"9A"

Approximation Algorithms

→ Many problems of practical significance are NP-Complete, yet they are too important to abandon merely because we don't know how to find an optimal solution in polynomial time.

→ Atleast 3 ways to get around NP-Completeness

(1) The actual inputs are small, an algorithm with exponential running time is perfectly satisfactory.

(2) We may be able to isolate important special cases that we can solve in polynomial time.

(3) We may come up with approaches to find near-optimal solution in polynomial time. (Either in the worst case or the expected case.)

→ In practice, "Near-Optimality" is often very good.

→ We call an algorithm that returns near-optimal solutions an "Approximation Algorithm."

→ This chapter presents polynomial-time approximation algorithms for several NP-Complete problems.

✱ Performance Ratios for Approximation Algo:

→ Algorithm for a problem has an approximation ratio of $S(n)$ if, for any input of size n, the cost C of the solution produced by the algorithm is within a factor of $S(n)$ of the cost C^* of an optimal solution:

$$\max\left(\frac{C}{C^*}, \frac{C^*}{C}\right) \leq S(n)$$

If an algo. achieves an approximation ratio of $S(n)$, we call it a $S(n)$ approximation algo.

✱ For Maximization Problem,

$$0 < C \leq C^*$$

C: Cost of the solution produced by approx. algo

C^*: Cost of the solution i.e. Optimal solution

∴ Ratio $= \frac{C^*}{C}$ gives the factor by which the cost of an optimal solution is larger than the cost of the approximate solution.

★ For Minimization Problem

$$0 < C^* \leq C$$

C^* : Optimal Solution

C : Obtained Approximate Solution

Here $\frac{C}{C^*} = \boxed{\text{Ratio} \leq \delta(n)}$

→ It gives the factor by which the cost of approximate solution is larger than the cost of an optimal solution.

→ For many problems, we have small Constant Approximation Ratios

E.g. For Approximate Min. Vertex Cover there exist Ratio 2

i.e. $\frac{C}{C^*} \leq \boxed{\delta(n) = 2}$

$$\therefore \boxed{C \leq 2C^*}$$

i.e. Approximate sol^n cost is at most 2 times the cost of the optimal sol^n.

→ For some problems ratios grow as a function of n.

E.g. "Set Cover" problem.

→ Since for Min. Problem

$$\frac{C}{C^*} \leq S(n)$$

→ For Maximiz. Problem

$$\frac{C^*}{C} \leq S(n)$$

∴ We can say

$$\text{Max}\left\{\frac{C}{C^*}, \frac{C^*}{C}\right\} \leq S(n)$$

→ In general for any type of problem.

→ It says Approximation Ratio is $S(n)$.

Note: Approximation Ratio can never be less than 1.

Why ⇒ Let's say Approx. Ratio is less than 1 for Minimization problem

$$\therefore \frac{C}{C^*} < 1 \Rightarrow C < C^*$$

Which Contradicts the fact that for Minimization Problem $C^* \leq C$ i.e optimal solution ≤ Approx. Soln

Date
Page

★ Say, for Maximization problem - Approx. Ratio is less than 1

i.e. $\frac{C^*}{C} < 1$

$\Rightarrow \boxed{C^* < C}$

Cost of Optimal soln — Cost of Approximate soln

→ This Contradicts the fact that the approximate soln for Maximization problem is less than Optimal soln cost.

∴ For Both the types of problems ratio can never be less than 1

∴ Approximation Ratio $\boxed{S(n) \geq 1}$

→ If $S(n) = 1$, which means $C = C^*$, which may not be practically Correct.

If $\rho(n) = 1 \Rightarrow C = C^*$

C: Approximate solution by Polytime algo

C^*: Optimal soln by Exponential time algo

Which means Approx. algo always returns exact soln. That is not true practically.

∴ For approximation algorithms

Ratio $\boxed{\rho(n) > 1}$

→ $\rho(n)$ can be constant.

or

$\rho(n)$ can be function of n

→ If $\rho(n)$ is constant, it is better approximation strategy

☆ Approximate Vertex Cover

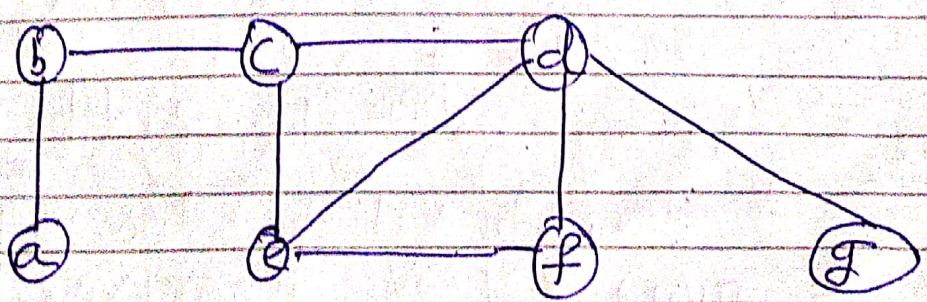
→ Vertex Cover of undirected graph $G=\langle V,E\rangle$ is a subset $V' \subseteq V$ such that if (u,v) is an edge of G, then either $u \in V'$ or $v \in V'$ (or both).

→ Size of vertex cover = Number of vertices in it.

Algo: Approx-Vertex-Cover(G)

```
{
(1) C = φ ;
(2) E' = G.E ;
(3) While E' ≠ φ
    {
(4) → Let (u,v) be any arbitrary edge
      of E'
(5) → C = C ∪ {u,v}
(6) → Remove from E' every edge
      incident on either u or v.
    }
  return C ;
}
```

E.g.



(1) $C = \phi$

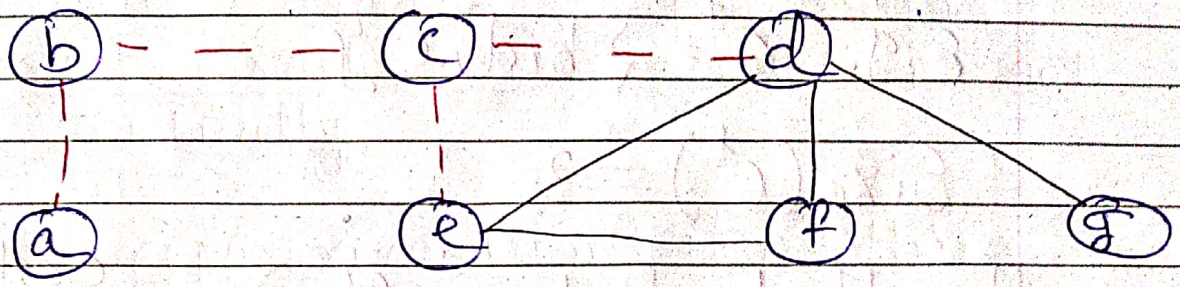
(2) $(u,v) = (b,c)$ be random edge selected.

$\therefore C = C \cup \{u, v\}$

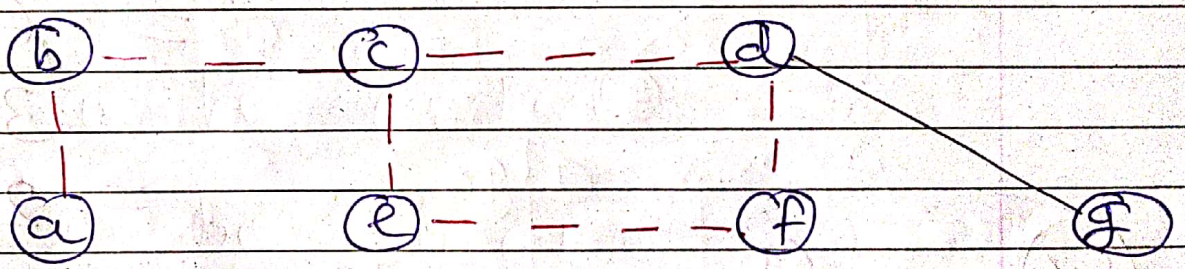
$= \phi \cup \{b, c\}$

$= \{b, c\}$

Remove from E' every edge incident on either b or c.



③ Select any edge (u, v) say it be (e, f)



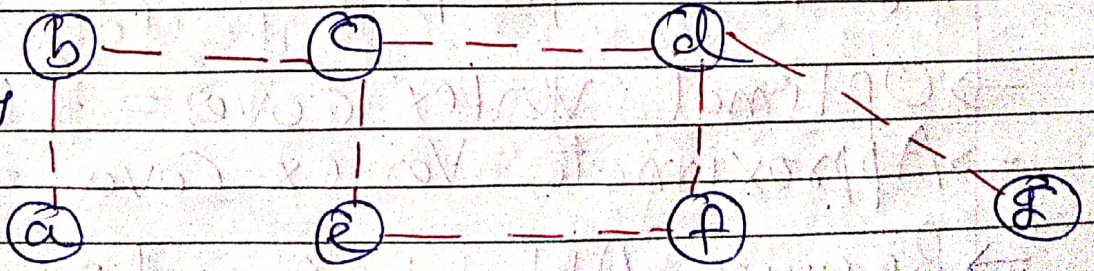
$C = C \cup \{e, f\}$

$= \{b, c\} \cup \{e, f\}$

$= \{b, c, e, f\}$

④ Select any edge $(u, v) = \{d, g\}$

$C = \{b, c, e, f, d, g\}$



⑤ Now No edges left

$\therefore$ Algo will terminate

$\therefore C = \{b, c, d, e, f, g\}$ Size $(C) = 6$

☆ What is the size of optimal vertex cover?

For this graph size 3 vertex cover is possible

e.g. $C^* = \{b, d, e\}$

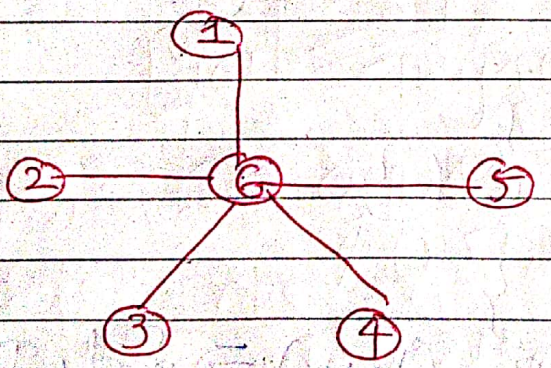
Size $(C^*) = 3$

For that Ratio $\delta(n) = \frac{\text{Cost of } C}{\text{Cost of } C^*}$

$= \frac{6}{3}$

$= 2$

(Ex)



For this graph

→ Optimal vertex cover = {6}

→ Approximate Vertex Cova = {1, 6}

→ Because Atleast 2 endpoints of ~~Every~~ edge will ~~form~~ any approximate vertex cover.

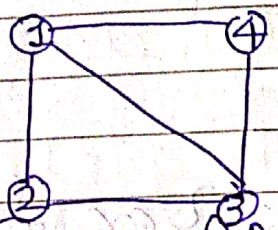
∴ Cost of $C = 2$
Cost of $C^* = 1$

$$\therefore \text{Ratio} = \frac{\text{Cost of approx.}}{\text{Cost of optimal}}$$

$$= \frac{2}{1}$$

$$= 2$$

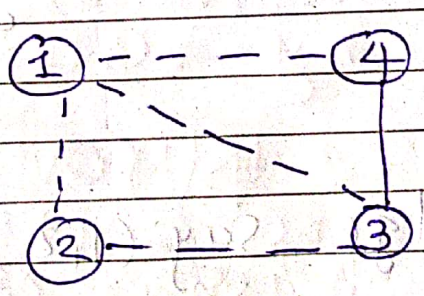
Ex



→ Optimal Vertex Cover = {1,3}

Say, Approx. Vertex Cover selects edge (1,2) first.

∴

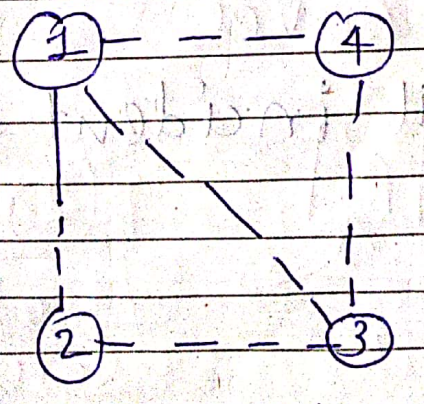


$C = C \cup \{1,2\}$
$= \phi \cup \{1,2\}$
$= \{1,2\}$

Still $E' \neq \phi$

∴ We need to select (3,4)

∴



$C = \{1,2,3,4\}$

∴ Cost of Approx. soln is 4
Cost of optimal soln is 2.

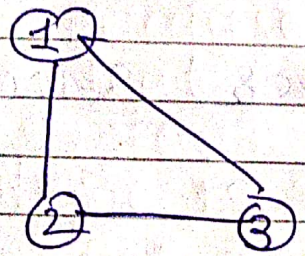
∴ Ratio = $\frac{\text{Cost of Approx sol}^n}{\text{Cost of Optimal sol}^n}$

$= \frac{\#\text{ of Vertices in Approx. Sol}^n}{\#\text{ of Vertices in Optimal sol}^n}$

$= \frac{4}{2}$

$= 2$

Ex



$C^* = \{1,2\}$ or $\{1,3\}$ or $\{2,3\}$

Approx. algo

Pick any edge say (1,2)

$C = C \cup \{1,2\}$

$= \phi \cup \{1,2\}$

$= \{1,2\}$

Remove all incident edges.

$\therefore E' = \phi$

Terminate

$\therefore C = \{1,2\}$

Ratio = $\frac{\#\text{ of vertices in approx. soln}}{\#\text{ of vertices in Optimal soln}}$

$= \frac{2}{2}$

$= 1$

∴ Here Ratio is 1.

So, we have seen examples where Ratio is either 1 or 2.

→ Now, we shall see a proof that for the above algorithm Approximation Ratio is atmost 2.
i.e. $\boxed{S \leq 2}$

Theorem: Prove that Approx. Vertex Cover is a polynomial-time 2-approx. algorithm.

Proof: Above algorithm runs in
$O(V+E)$ if we use adjacency List for E
$O(V^2)$ if we use adj. Matrix for E.
∴ Approx. Vertex Cover is poly time algorithm.

→ Let A denote the set of edges that line 4 of approx. Vertex cover picked.

→ In order to cover the edges in A, any vertex cover – in particular an Optimal cover C^* – must include at least one endpoint of each edge in A.

→ No two edges in A share an Endpoint

Why?

Because once edge is picked, all incident edges are removed.

∴ There is no Common vertex among edges selected by Vertex cover

$$\therefore |C^*| \geq |A| \longrightarrow (1)$$

$|C^*|$ ↑ Number of Elements in Optimal Vertex cover

$|A|$ ↑ Number of Edges Selected by algo

→ But $|C| = 2|A| \longrightarrow (2)$

$|C|$ ↑ Number of Elements in Approx. Vertex Cover algo

$|A|$ ↑ # of edges selected by algo

This is true because we take both endpoints of the selected edge in the answer.

$$|C| = 2|A|$$
$$\leq 2|C^*| \quad (\because |C^*| \geq |A| \Rightarrow |A| \leq |C^*|)$$

$$\therefore \boxed{|C| \leq 2|C^*|}$$

$$\therefore \frac{|C|}{|C^*|} \leq 2$$

$$\therefore \frac{\text{Cost of Approx}}{\text{Cost of Optimal}} \leq 2$$

$$\therefore \boxed{\text{Ratio } \delta \leq 2}$$

∴ Above algo is having Ratio 2. ∎