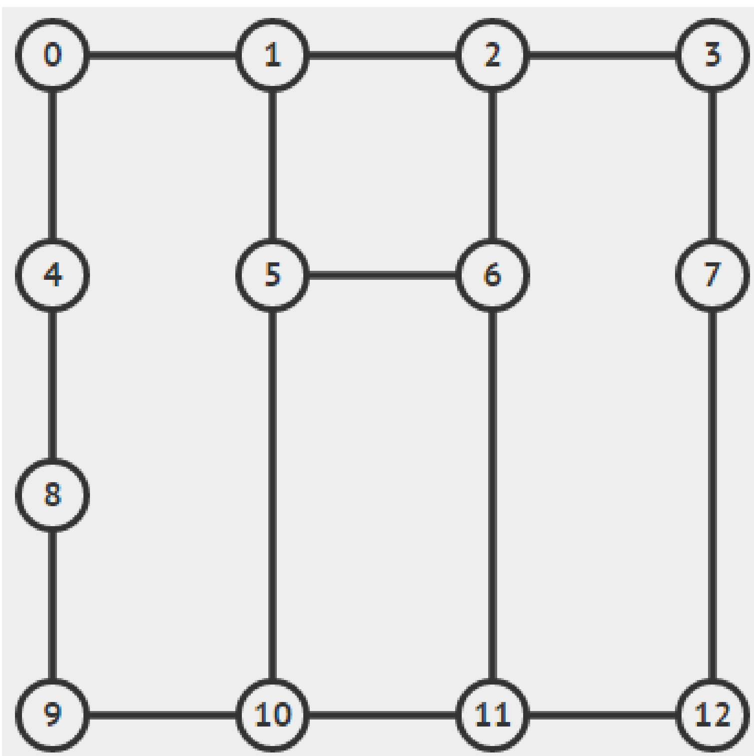
Bonus:

- Solving MVC also solves another NP-hard problem (details in tutorial): Max-Independent-Set (MIS)
  - MIS: un-selected vertices in the example



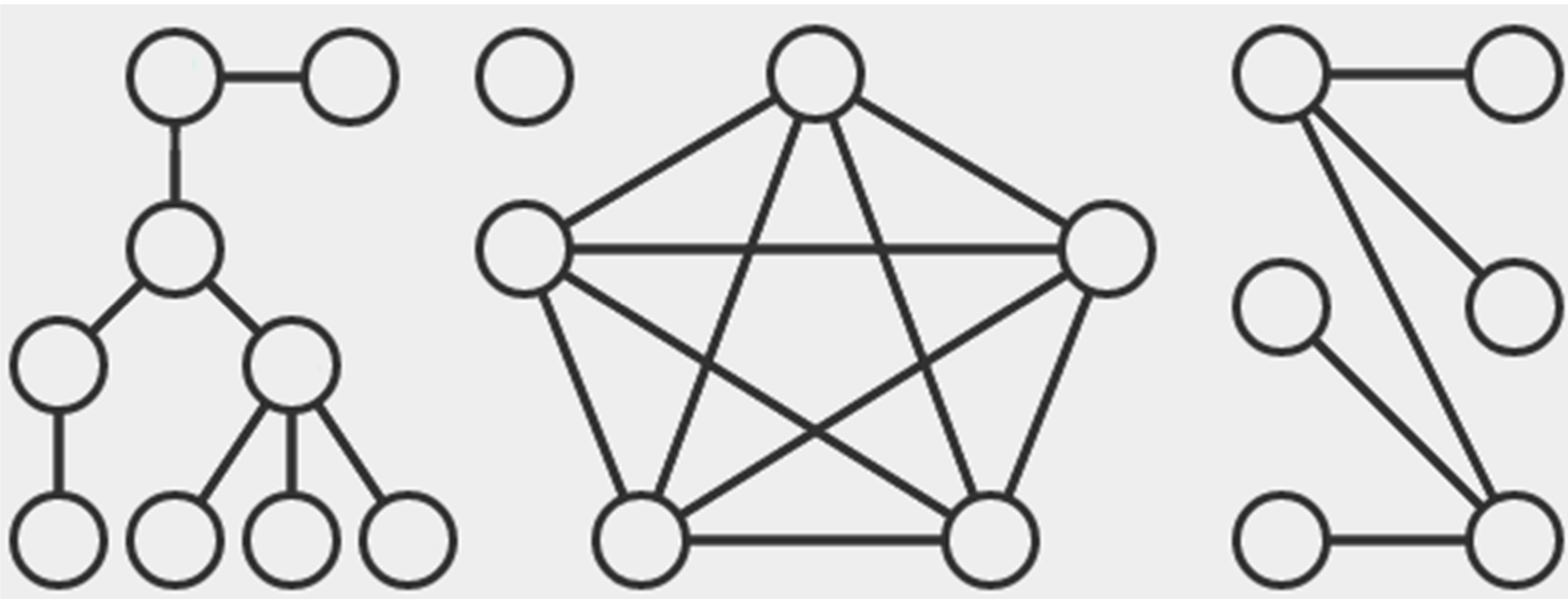
It is really hard... (1)

What is the minimum-sized vertex cover of this graph?

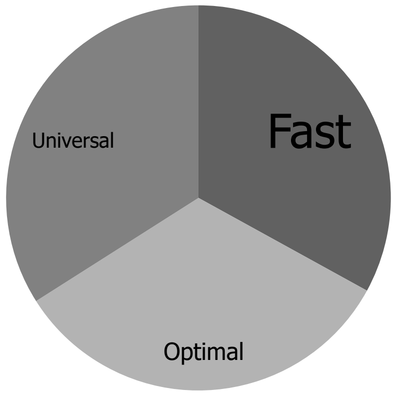


It is really hard… (2)

What is the minimum-sized vertex cover of this graph?



Dealing with an NP-hard Problem



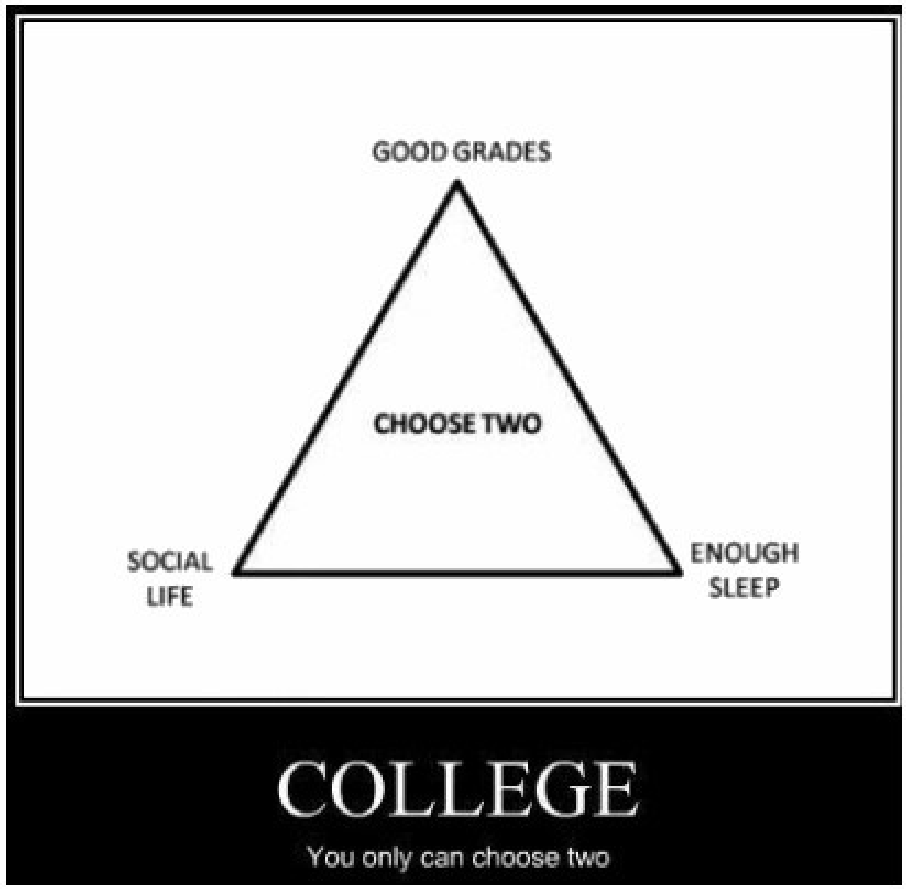
Desirable Solution:

1. Fast (i.e., polynomial time)
2. Optimal (i.e., yielding the best solution possible)
3. Universal (i.e., good for all instances/inputs)

In reality: Choose 2 out of 3:

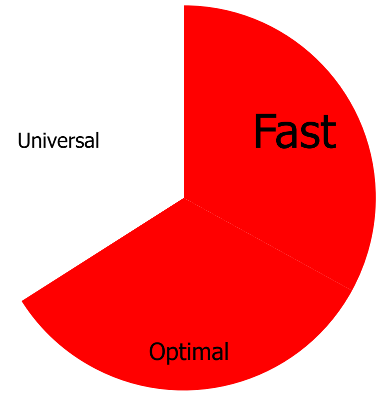
1. Deal with the special case (lost universality)
2. Deal with parameterized solution (lost speed)
3. Consider approximate solution (lost optimality)

Intermezzo



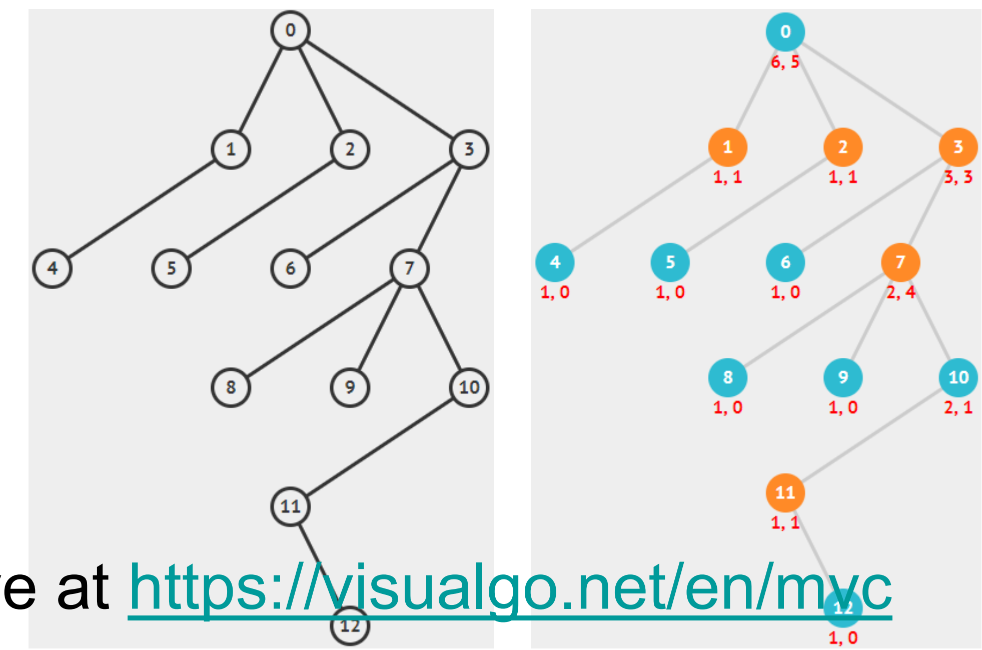
PS: Similar pictures can be easily Googled online

MVC on Tree – Special Case



Dynamic Programming solution:

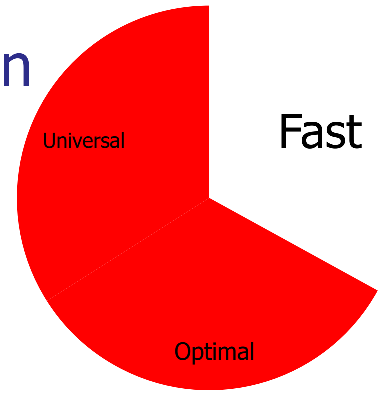
- $in(v) = 1 + \Sigma_{c \subset children(v)} \min(in(c), out(c))$
- $out(v) = \Sigma_{c \subset children(v)} in(c)$
- Base case at a leaf v: $in(v) = 1$, $out(v) = 0$
- answer = min(in(r), out(r)), computable in **O(n)**
- The tree can be a general tree (it does not have to be a binary tree)



Try DP on Tree live at https://visualgo.net/en/mvc

PS: There are a few other known special cases of MVC ☺

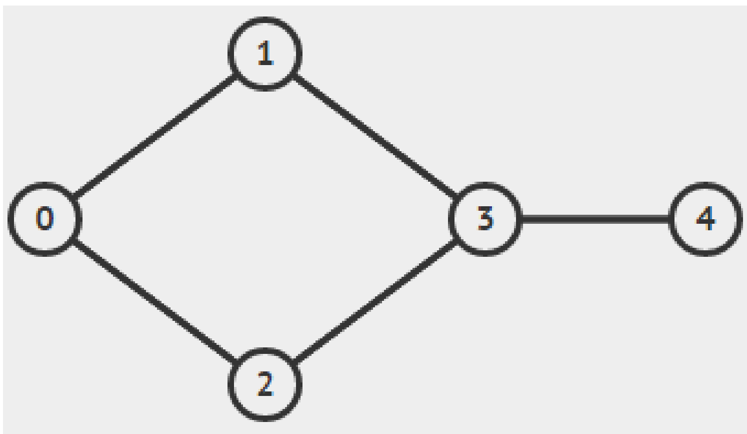
MVC with small k – Parameterized Solution

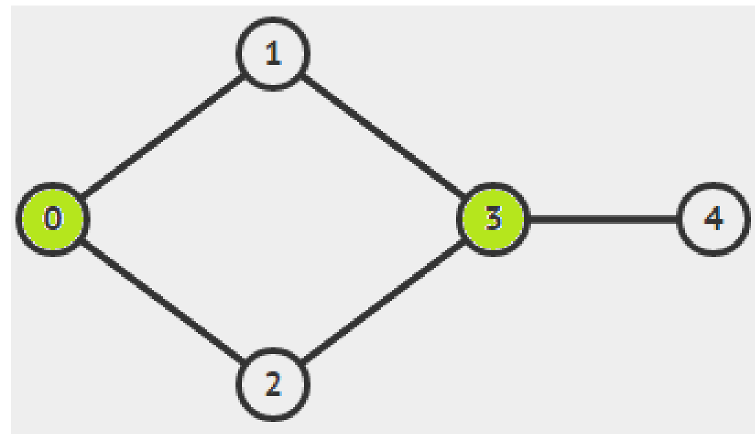


PS: Pure Brute Force is $O(2^n m)$, see https://visualgo.net/en/mvc

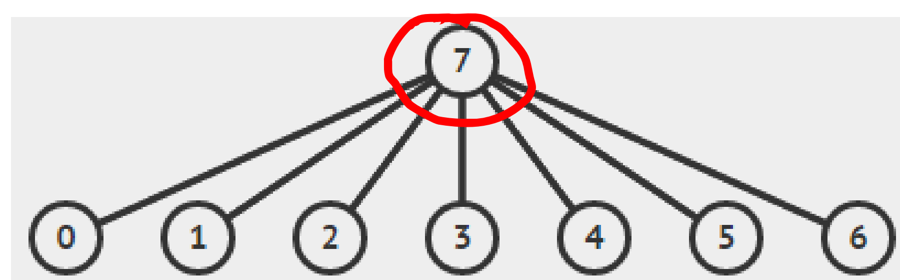
Parameterized Complexity

- What if you are told that $k \leq 2$?





- What if you are told that k is much smaller than n?
  - Naïve $O(n^k m)$ algorithm
    - Very not scalable ☹
  - Better $O(2^k m)$ algorithm
    - Also not scalable, but much better than the naïve one



MVC with small k – Parameterized Solution

Algorithm: ParameterizedVertexCover($G = (V, E), k$)

Procedure:

/* Base case of recursion: $\perp$ = failure, $\emptyset$ = empty set */

if $k = 0$ **and** $E \neq \emptyset$ **then return** $\perp$

if $E = \emptyset$ **then return** $\emptyset$

/* Recurse on arbitrary edge e: */

Let $e = (u, v)$ be any edge in G.

S_u = ParameterizedVertexCover($G_{-u}, k-1$)

S_v = ParameterizedVertexCover($G_{-v}, k-1$)

if $S_u \neq \perp$ **and** $|S_u| < |S_v|$ **then**

  return $S_u \cup \{u\}$

else if $S_v \neq \perp$ **and** $|S_v| \leq |S_u|$ **then**

  return $S_v \cup \{v\}$

else return $\perp$

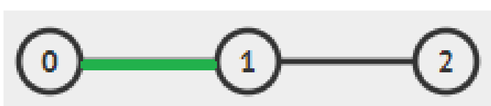
Analysis:

$T(k, m) \leq 2T(k-1, m) + O(m)$

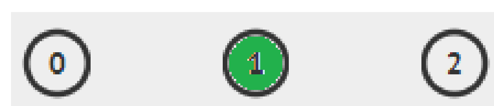
$O(2^k m)$

k = 0?

No… $\perp$



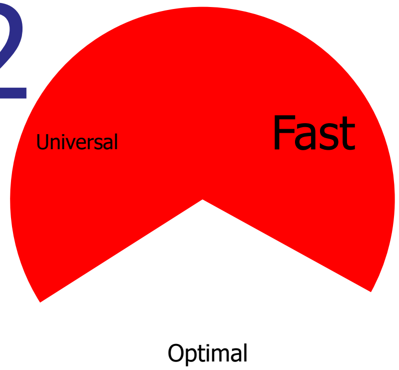
k = 1



k = 0?

Yes, $\emptyset$ (we are done)

To be continued on Week 02



- The third slice of the pie…

Summary

- Re-introducing the MVC problem
- Re-proof that VC (decision) is NP-complete
- ${}_3C_2$ scenarios
- Special case of MVC: On (Binary) Tree, use DP
- Parameterized MVC: Small k, good brute force
- Approximation algorithms for MVC → Week 02