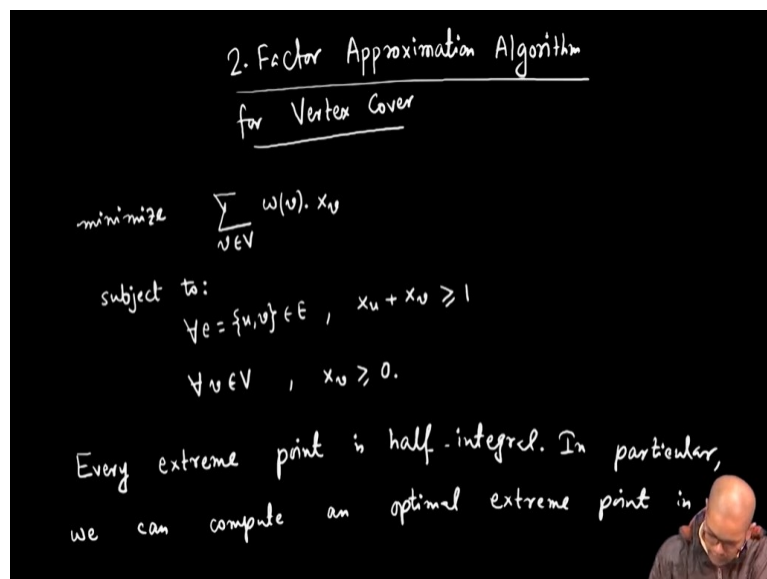


Selected Topics in Algorithms
Prof. Palash Dey
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Module No # 11
Lecture No # 55
Randomized Rounding

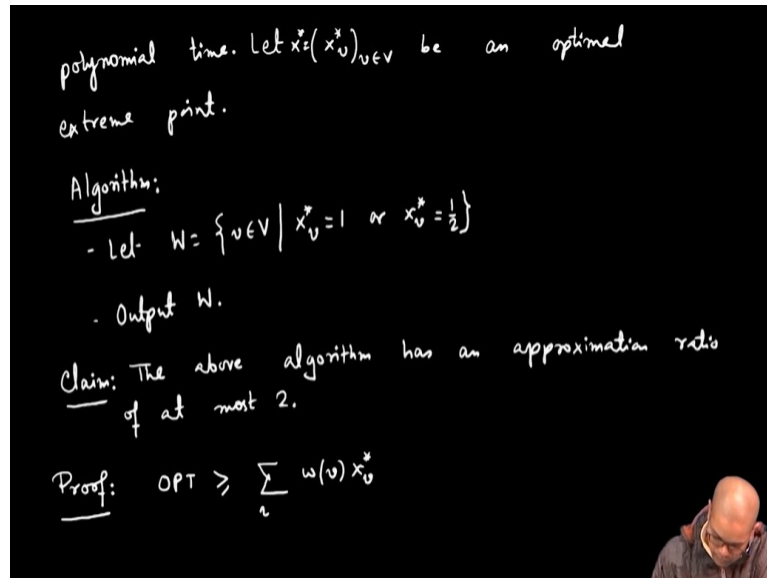
Thank you welcome so we are doing approximation algorithm design using linear programming rounding. And in the last class we have seen that the vertex cover linear program is half, integral now using that we will design a 2 factor approximation algorithm for vertex cover.

(Refer Slide Time: 00:51)



Let us recall the linear program minimize $\sum_{v \in V} w(v) x_v$ subject to each edge being, covered means for each $e = \{u, v\} \in E$ the constraint is $x_u + x_v \geq 1$. And for all vertex v , $x_v \geq 0$. And this every extreme point is half integral in particular typical algorithms for solving linear program for finding the optimum solution for linear program outputs are extreme point. So in particular we can compute and optimal extreme, point in polynomial time.

(Refer Slide Time: 03:53)



Let $x^* = (x_v^*)_{v \in V}$ be an optimal extreme point what our algorithm does is simply picks all those vertices whose corresponding variable the value is either half or 1. So algorithm let $W = \{v \in V | x_v^* = 1 \vee x_v^* = \frac{1}{2}\}$ output W . Claim the above algorithm has an approximation ratio of 2 of at most 2 proof the value of the objective function at x^* is, called this x^* is a lower bound on opt.

(Refer Slide Time: 06:50)

$$\begin{aligned}
 \text{ALG} &= \sum_{v \in W} w(v) \\
 &\leq \sum_{v \in W} w(v) \cdot 2 \cdot x_v^* \\
 &= 2 \sum_{v \in W} w(v) x_v^* \\
 &= 2 \sum_{v \in V} w(v) x_v^* \\
 &\leq 2 \text{OPT} \\
 \Rightarrow \frac{\text{ALG}}{\text{OPT}} &\leq 2
 \end{aligned}$$

So opt is greater than equal to $\sum_{v \in V} w(v)x_v^*$ on other hand ALG gives $\sum_{v \in W} w(v)$. Now these can be written as this is less than equal to for every vertex $v \in W$ the value of x_v^* is at most 1 this is $w(v)2x_v^*$. But the vertices who are not in W their x^* value is, 0 so this is $2 \sum_{v \in V} w(v)x_v^*$ but this is at most opt this is at most opt.

So we have ALG by opt is less than equal to 2 which concludes the proof this is a typical LP rounding that you solve the LP take the solution find out interesting structure and see how you can exploit it to design an approximation algorithm. So you have seen 2 algorithms for this, kind one is f factor approximation algorithm for set cover and 2 factor approximation algorithm for vertex cover next. We see another interesting technique which is called randomized rounding.

(Refer Slide Time: 09:13)

Randomized Rounding

Weighted Set Cover:-

$$\text{minimize } \sum_{S \in \mathcal{S}} c(S) \cdot x_S$$

$$\text{subject to: } \forall e \in U, \sum_{S \in \mathcal{S}: e \in S} x_S \geq 1$$

$$\forall S \in \mathcal{S}, x_S \geq 0$$

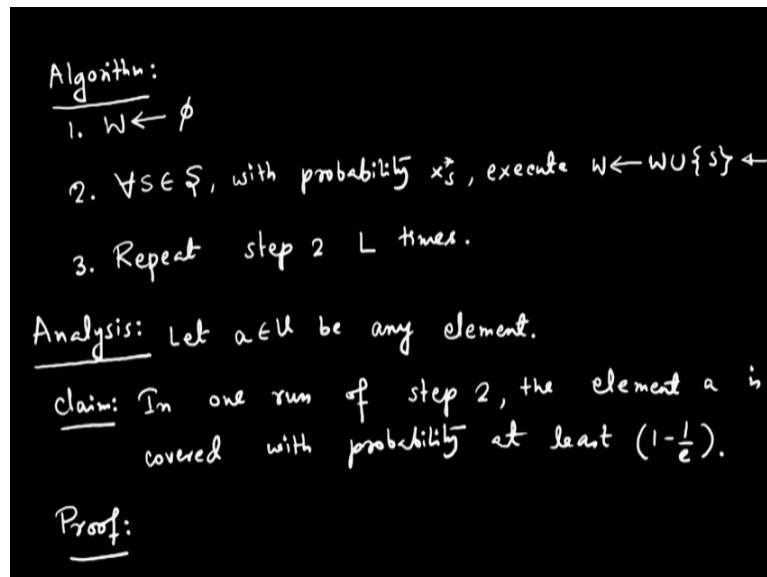
Let $x^* = (x^*_S)_{S \in \mathcal{S}}$ be an optimal solution.

$$0 \leq x^*_S \leq 1$$

So for that let us apply it on weighted set cover so let us write the linear programming for linear programming relaxation minimize summation sum of our sets in the collection cost of is times x_s , of is subject to constraint is for all element e in the universe over all sets s script S such that e belongs to s x_s is this should be greater than equal to 1. And we have for all set is in script S x_s is greater than equal to 0.

So now what we do is that we solve it so let $(x^*_S)_{S \in \mathcal{S}}$ we and optimal solution next approach is treat this numbers like x 's these are, positive numbers and for optimal solution x^* is lies in between 0 and 1.

(Refer Slide Time: 12:15)




So treat x^* as like probabilities of picking s so in particular the algorithm is for all set s in script S so construct a solution. So let w be the set cover that we are constructing initial is an empty set and for all s you know with probability x^* is execute w is $W \cup \{s\}$ means for all s is put s in the set cover, that we are constructing with probability x_s^* . And we will see that this step you know this with this just running this step once all elements are covered with constant probability.

Now we have seen in randomized algorithms that to boost the success probability we need to execute some steps execute the algorithm many times. So that has the third step repeat step 2 say l times let the, analysis dictate how we should pick n what should be the value of l so that all elements are covered with say constant probability it is a probability at least 99 percent. So analysis first is what is the probability that an element is covered by executing step 2 once?

So let so here is the claim in one run of step 2 every element let e be any element for in one run of step 2 the element e , let us call it x because you know this will come then is also. So let us call this e the element e is covered with probability at least $1 - \frac{1}{e}$ proof so what is the probability that an element is covered so is x^* .

(Refer Slide Time: 16:51)

Let a is present in $S_1, \dots, S_k \in \mathcal{S}$. The element a is covered if any one of S_1, \dots, S_k is picked.

$$\begin{aligned}
 \Pr[a \text{ is covered}] &= 1 - \Pr[a \text{ is not covered}] \\
 &= 1 - \Pr[\text{none of } S_1, \dots, S_k \text{ is picked}] \\
 &= 1 - \prod_{i=1}^k \Pr[S_i \text{ is not picked}] \\
 &= 1 - \prod_{i=1}^k (1 - x_i^*) \\
 &\geq 1 - \prod_{i=1}^k e^{-x_i^*} \quad \left[\because 1 + \lambda \leq e^\lambda \quad \forall \lambda \in \mathbb{R} \right]
 \end{aligned}$$


So let e is present in case it is in S_1, \dots, S_k in the collection so e will be covered if any one of them is picked. So the element this a if any one of S_1, \dots, S_k is, picked so probability that a is covered is $1 -$ probability that a is not covered this is $1 -$ probability a is not covered this will happen if and only if none of S_1, \dots, S_k are covered. This is probability that none of S_1, \dots, S_k is picked now each S_1, \dots, S_k is not picked independently of each other.

So this is product to k probability that S_i is not picked $1 -$ product $i = 1$, to k S_i is not picked that happens with probability $1 - x_i^*$. Now this can be written as $1 - \prod_{i=1}^k e^{-x_i^*}$ since $1 + \lambda \leq e^\lambda$ for all real number λ .

(Refer Slide Time: 20:51)

$$\begin{aligned}
 &= 1 - e^{-\sum_{i=1}^k x_i^*} \\
 &\geq 1 - e^{-1} \\
 &= 1 - \frac{1}{e}
 \end{aligned}$$

$\Pr[a \text{ is not covered after repeating step 2 } L \text{ times}]$

$$\begin{aligned}
 &\leq \left(\frac{1}{e}\right)^L \\
 &\leq \frac{1}{n^2} \quad \left[\text{choose } L = 2 \ln n \right]
 \end{aligned}$$

So we apply this on $-x_s^*$ this is equal to $1 - e^{-\sum_{i=1}^k x_{S_i}^*}$ but what is, $\sum_{i=1}^k x_{S_i}^* \geq 1$. Because x^* is a solution to this LP and S_1, \dots, S_k are the sets where this element a , belongs this is at least one. So this is greater than equal to $1 - e^{-1}$ which concludes the proof. Now each element is covered with this much probability now what is the probability that all elements is covered?

So, probability that e is not covered after repeating tape 2 l times so what is the probability that a , is covered is at least $1 - e^{-l}$. So the probability that element a , is not cover. So in one run of the algorithm of one of step 2 is at most $\frac{1}{e}$ and each run is independent so this is element a is not covered this at most $1 - \frac{1}{e}$ by e times to the power l which means $\frac{1}{e}$ times every time it is not covered.

Now we want to ensure that this probability is 1 over poly in n because now at the end we need to do a union bound over all elements. So choose l equal to $2 \ln n$ let me write this way this is less than equal to say $\frac{1}{n^2}$ this is for what is choose l equal to twice. So for l equal to $2 \ln n$ is one by n square.

(Refer Slide Time: 24:36)

Using union bound, we obtain

$$\Pr[\text{every element is covered by } w \text{ after executing step 2 } 2 \ln n \text{ times}]$$

$$= 1 - \Pr[\exists a \in U, a \text{ is not covered by } w \text{ after executing step 2 } 2 \ln n \text{ times}]$$

$$\geq 1 - n \cdot \frac{1}{n^2}$$

$$= 1 - \frac{1}{n}$$

Now the probability now do a union bound, using union bound we obtain probability that every element is covered by w after executing step 2 $2 \ln n$ times. This is 1 minus probability that there exists an element $a \in U$ such that a is not covered by w after executing

step 2 twice $\ln n$ times this is greater than equal to $1 - \frac{1}{n}$. So n times we are adding $1 - \frac{1}{n}$.

(Refer Slide Time: 27:12)

With probability at least $1 - \frac{1}{n}$, our algorithm outputs a valid set cover.

$$\text{OPT} \geq \sum_{S \in \mathcal{F}} c(S) \cdot x_S^*$$

$$\mathbb{E}[\text{ALG}] = \sum_{S \in \mathcal{F}} c(S) \cdot \Pr[S \in W]$$

$$\leq L \sum_{S \in \mathcal{F}} c(S) \cdot x_S^*$$

$$= 2 \ln n \sum_{S \in \mathcal{F}} c(S) \cdot x_S^*$$

$$\leq 2 \ln n \cdot \text{OPT} \Rightarrow \frac{\mathbb{E}[\text{ALG}]}{\text{OPT}} \leq 2 \ln n$$

So, with probability at least $1 - \frac{1}{n}$ our algorithm outputs are valid set cover okay now what is the approximation ratio approximation guarantee? So again the value of the objective functions at an optimal point that is a lower bound on optimum. So opt is greater than equal to summation is in script is cost of is times x_S^* but what is ALG? ALG is the output of the algorithm it is, a cost of the state w it is a randomized algorithm so you should be talking about expected value of ALG.

That is summation is in script is c is times probability that is belongs to w , a is picked by the algorithm. But this is submission in 1; iteration it is fig with probability x_S^* and in the worst case in each iteration and we run it for l iterations and in the worst case all the sales that you pick could be disjointed. And so this is l times at most this because if we pick the same set in 2 repetitions or 2 run of step 2 then we just store one copy this is $2 \ln n \sum_{S \in \mathcal{F}} c(S) x_S^*$.

But this is at most of this is twice learn in times out so hence we have expected value of ALG by opt is less than equal to $2 \ln n$. So it is the approximation factor is at most twice long so, we will stop here today.