Algorithms + Complexity
Spring 2026

Greedy Approx, part 2

Recap

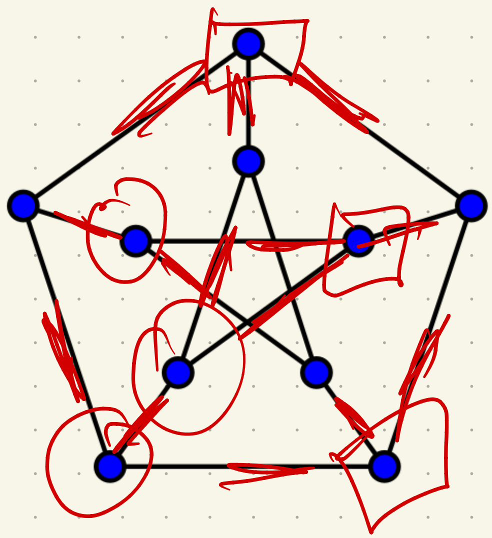
- Next reading :- Redux
- Next HW posted → Dyn. Pro.
  - ↳ orally presented
    - join a group
    - sign up on Canvas calendar

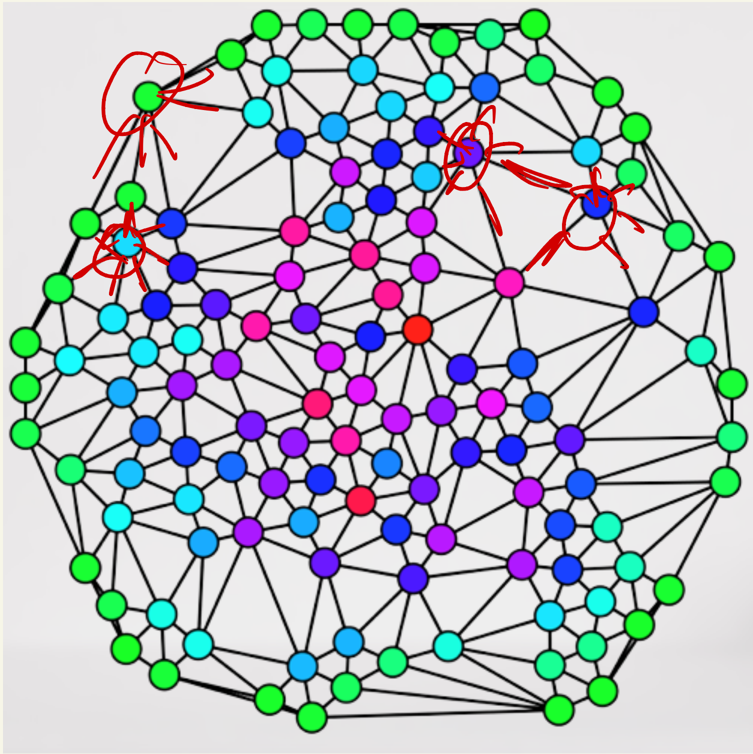
Mon + Tuesday → 16th + 17th

First example

Vertex cover: Given a graph $G=(V,E)$, choose a set of vertices $S \subseteq V$ such that every $e \in E$ is incident to some $v \in S$.

Examples:





How hard?

Easy to find a cover:

$S = V$ $\qquad$ $|V|$ or V

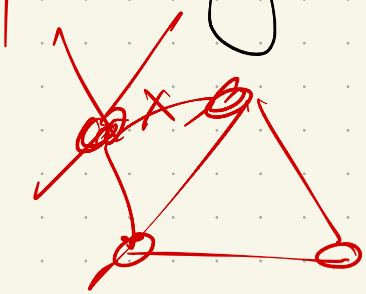
Challenge: make it smaller

↳ minimization problem

Note: In general, NP-Hard. (More later...)

One idea: Use vertices with high degree.

Why? Take lots of edges!

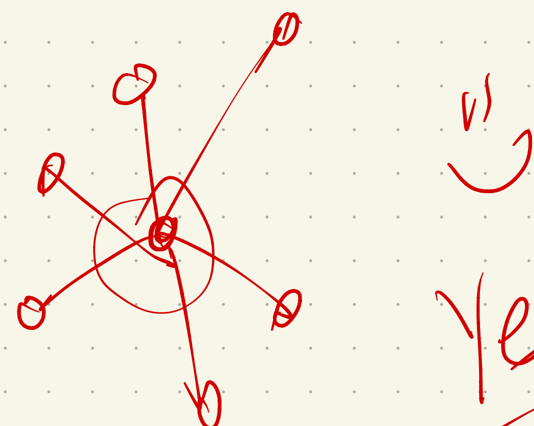


Greedy algorithm:

```
GREEDYVERTEXCOVER(G):
  C ← ∅
  while G has at least one edge
      v ← vertex in G with maximum degree
      G ← G \ v
      C ← C ∪ v
  return C
```

→ Why? these edges are covered
↳ remove v & all incident edges

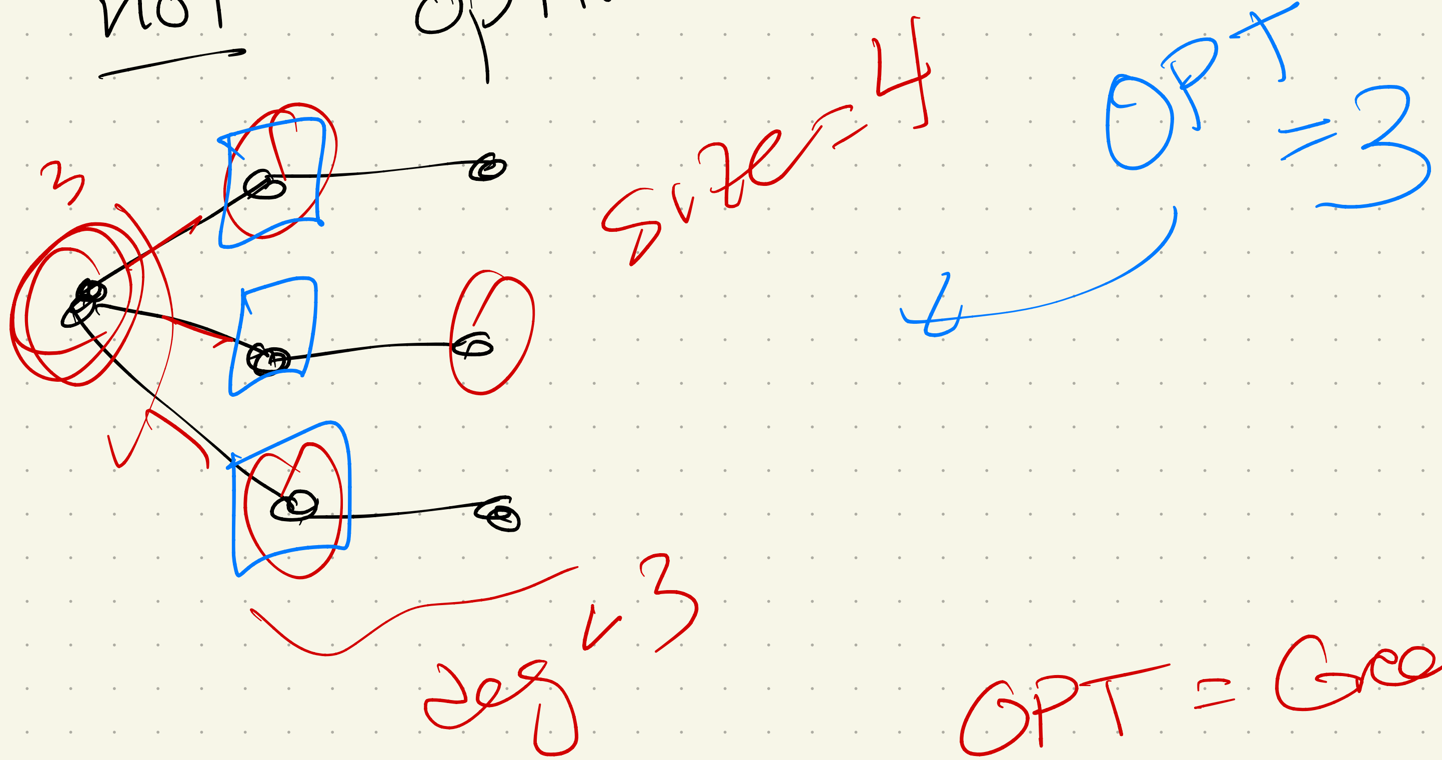
Question: does this ever give the min set?



Yes

Question: how to make it fail?

Need high degree vertices that are not optimal.



OPT = Greedy − 1

But!

only +1 apart

Can we prove this is an approximation to optimal? last slide: example with $|C| = opt+1$

ie $|C| > |opt|$ (see last slide)

but $|C| \leq \alpha \cdot |OPT|$?

Note: Nothing in our algorithm tells us what to aim for!

prev. example $\Rightarrow$ $\alpha > 1$

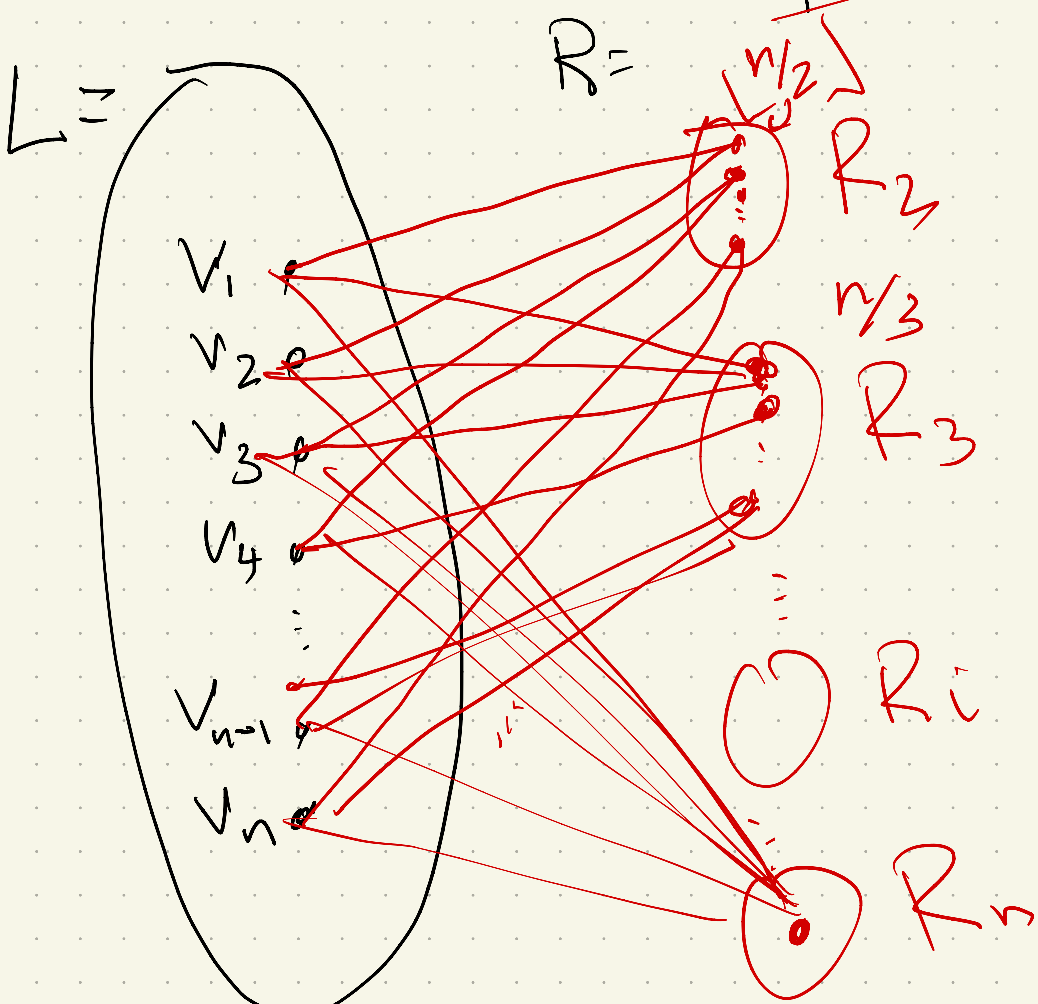
Let's check some notation here...

Back to VC → lower bound: a graph where we fail badly

Question: is it a 2-approximation?

No. (But not obvious...)

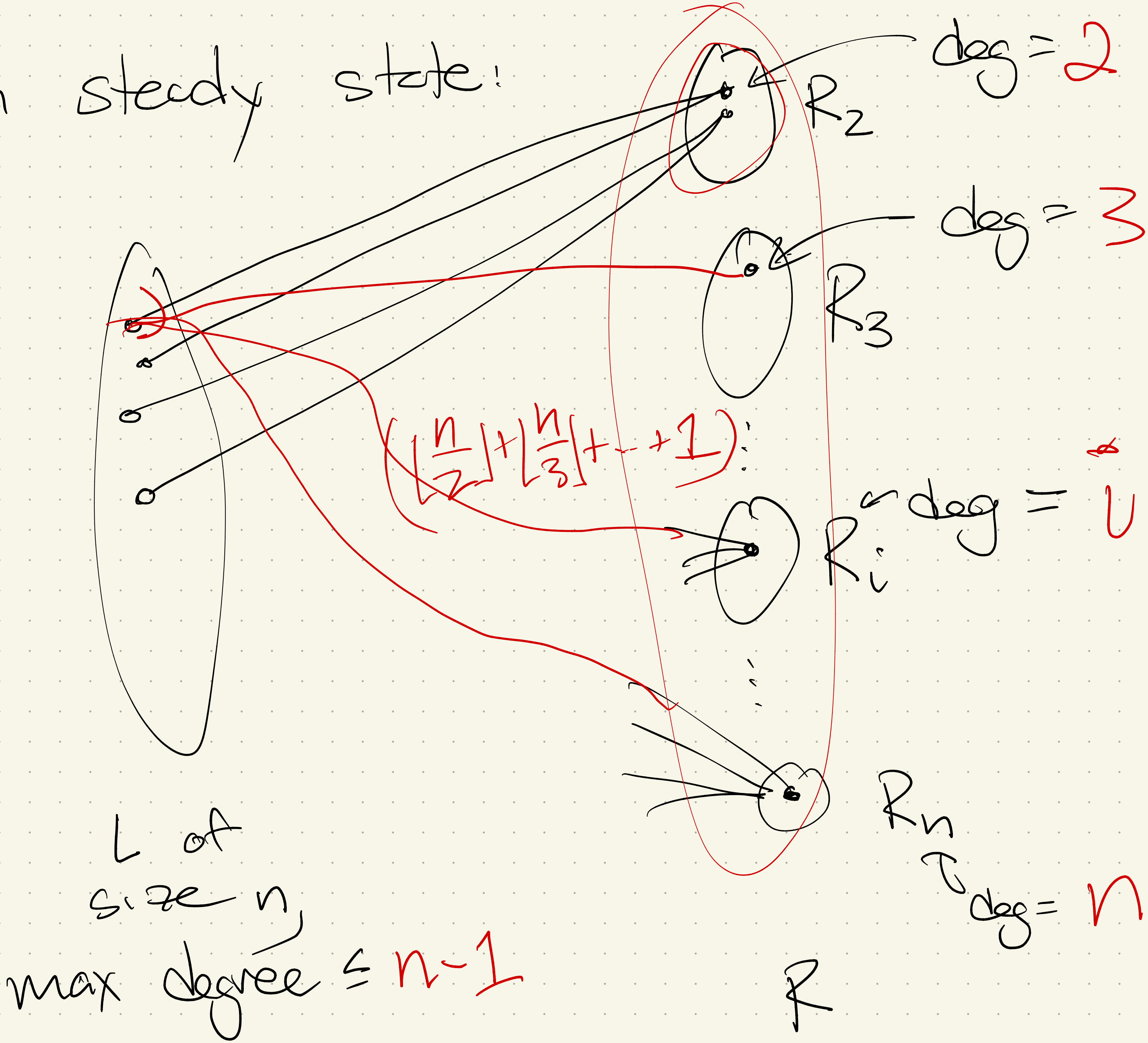
Construction: bipartite graph $G=(V,E)$ where $V = L \cup R$



For R: for each $i \leftarrow 2..n$, add $\lfloor \frac{n}{i} \rfloor$ vertices, each degree i, & connect to <u>different</u> vertices in L.

↳ call these $R_i \subseteq R$

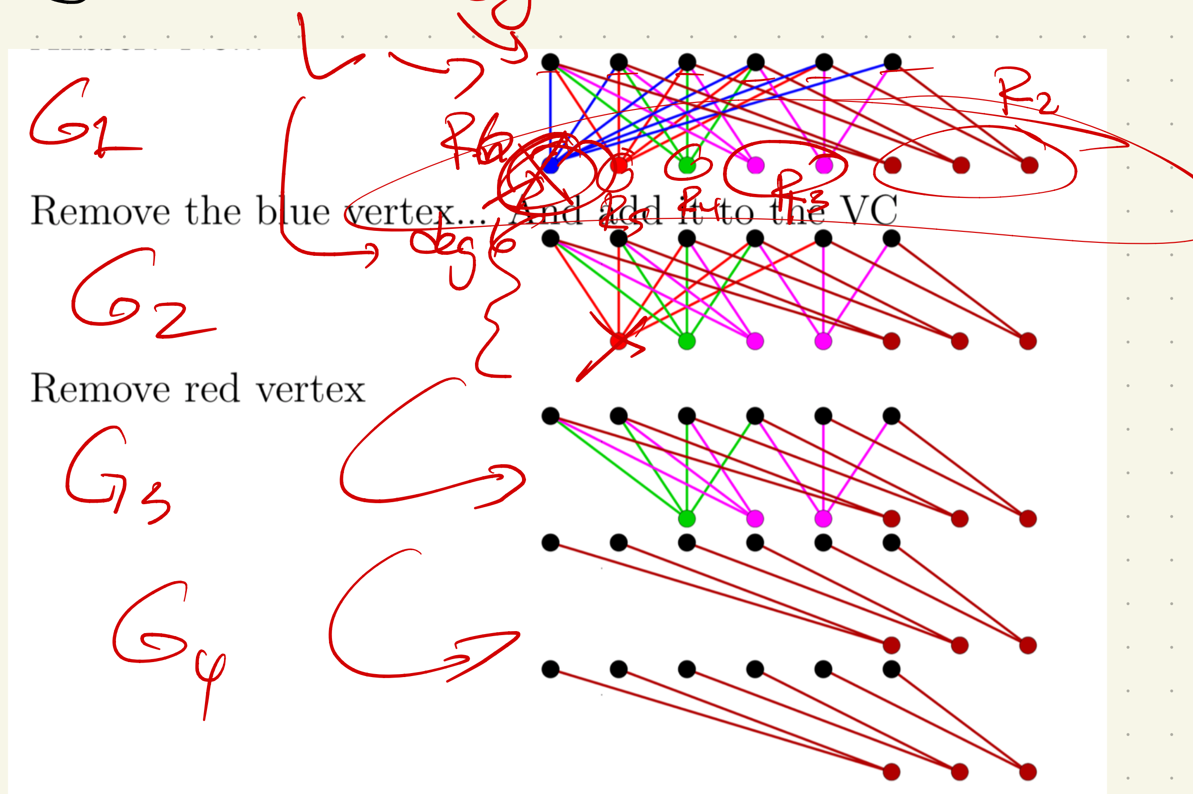
In steady state:



What does our algorithm do?

OPT?

```
GreedyVertexCover(G):
    C ← ∅
    while G has at least one edge
        v ← vertex in G with maximum degree
        G ← G \ v
        C ← C ∪ v
    return C
```



Highest degree vertex?
↳ in R_1 one of degree n.

When removed:

$R_2 \ldots R_{n-1}$ → each of degree $2 \ldots n-1$

in L: each vertex lost ≤ 1 edge
↳ deg $\le n-2$

Remove R_{n-1}
↳ still have $R_2 \ldots R_{n-2}$ & L
↳ deg $\le n-3$

So, in end, all R vertices chosen.

What is $|R|$?

$$|R| = \sum_{i=2}^{n} |R_i| = \sum_{i=2}^{n} \left\lfloor \frac{n}{i} \right\rfloor$$

$$\geq \sum_{i=2}^{n} \frac{n}{i} = n \sum_{i=2}^{n} \frac{1}{i} \geq n \ln n$$

$\ln n = \log_e n$

Recall that "cheat sheet":

Harmonic series:

$$H_n = \sum_{i=1}^{n} \frac{1}{i},$$

&

Harmonic numbers:

$1, \frac{3}{2}, \frac{11}{6}, \frac{25}{12}, \frac{137}{60}, \frac{49}{20}, \frac{363}{140}, \frac{761}{280}, \frac{71}{25}$

$$\ln n < H_n < \ln n + 1,$$

$$H_n = \ln n + \gamma + O\left(\frac{1}{n}\right).$$

So, back to $\alpha(n)$ stuff:

$$|R| \geq n(H_n - 2)$$

$$|L| = n$$

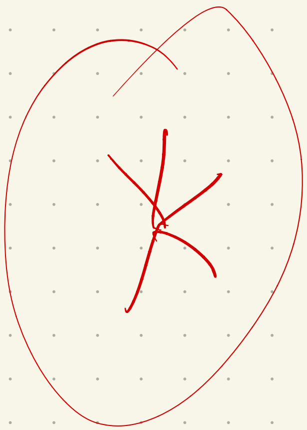
so, greedy factor $\alpha(n) \geq \dfrac{|R|}{|L|}$

& $\dfrac{|R|}{|L|} \geq \dfrac{\cancel{n}(H_n - 2)}{\cancel{n}} = \underline{\underline{\Omega(\ln n)}}$

Note: lower bound! Can we show it always gets at least this?

Theorem Greedy algorithm always chooses a set of size $\leq (\log n) \cdot OPT$

To prove: Rewrite slightly:



GREEDYVERTEXCOVER(G):
- $C \leftarrow \emptyset$
- $G_0 \leftarrow G$
- $i \leftarrow 0$
- while G_i has at least one edge
  - $i \leftarrow i + 1$
  - $v_i \leftarrow$ vertex in G_{i-1} with maximum degree
  - $d_i \leftarrow \deg_{G_{i-1}}(v_i)$
  - $G_i \leftarrow G_{i-1} \setminus v_i$
  - $C \leftarrow C \cup v_i$ ← add 1 vertex
- return C

$|C| \leq OPT \cdot \log(|E(G)|)$

$G = G_1$
↳ remove some v_1, C adds v_1
G_2
↓ remove v_2, C add v_2
G_3
⋮

Let G_i = graph in i^{th} iteration. (edges still not covered)

Let d_i = max degree in G_i

Let C^* = optimal vertex cover in G (which must exist but which we don't know)

We do know that C^* is a vertex cover for each G_i.

So:

$$\sum_{v \in C^*} \text{degree of } v \text{ in } G_i \geq \# \text{ edges in } G_i$$

Why? b/c still a cover, so each edge must attach to <u>some</u> $v \in C^*$

Since $\sum_{v \in C^*} \deg_{G_i}(v) \geq |E(G_i)|$

$\Rightarrow$ average degree in G_i of C^* is $\geq \frac{|E(G_i)|}{|C^*|}$

Why? if all <, then not $\geq |E(G_i)|$

But: this means max degree in G_i is at least this size.

$$\Rightarrow d_i \geq \frac{|E(G_i)|}{|C^*|} = \frac{|E(G_i)|}{OPT}$$

averages

$$x_1 + x_2 + \cdots x_n = k$$

average is $\frac{\sum x_i}{k} = A$

some $x_i \geq A$

why? spps not. each $x_i < A$

$$x_1 + \cdots + x_n < k$$

Also: # of edges in G_i decreases

$$d_i \geq \frac{|E(G_i)|}{OPT} \geq \frac{|E(G_j)|}{OPT} \text{ for } j \geq i$$

$\rightarrow |C^*|$

Now, consider first OPT iterations of loop:

$$G_1 \rightarrow G_1 \rightarrow G_2 \longrightarrow \cdots \longrightarrow G_{OPT}$$

How many edges get removed?

$$\sum_{i=1}^{OPT} d_i \geq \sum_{i=1}^{OPT} \frac{|E(G_i)|}{OPT} \geq \sum_{i=1}^{OPT} \frac{|E(G_{OPT})|}{OPT}$$

$$= |E(G_{OPT})|/OPT + \frac{|E(G_{OPT})|}{OPT} + \cdots + \frac{|E(G_{OPT})|}{OPT}$$

So! $\sum_{i=1}^{OPT} d_i \geq |E(G_{OPT})|$

But! $|E(G_{OPT})| = |E(G)| - \sum_{i=1}^{OPT} d_i$

Why? b/c we remove $d_1, d_2, \ldots, d_{OPT}$ edges

Crazy sums: $\sum_{i=1}^{OPT} d_i \geq |E(G)| - \sum_{i=1}^{OPT} d_i$

$\Rightarrow 2\left(\sum_{i=1}^{OPT} d_i\right) \geq |E(G)|$

$\Rightarrow \sum_{i=1}^{OPT} d_i \geq \frac{E(G)}{2}$

In other words:

OPT iterations removes at least half the edges.

$$|E| \xrightarrow{OPT} \frac{|E|}{2} \xrightarrow{OPT} \frac{|E|}{4} \xrightarrow{OPT} \frac{|E|}{8} \rightarrow \cdots$$

2·OPT

Keep going: OPT iterations more

How many times? $\frac{|E|}{2^d} < 1$

After $\log_2(|E|)$ rounds, done.

~~How many per round?~~ 1 per iteration of loop

↳ OPT iterations, have OPT vertices

$OPT \cdot \log_2(E)$ ←

Runtime & space:

```
GreedyVertexCover(G):
  C ← ∅
  G_0 ← G
  i ← 0
  while G_i has at least one edge
      i ← i + 1
      v_i ← vertex in G_{i−1} with maximum degree
      d_i ← deg_{G_{i−1}}(v_i)
      G_i ← G_{i−1} \ v_i
      C ← C ∪ v_i
  return C
```

$O(V)$

$V+E$

$O(E)$

VE

Input: graph G

adj list

v: nbrs

$O(V+E)$

adj matrix

v | 1 ← connected by edge

$O(V^2)$

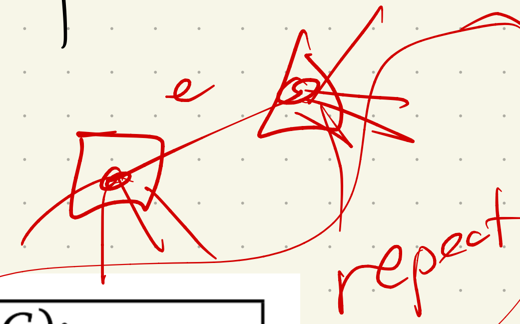
$E \le \binom{V}{2} = \frac{V(V-1)}{2} = V^2$

$OPT \cdot \log(E)(V+E)$

& $O(V+E)$ space to build all G_i's

A different approximation – simpler idea:

- pick any edge + add its endpoints to the cover
- delete all "covered" edges
- Repeat

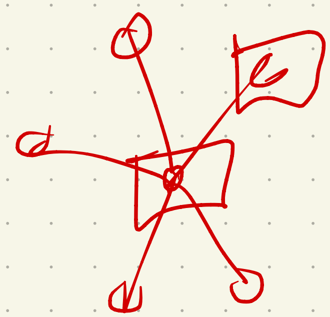


DUMBVERTEXCOVER(G):

$C \leftarrow \varnothing$
while G has at least one edge
  $(u, v) \leftarrow$ any edge in G
  $G \leftarrow G \setminus \{u, v\}$
  $C \leftarrow C \cup \{u, v\}$
return C

Seems worse, right?

- adding a redundant vertex each time!
- vertices might have low degree!

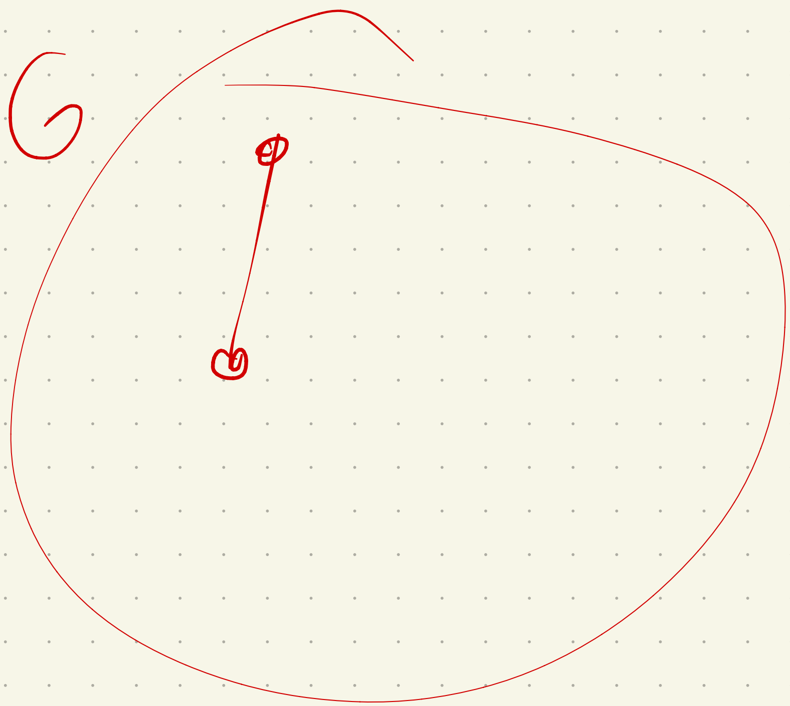


Theorem

Dumb vertex cover is a 2-approximation. (versus log E-approx)

Proof Let C be greedy cover here, & C^* be OPT. ← exists but it is unknown

For each edge $e = \{uv\}$:



↳ either u or v is in C^*.

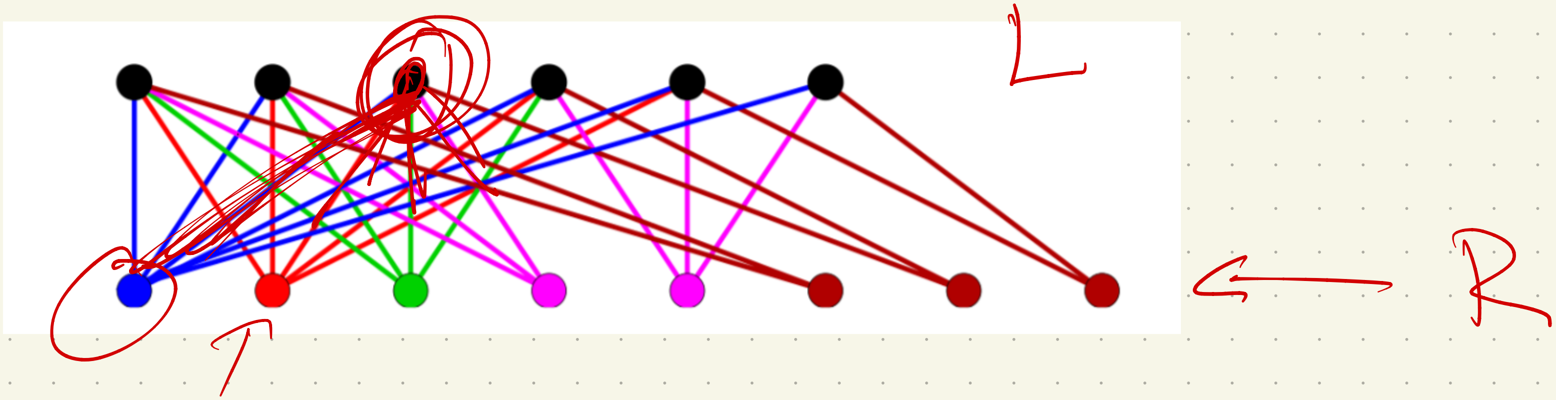
$$|C^*| \leq 2|C|$$

↑

$$\alpha(n) = 2$$

∎

Huh?



Defects worst case pattern by adding $v \in L$ immediately.

Parameterized Complexity

Next section considers: can we get an exact solution that is exponential, but in some other parameter?

Example: in Vertex Cover, can get exact answer in $O(n^{k+2})$, where $k = |Opt|$

↳ Can improve to $O(3^k n^2)$

Brute force: try all $\binom{n}{k}$ size subsets, for $k = 1, 2, 3, 4, \ldots$ ~ $\binom{n}{k} = \frac{n!}{k!(n-k)!} \approx O(n^{k+1}$

Traveling Salesman (TSP)

Given n cities with pairwise distances between them, find the shortest cycle visiting all cities.

This is NP-Hard: more next week!

But idea: Take some problem X where we have reason to believe it will not have a polynomial solution.

Show any alg for TSP would be a subroutine to solve X.

So: TSP is probably hard (reduction)

Additional benefit:

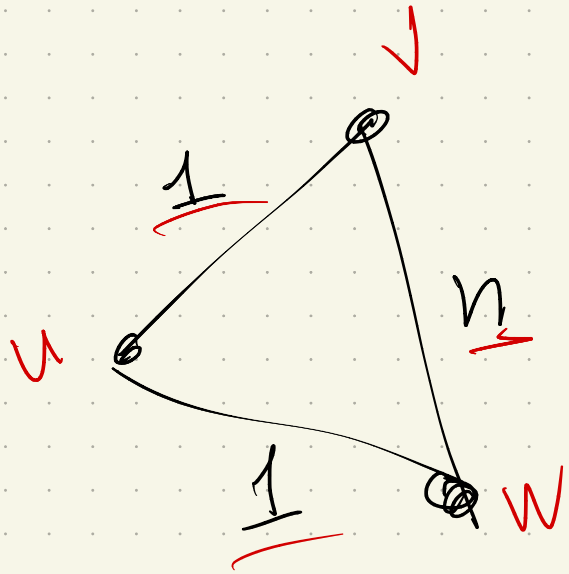
The reduction for TSP is from Hamiltonian cycle:

Any approximation algorithm for TSP would give an exact solution to Ham cycle

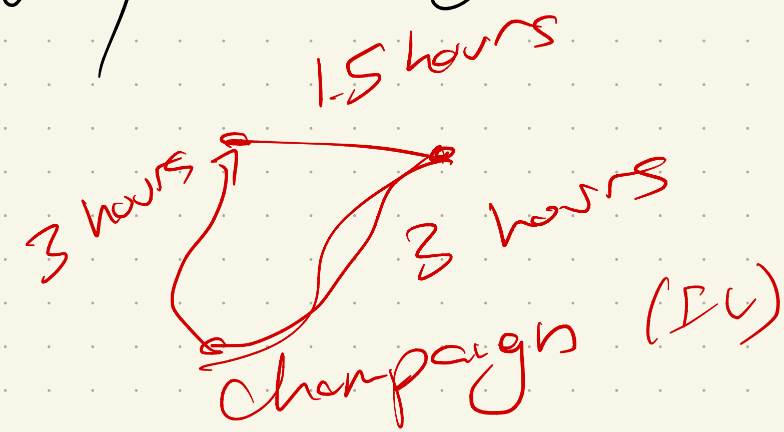
⇒ Hard to approximate (unless P=NP)

So, why study?

The reduction builds strange graphs!



why strange?



Triangle inequality:

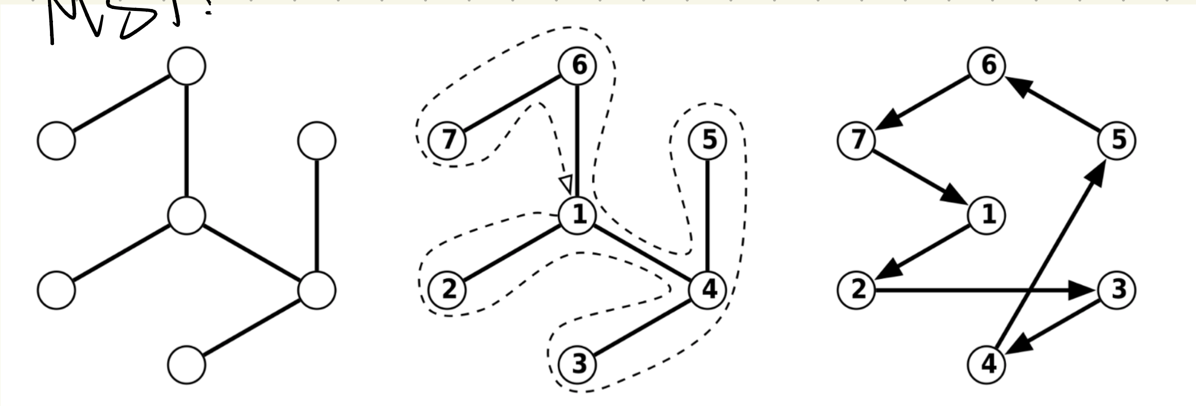
$$d(u,v) \le d(u,w) + d(w,v)$$

common assumption

Theorem: If G satisfies the triangle inequality, can compute a 2-approx for TSP.

Idea: Use Minimum Spanning Tree (MST):

MST:



Then:

Proof:

Let OPT be the cost of optimum solution to TSP.

Let MST be the weight of the Minimum spanning tree.

And let X be the weight of our constructed cycle.

Need to show:

$$X \leq 2 \cdot OPT$$

Step 1: Prove $X \leq 2\,MST$:

Step 2: Prove $MST \leq OPT$:

Why? TSP solution is a cycle, so:

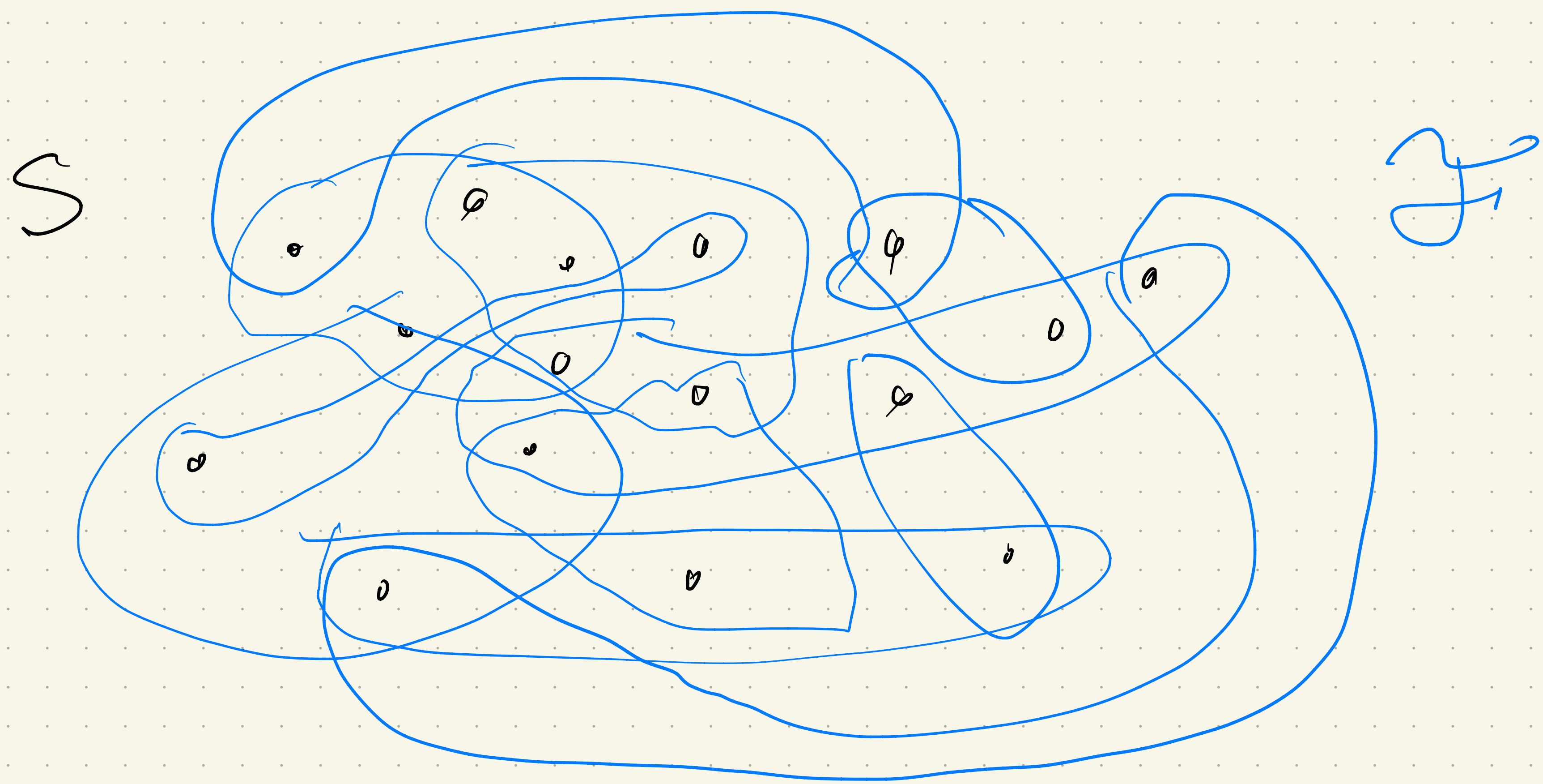
Set Cover

Another classic NP-Hard Problem.

Set Cover

Instance: $(\mathsf{S}, \mathcal{F})$:
S - a set of n elements
$\mathcal{F}$ - a family of subsets of S, s.t. $\bigcup_{X \in \mathcal{F}} X = \mathsf{S}$.

Question: The set $\mathcal{X} \subseteq \mathcal{F}$ such that $\mathcal{X}$ contains as few sets as possible, and $\mathcal{X}$ covers S. Formally, $\bigcup_{X \in \mathcal{X}} X = \mathsf{S}$.



Greedy Set Cover

How should we be greedy?

Sanity Check: does it work?

GreedySetCover$(S, \mathcal{F})$
  $\mathcal{X} \leftarrow \emptyset$; $T \leftarrow S$
  while T is not empty **do**
    $U \leftarrow$ set in $\mathcal{F}$ covering largest
      # of elements in T
    $\mathcal{X} \leftarrow \mathcal{X} \cup \{U\}$
    $T \leftarrow T \setminus U$

  return $\mathcal{X}$.

Observation: Let α_i be # of new elements covered in iteration i of loop.
Then, $\alpha_1 \geq \alpha_2 \geq \cdots \geq \alpha_m$, where m is total # of iterations.

Why?

Notation:

Let $\{V_1, \ldots, V_k\}$ be OPT set cover.

Let T_i = uncovered elements at beginning of iteration i

and U_i = set chosen in i^{th} iteration

```
GreedySetCover(S, 𝓕)
    𝒳 ← ∅;  T ← S
    while T is not empty do
        U ← set in 𝓕 covering largest
            # of elements in T
        𝒳 ← 𝒳 ∪ {U}
        T ← T \ U

    return 𝒳.
```

Lemma: $\alpha_i \geq \frac{|T_i|}{k}$.

Proof: Consider OPT again → k sets, & covers T_i.

Some set in OPT must have size $\frac{|T_i|}{k}$ → why?

Greedy picks biggest coverage, so

$|U_i| \geq$

Rewrite: if $\alpha_i \geq \frac{|T_i|}{k}$ and

$$|T_{i+1}| = |T_i| - \alpha_i$$

$$\Rightarrow |T_{i+1}| \leq$$

<u>Thm</u>: GreedySet cover is $O(\log n)$ approx.

<u>Proof</u>: Need to know how many times the loop runs (since adds 1 set per iteration.)

```
GreedySetCover(S, F)
    X ← ∅;  T ← S
    while T is not empty do
        U ← set in F covering largest
            # of elements in T
        X ← X ∪ {U}
        T ← T \ U

    return X.
```

Well,

$$|T_i| \le (1-\frac{1}{k})|T_{i-1}| \le$$

When do we reach an iteration M where this bound shows $|T_M| < 1$?

(Loop would then stop).

Math tricks: $1-x < e^{-x}$ for $x \ge 0$.

Let $M = \lceil 2k \ln n \rceil$:

$$|T_M| \le$$

(Math, cont):

End recap: If $|OPT| = k$:

After $2k \ln n$ repetitions, T is empty.

```
GreedySetCover(S, 𝓕)
    𝒳 ← ∅;  T ← S
    while T is not empty do
        U ← set in 𝓕 covering largest
            # of elements in T
        𝒳 ← 𝒳 ∪ {U}
        T ← T \ U

    return 𝒳.
```

So: $|X| \le \lceil 2k \ln n \rceil$

$$\Rightarrow \frac{|X|}{|OPT|} \le O(\log n)$$

$\Rightarrow O(\log n)$ approximation.