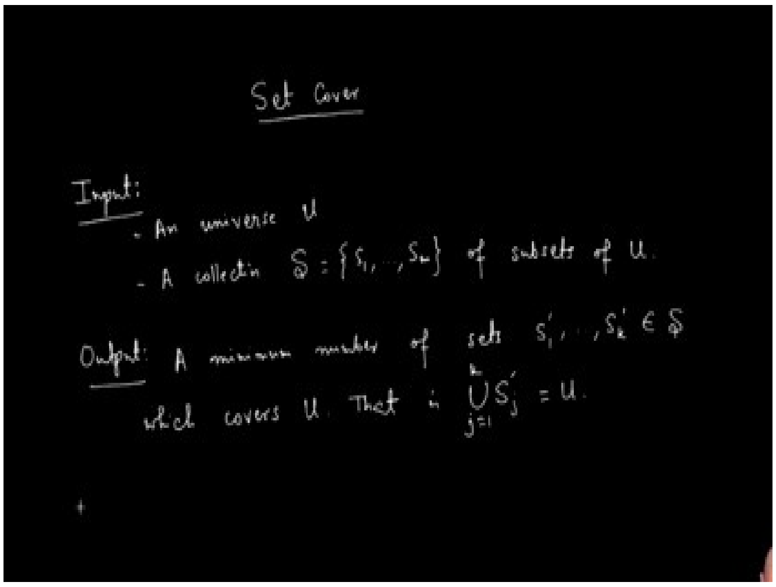
Selected Topics in Algorithm

Prof. Palash Dey
Department of Computer Science and Engineering
Indian Institute of Technology- Kharagpur

Lecture - 45
Approximation Algorithm for Set Cover

Welcome. So in the last class we have seen a $\frac{3}{2}$ factor approximation algorithm for metric TSP problem. So in today's class we will see an approximation algorithm for the set cover problem.

(Refer Slide Time: 00:44)



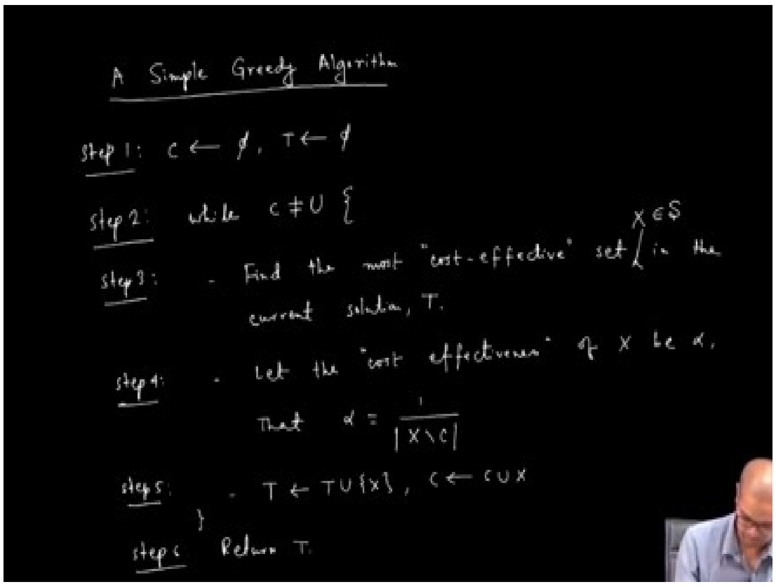
Set Cover

Input:
- An universe U
- A collection $\mathcal{S} = \{S_1, \ldots, S_m\}$ of subsets of U.

Output: A minimum number of sets $S_1', \ldots, S_k' \in \mathcal{S}$ which covers U. That is $\bigcup_{j=1}^{k} S_j' = U$.

So today's problem is set cover. So let us recall the problem statement input. A universe U. A collection of, collection is equal to $S_1, \ldots, S_m$, a collection of subsets of U. Output. A minimum number of sets $S_1', \ldots, S_k'$ from this collection, which covers U. That is $\cup_{j=1}^{k} S_j' = U$, okay. So for this problem we will see a simple greedy algorithm and we will see its analysis. So the algorithm.

(Refer Slide Time: 03:21)



A Simple Greedy Algorithm

```
Step 1:  $C \leftarrow \phi$, $T \leftarrow \phi$
Step 2:  while $C \neq U$ {
Step 3:      - Find the most "cost-effective" set $X \in S$ in the
               current solution, T.
Step 4:      - Let the "cost effectiveness" of X be $\alpha$,
               That $\alpha = \frac{1}{|X \setminus C|}$
Step 5:      - $T \leftarrow T \cup \{X\}$, $C \leftarrow C \cup X$
         }
Step 6:  Return T.
```

Or let me present the algorithm here. A simple greedy algorithm. So it is a iterative algorithm. So step 1. We build the solution iteratively. So initially I do not pick any set while, sorry this is step two. In C I maintain the set of elements which are covered. So while $C \neq U$ do this. Find the most, let me introduce the term cost effective set in the current solution is, current solution let us call it T.

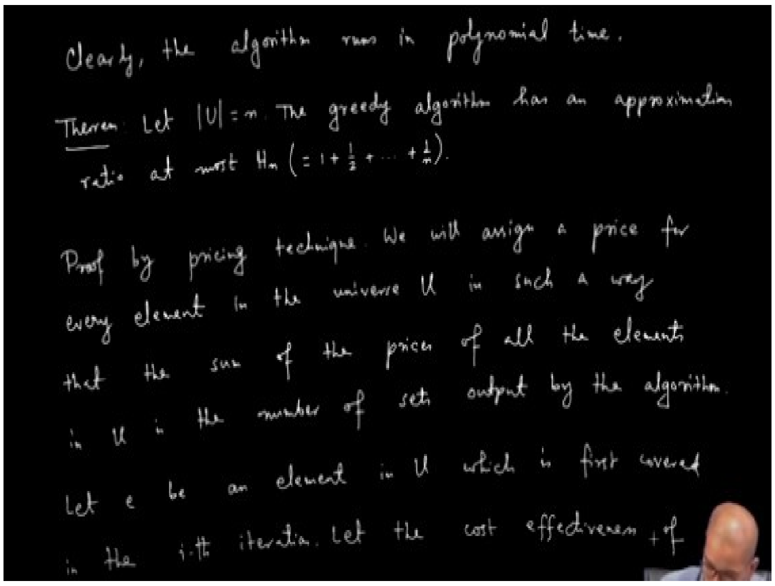
So in T we are putting sets one at a time. So here T also initialize it to empty set and find the most cost effective set you know let us call that set X, X in cal S in the current solution T. Let the cost effectiveness of X cost effectiveness of X be α. That is α equal to, now we, so α equal to 1 by, so if I pick the set X it covers all elements, but some elements are already covered.

So the cost effectiveness is the number of elements that is yet to be covered, the number of new elements that is covered by the set X. So all the elements in C they are covered. So X covered $X \setminus C$ these set of new elements. And the more number of new elements that set covers it is more cost effective. So its cost is less. So most of cost effective means whose cost is least.

That means least α. So you pick an X with least α, which is the cost effectiveness and we set we put X in our solution $T = T \cup X$, $C = C \cup X$, these are the elements covered, okay? And that is it and at the end this is step 3, step 4, step 5. Step 6 is return T. So why this is a greedy algorithm? In every iteration, it picks a set which covers as many new elements as possible.

Whichever set covers the highest number of elements which are still uncovered, those sets are picked because those sets have the least α, least cost, okay. So that way this is a greedy algorithm.

(Refer Slide Time: 09:26)



Clearly, the algorithm runs in polynomial time.

Theorem: Let $|U| = n$. The greedy algorithm has an approximation ratio at most $H_n \left(= 1 + \frac{1}{2} + \dots + \frac{1}{n}\right)$.

Proof by pricing technique. We will assign a price for every element in the universe U in such a way that the sum of the prices of all the elements in U is the number of sets output by the algorithm.

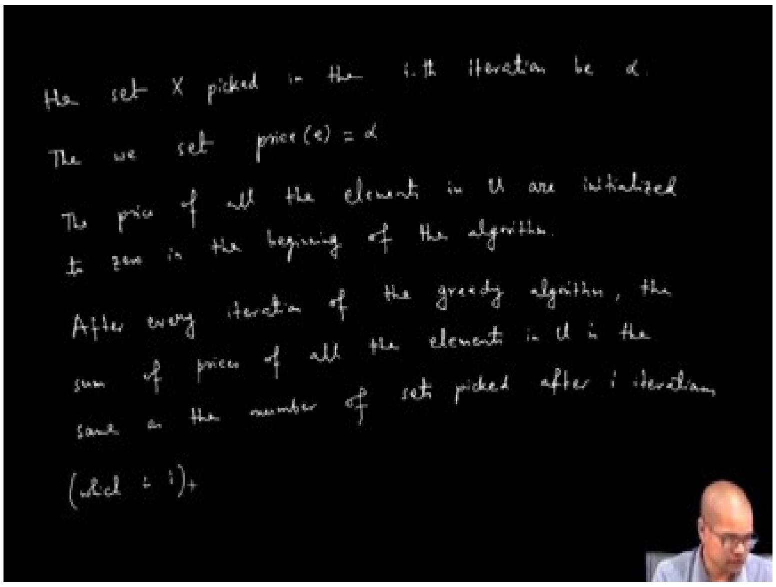
Let e be an element in U which is first covered in the i-th iteration. Let the cost effectiveness of

Clearly the algorithm runs in polynomial time, okay? So to analyze this, so now we will prove that theorem. Let cardinality U be n. Then the greedy algorithm has an approximation ratio at most H of n where H_n equal to the harmonic series up to first n term, $1+\frac{1}{2}+...+\frac{1}{n}$, okay. So to prove that we will do a pricing technique. So proof by pricing technique.

What is the pricing technique? So for each element in the universe we will set a price in such a way that the sum of the price of all the elements is the number of sets picked. So we will assign a price for every element in the universe U in such a way that the sum of the prices, the sum of the prices of all the elements in U is the number of sets output by the algorithm, okay. So what is the price?

So let e be an element in U which is first covered. That means the first time the algorithm picks a set which contains this element e which is first covered in the i-th iteration, okay. So suppose that means in the first $i-1$ iterations, the $i-1$ sets that the algorithm picked, those sets does not contain e.

(Refer Slide Time: 15:21)



the set X picked in the i-th iteration be α.

The we set price(e) = α

The price of all the elements in U are initialized to zero in the beginning of the algorithm.

After every iteration of the greedy algorithm, the sum of prices of all the elements in U is the same as the number of sets picked after i iterations

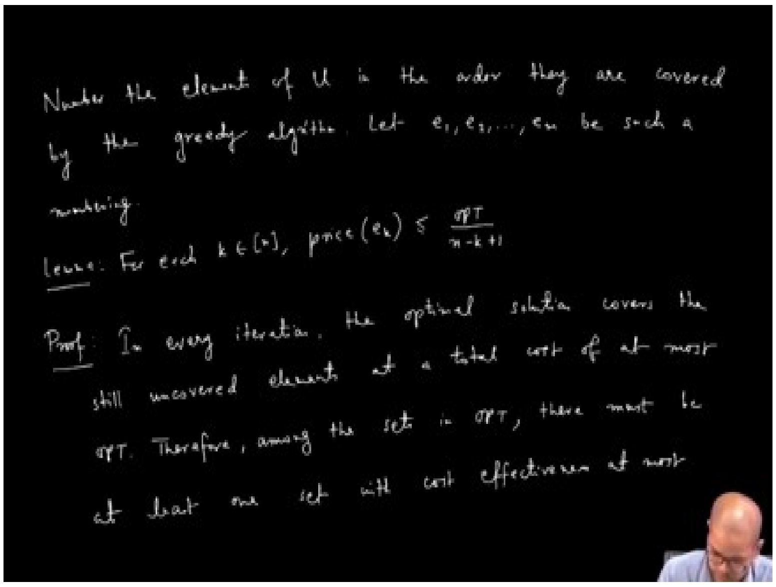
(which is i)+

So we set okay, so and let the cost effectiveness of the set X picked in the i-th iteration be α. Then we set price of e to be α. Now what are the new elements that are covered in the i-th iteration? It is $|X \setminus C|$. So the number of new elements covered in the i-th iteration is 1 by α.

And if we set the price of each element to be α, so after i-th iteration the number of sets picked has increased by one and the sum of the cost of the elements has also increased by one. Because some of the, there are one by α elements whose prices have been increased, have been set to 1 by α. So initially for the prices of all the elements are initialized to zero and hence the price of all the elements in u are initialized to zero in the beginning of the algorithm, okay.

And at that time no set is picked. So after every iteration of the algorithm, every iteration of the greedy algorithm, we pick one set and we update the prices of the newly covered elements in such a way that the sum of prices is same as the number of sets picked. So after every iteration of the greedy algorithm the sum of prices of all the elements in u is the same as the number of sets picked after i iterations which is i, okay?

(Refer Slide Time: 19:55)


Number the element of U in the order they are covered by the greedy algorithm. Let $e_1, e_2, \ldots, e_n$ be such a numbering.

<u>Lemma</u>: For each $k \in [n]$, $\text{price}(e_k) \leq \frac{OPT}{n-k+1}$

<u>Proof</u>: In every iteration, the optimal solution covers the still uncovered elements at a total cost of at most OPT. Therefore, among the sets in OPT, there must be at least one set with cost effectiveness at most

So next what we claim is that number the elements, number the elements of u in the order they are covered by the greedy algorithm. Let $e_1, e_2, \ldots, e_m$ be the numbering, be such a numbering. So what do we mean by that? Suppose in the first iteration, the algorithm picks a set, which has five elements. So it covers five elements. So what are the elements? Let us call those elements $e_1, e_2, \ldots, e_m$.

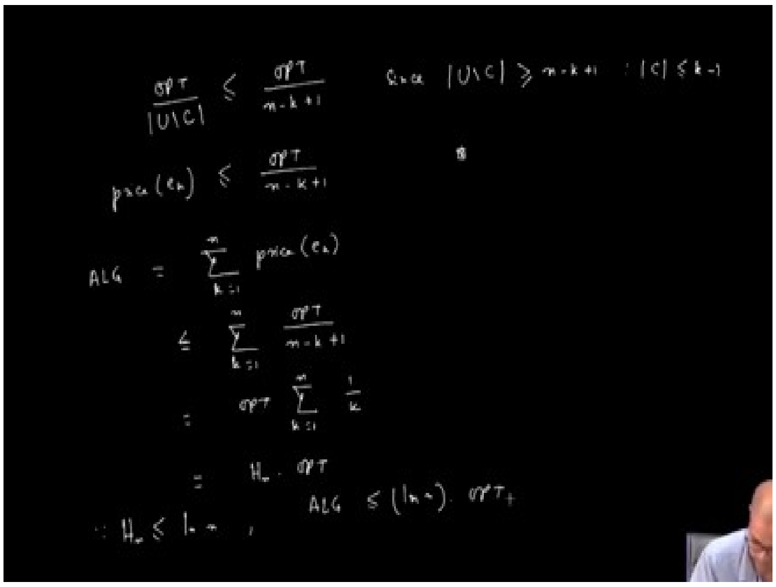
Now those five elements can be, any one of them can be called e_1 and any one is called e_2 and so on. Now in the next iteration, suppose the algorithm covers four elements, four new elements. So then let us call, we will call those elements is 6, 7, 8, 9 and so on. So that is what we mean by numbering the elements of u in the order they are covered by the greedy algorithm.

So then here is the lemma. Then we can bound the price because now the idea is ALG is summation of prices and we bound the price of each element as a function of OPT. So for, so this is the crucial lemma. For each $k \in [n]$, price of k, price of e_k is less than equal to $\frac{OPT}{n-k+1}$.

Proof. First observe that in any iteration you know some elements are yet to be covered and the algorithm covers that with OPT many number of sets, sorry the optimal solution covers those elements with OPT many number of sets. So in every iteration the optimal solution covers the still uncovered elements at a total cost of at most OPT, okay?

So maybe all elements in the optimal set may not be required to cover the still uncovered element after some iteration, but the cost is at most OPT. Therefore among the optimal set of, optimal sets, among the sets in OPT, among the sets in OPT, there must be, so look at the cost effectiveness of those sets in OPT. So there are okay, so there must be to bound the cost effectiveness of every set in the OPT. So there must be at least one set with cost effectiveness at most.

(Refer Slide Time: 26:06)



$$\frac{OPT}{|U \setminus C|} \leq \frac{OPT}{n-k+1} \quad \text{Since } |U \setminus C| \geq n-k+1 \quad \because |C| \leq k-1$$

$$price(e_k) \leq \frac{OPT}{n-k+1}$$

$$ALG = \sum_{k=1}^{n} price(e_k)$$

$$\leq \sum_{k=1}^{n} \frac{OPT}{n-k+1}$$

$$= OPT \sum_{k=1}^{n} \frac{1}{k}$$

$$= H_n \cdot OPT$$

$$\because H_n \leq \ln n, \quad ALG \leq (\ln n) \cdot OPT$$

You know total OPT is the total cost and the number of elements remaining is u minus, in the algorithm we are maintaining c, C is the set of elements covered till now. So the number of elements, the set of elements still uncovered is u – c. This is the number of elements remaining. So this is, okay. But so this is a set with cost effectiveness at most this.

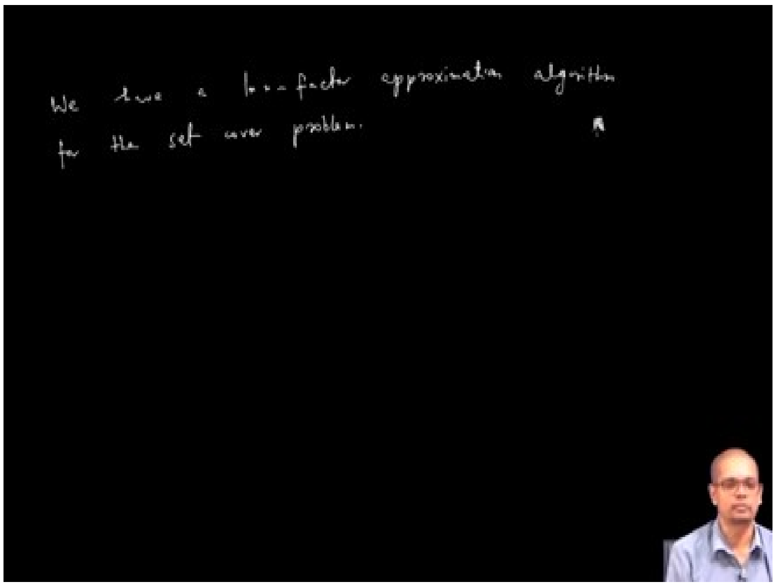
But this is you know $\frac{OPT}{n-k+1}$ because at when I am considering e k then at least that the number of elements which are remaining to be, which are yet to be covered is the at at least n – k + 1. Since, because this since $|U \setminus C|$ is greater than equal to n - k + 1. Since cardinality C is less than equal to k – 1; k - 1 elements have been covered.

And to cover k we picked the most uncovered set, the most cost effective set. So price of e_k is less than equal to $\frac{OPT}{n-k+1}$, okay. So this concludes the proof of this lemma.

Now using this we can prove, so what is ALG? ALG is $\sum_{k=1}^{n} price(e_k)$ which is price of e_k is at most $\frac{OPT}{n-k+1}$. So this is OPT by reindexing. This is $\frac{1}{k}$, k equal to 1 to n.

This is H_n times OPT. And H_n is since H_n is less than equal to $\ln n$ so you have ALG is less than equal to $\ln n$ times OPT.

(Refer Slide Time: 30:14)



We have a ln n-factor approximation algorithm for the set cover problem.

So we have a $\ln n$ factor approximation algorithm for the set cover problem. So this concludes the analysis of this approximation algorithm. Okay, so let us stop here for today.