

COL 351:

Analysis and Design of Algorithms

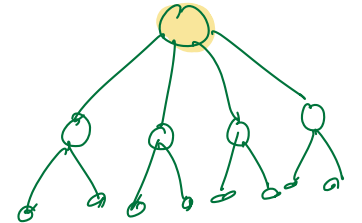
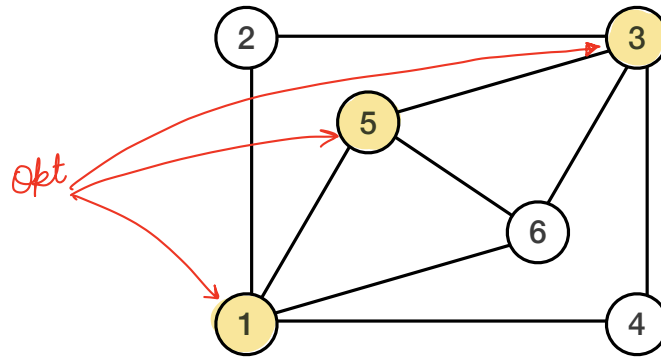
Lecture 3

Facebook Network

Given: A Facebook network $G = (V, E)$ with n users.

Question: What is the minimum number of accounts that when deactivated **results in zero** friends of all remaining users.

Example:



counter-example

Ques: Can greedily removing vertices of mandegree gives optimal ?

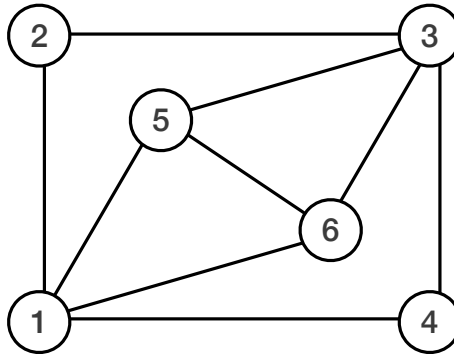
No

Vertex Cover

Given: A graph $G = (V, E)$ with n vertices.

Def: A subset $S \subseteq V$ such that for each $(a, b) \in E$, **at least** one end-point of (a, b) lies in S .

Example:



Optimization question: Find a vertex-cover of minimum possible size.



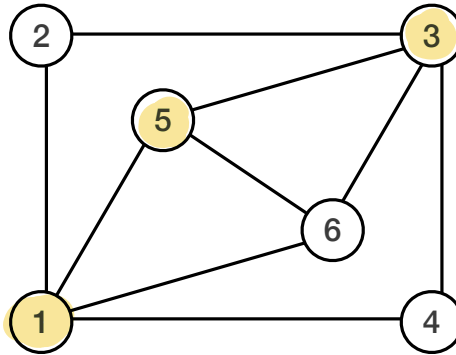
Can we find optimal solution efficiently?

No, vertex cover is NP-hard problem and a poly-time solⁿ is very-unlikely.

Approximate Vertex Cover

Given: A graph $G = (V, E)$ with n vertices.

Def: A subset $S \subseteq V$ such that for each $(a, b) \in E$, **at least** one end-point of (a, b) lies in S .



$$\text{opt sol}^n = \{1, 3, 5\}$$

Approximation Problem: Find a solution S satisfying ...

$$|VC_{\text{opt}}| \leq |S| \leq \text{constant} * |VC_{\text{opt}}|$$

Greedy Approach-I

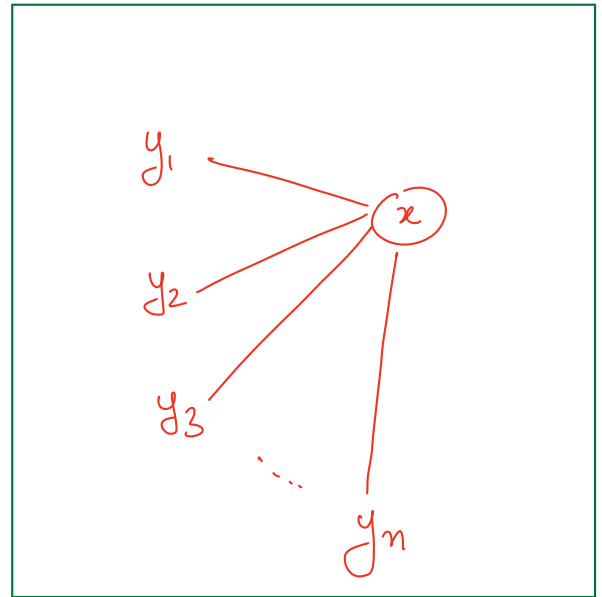
Repeatedly pick an uncovered edge, and add one of its end-points in the solution set.

Initialise $S := \phi$.

While there is an uncovered edge :

- Pick an uncovered edge (x, y) .
- Add x **or** y to S .
- Mark all edges incident to “new” vertex added to S as covered.

Return S .



$$|opt| = 1$$
$$|greedy| = n-1$$

$$OPT = \{x\}$$
$$greedy = \{y_1, \dots, y_n\}$$
$$\text{Approx-factor} = (n-1)$$

Greedy Approach-II

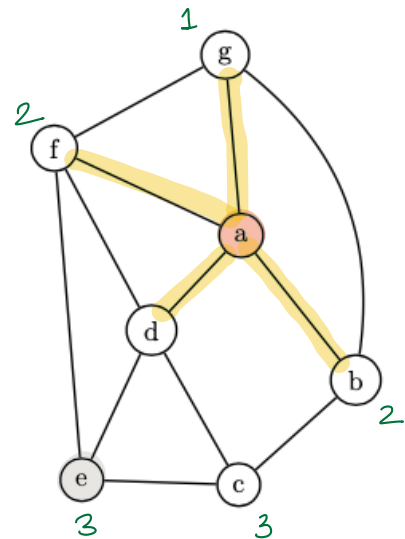
Repeatedly choose vertices incident to the largest number of *currently* uncovered edges.

Initialise $S := \phi$.

While there exists an uncovered edge :

- Pick a vertex x incident to maximum number of uncovered edges.
- Add x to S .
- Mark all edges incident to x as covered.

Return S .



$$S = \{a, e, \dots\}$$

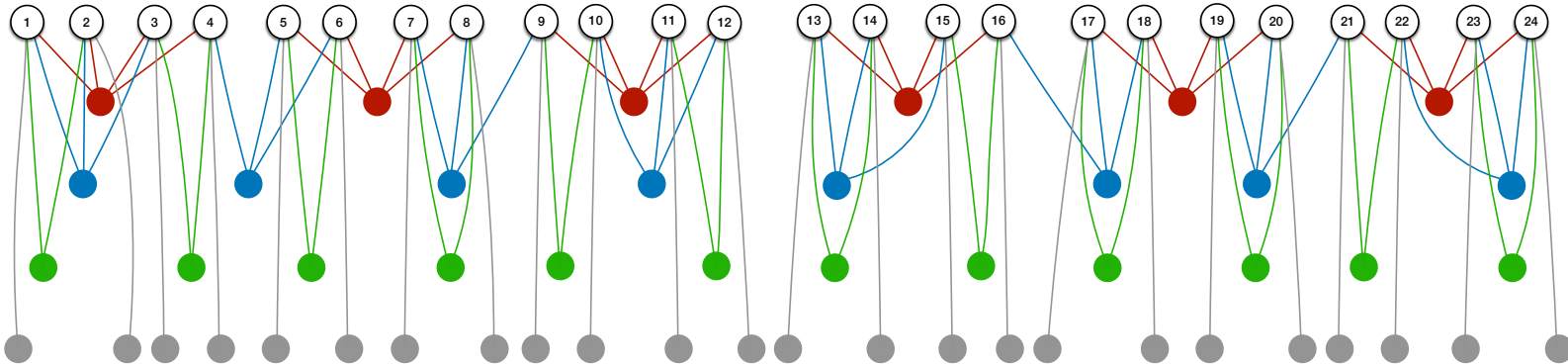
We will prove in lecture 3 that
approximation is $O(\log n)$

Greedy Approach-II

$$\textcircled{*} |Input| = O(n \cdot \log n)$$

$$\textcircled{*} \text{ approx-factor} = O(\log n)$$

Repeatedly choose vertices incident to the largest number of *currently* uncovered edges.



$$|opt\text{-}vertex\text{-}cover| = 24 = \lfloor n \rfloor$$

greedy sol = Red vertices \cup Blue vertices \cup Green vertices
= colored vertices.

$$\begin{aligned} |greedy\ sol| &= 24 + 12 + 8 + 6 = \\ &= 24 \left(1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} \right) \approx \lfloor n \log n \end{aligned}$$

deg = 4	Red — <u>6</u>
deg = 3	Blue — 8
deg = 2	Green — 12
deg = 1	Grey — 24

Greedy-III (and stupid?)

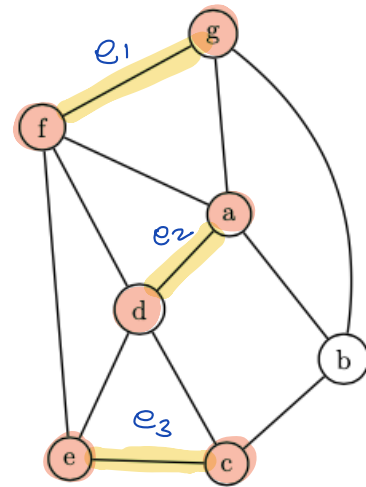
Repeatedly pick an uncovered edge, and add both of its end-points in the solution set.

Initialise $S := \phi$.

While there exists an uncovered edge :

- Pick an uncovered edge (x, y) .
- Add x **and** y to S .
- Mark all edges incident to “new” vertices added to S as covered.

Return S .



Suppose edges ~~is~~ covered are : e_1, e_2, e_3

$$|VC_{opt}| \geq 3$$

This gives us 2-approximation.

Greedy-III

Repeatedly pick an uncovered edge, and add both of its end-points in the solution set.

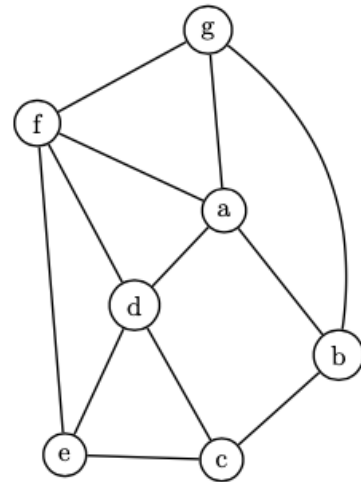
Edges considered at this step don't share a vertex

Initialise $S := \phi$.

While there exists an uncovered edge :

- Pick an uncovered edge (x, y) .
- Add x **and** y to S .
- Mark all edges incident to “new” vertices added to S as covered.

Return S .



Theorem:

If S is the output of the above algorithm, then $|S| \leq 2 \text{ VC}_{opt}$, where VC_{opt} is an optimal vertex cover.

Correctness

Theorem:

If S is the output of the algorithm, then $|S| \leq 2 VC_{opt}$, where VC_{opt} is an optimal vertex cover.

Proof:

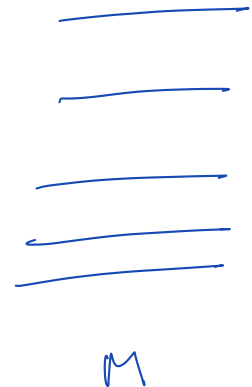
Claim (HW): Edges considered do not share a ^{vertex} edge.

Let M = set of considered edges

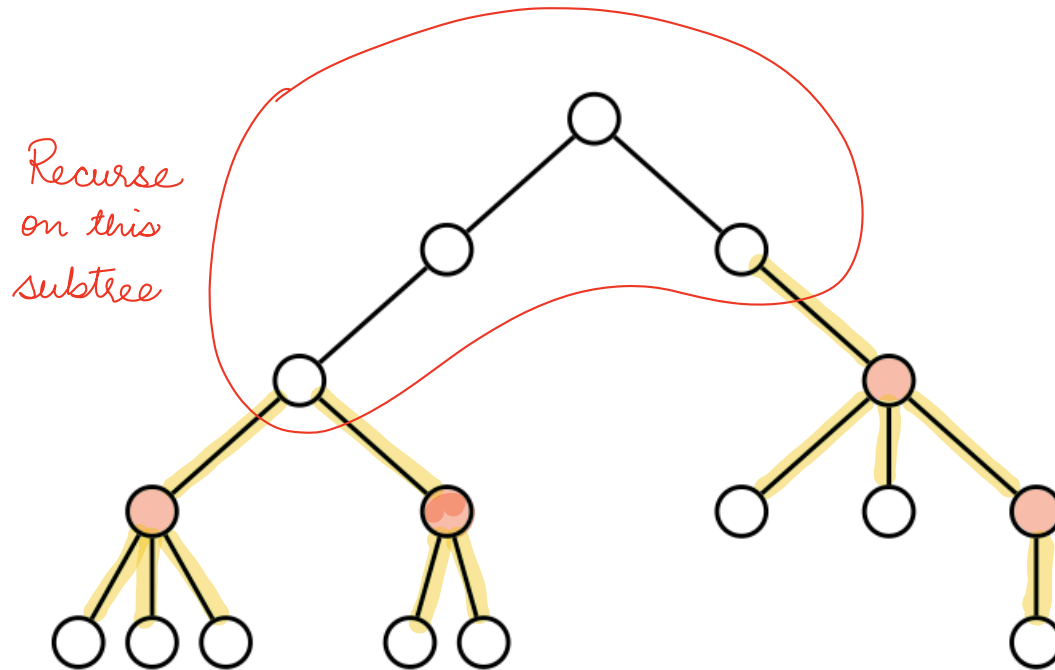
① greedy solⁿ size = $|S| = 2|M|$

② $|VC_{opt}| \geq |M| = \frac{|S|}{2}$

Thus, we get $|S| \leq 2 \cdot |VC_{opt}|$



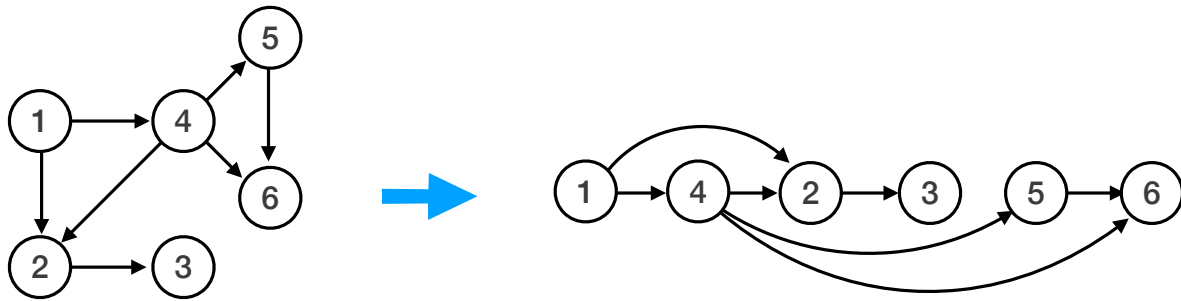
Vertex Cover in Trees/Forests



Theorem: We can compute a smallest vertex cover for tree on n vertices in $O(n)$ time

Topological Sort: Greedy Algorithm

GOAL : A linear ordering of vertices of DAG such that for every directed edge (u, v) , u comes before v in the ordering.



H.W. 1 Prove that DAG contains a vertex of deg "0"

H.W. 2 Give $O(m+n)$ time algo to find topological ordering of DAG using only ARRAYS & DOUBLY-LINK-LIST.