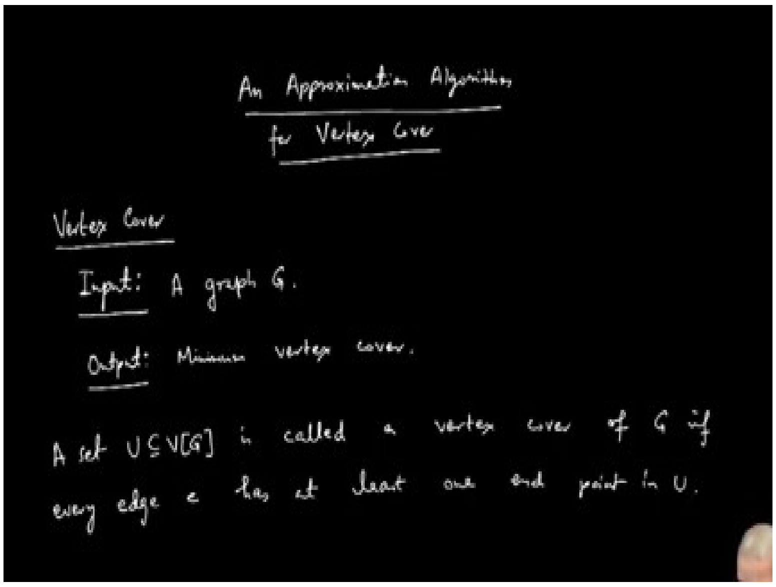
Selected Topics in Algorithm
Prof. Palash Dey
Department of Computer Science and Engineering
Indian Institute of Technology- Kharagpur

Lecture - 42
Travelling Salesman Problem

Welcome. So in the last class we have seen derandomization as a technique and we will continue our study of approximation algorithm. So today we will see an approximation algorithm for vertex cover.

(Refer Slide Time: 00:41)



An Approximation Algorithm for Vertex Cover

Vertex Cover

Input: A graph G.

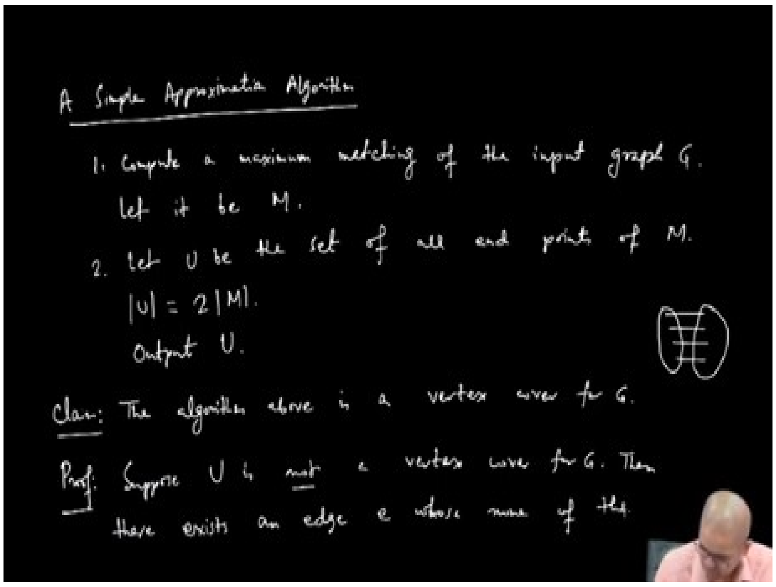
Output: Minimum vertex cover.

A set $U \subseteq V[G]$ is called a vertex cover of G if every edge e has at least one end point in U.

We have seen this vertex cover problem. We have shown that it is NP complete. We have reduced from independent set. So let us recall the problem definition, vertex cover. So what is the input? Input is a graph G. What is output? Here now we are dealing with decision problems, sorry optimization version. So output is compute minimum vertex cover, okay. So recall what is a vertex cover.

A set say u subset of the vertex set of the graph is called vertex cover of G if every edge e has at least one endpoint in U. We have seen that it is NP complete and hence we do not expect to have a polynomial time algorithm. But here is a simple approximation algorithm for vertex cover.

(Refer Slide Time: 03:09)



A Simple Approximation Algorithm

1. Compute a maximum matching of the input graph G. Let it be M.
2. Let U be the set of all end points of M.
   $|U| = 2|M|$.
   Output U.

Claim: The algorithm above is a vertex cover for G.

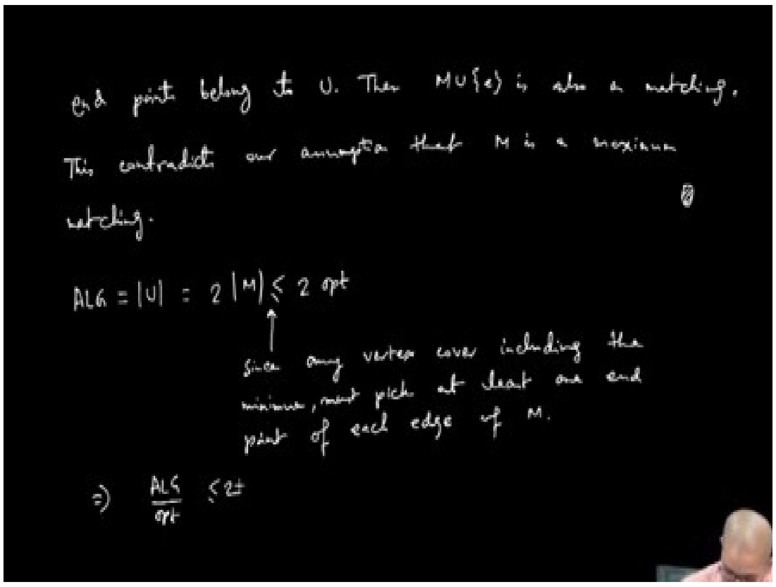
Proof: Suppose U is not a vertex cover for G. Then there exists an edge e whose none of the

Simple approximation algorithm. So first is compute a maximum matching which can be computed in polynomial time for arbitrary graph. You have seen Edmonds' blossom algorithm. So compute maximum matching of the input graph G. Let it be M. Let u be the set of all endpoints of M. So matching, let us recall what is matching. It is a collection of edges sharing no endpoint.

So it is a matching looks like a collection of edges like this. And u be the set of all endpoints. So these set of vertices and these set of vertices together forms U. That means cardinality U is twice the cardinality of M, okay and then output U. So two things, it outputs a set of vertices. First we need to prove that it is indeed a vertex cover. That means for each edge at least one endpoint is there in u.

So first claim, the algorithm above is a vertex cover for G. So it is a proof by contradiction. Suppose not. Suppose U is not a vertex cover, not a vertex cover for G. That means what? That means there exist an edge whose none of the endpoints belong to U.

(Refer Slide Time: 07:09)


end point belong to U. Then $M \cup \{e\}$ is also a matching. This contradict our assumption that M is a maximum matching.

$ALG = |U| = 2|M| \leq 2\, opt$

since any vertex cover including the minimum, must pick at least one end point of each edge of M.

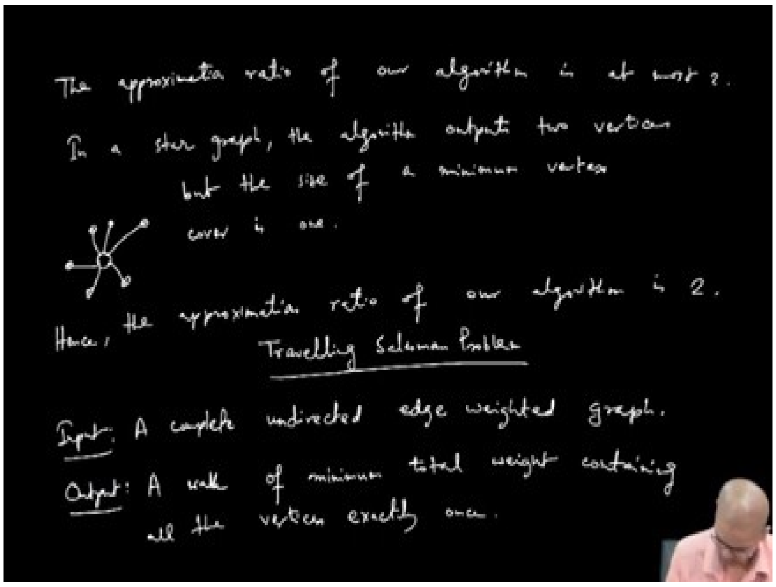
$\Rightarrow \frac{ALG}{opt} \leq 2$

Then there exists an edge e whose none of the endpoints belong to u. But then how does the edge look like? You know suppose this is matching M and there is an edge whose none of the endpoint belong to U. That means, this contradicts that this is a maximum matching, M is a maximum matching because but then $M \cup e$ is also a matching. This contradicts our assumption that M is a maximum matching.

Hence the output of the algorithm indeed forms a vertex cover of the graph. Now what is the approximation ratio? So ALG is the size of the vertex cover output by the algorithm. This is cardinality U, this is twice cardinality M. M is a maximum matching. Now we see that any optimal solution needs to cover at least these edges of the matching M and hence at least pick one of the endpoints of each of the edge.

Hence the size of opt is greater than equal to M. This is twice opt because, this is since any optimal solution or any vertex cover, any vertex cover including the optimal one including the minimum must pick at least one end point of each edge of M, okay? Hence we have ALG by opt is less than equal to 2.

(Refer Slide Time: 10:51)



The approximation ratio of our algorithm is at most 2.

In a star graph, the algorithm outputs two vertices but the size of a minimum vertex cover is one.

Hence, the approximation ratio of our algorithm is 2.

<u>Travelling Salesman Problem</u>

<u>Input:</u> A complete undirected edge weighted graph.

<u>Output:</u> A walk of minimum total weight containing all the vertices exactly once.

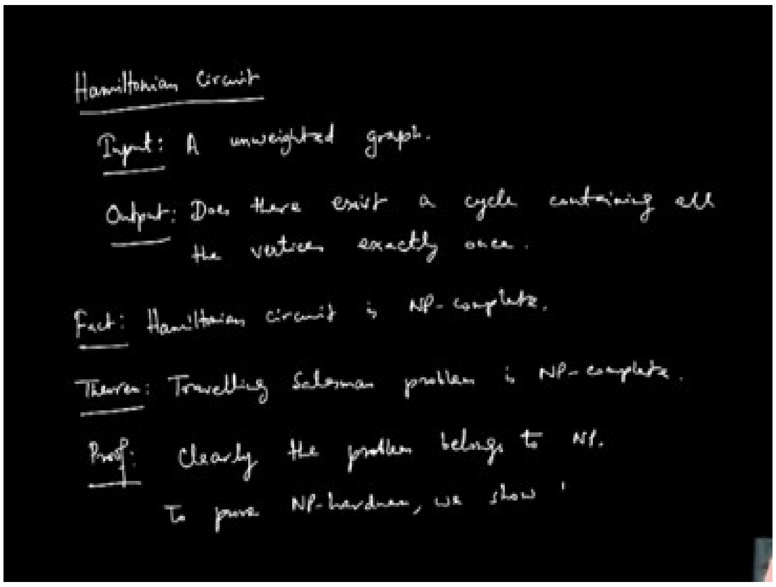
Hence the approximation ratio of our algorithm is 2. Is our analysis tight? That means is our, is the approximation ratio of our algorithm is, here it shows is at most 2. So is this analysis tight? That means, is it that the approximation ratio for algorithm is less than 2 or we will show that no. This analysis is tight that means, this there exist an instance where the output of the algorithm is twice the size of the minimum vertex cover.

So for that consider a star graph. In a star graph, how does it look like? One spoke vertex and there are lot of other vertices connected directly to it. In a star graph the algorithm output two vertices but the size of a minimum vertex cover is 1. Hence the approximation ratio is exactly 2. Hence the approximation ratio of our algorithm is 2, okay. So good.

So next we consider our next optimization problem which is very popular which is called traveling salesman problem. Traveling salesman problem. What is this problem? Input a complete edge weighted or complete undirected edge weighted graph. Output cycle of minimum total weight containing all vertices. Or it need not be a cycle. More generally it will be a walk.

A walk of minimum total weight containing all the vertices. Now this problem is studied in many ways some variations. For example, sometimes we do not assume that input graph is complete. In that case if the input graph is arbitrary edge weighted we will assume that the weight of the missing edges are infinite or very large, okay. Is the decision version of this problem NP complete?

(Refer Slide Time: 16:11)



Hamiltonian Circuit

Input: A unweighted graph.

Output: Does there exist a cycle containing all the vertices exactly once.

Fact: Hamiltonian circuit is NP-complete.

Theorem: Travelling Salesman problem is NP-complete.

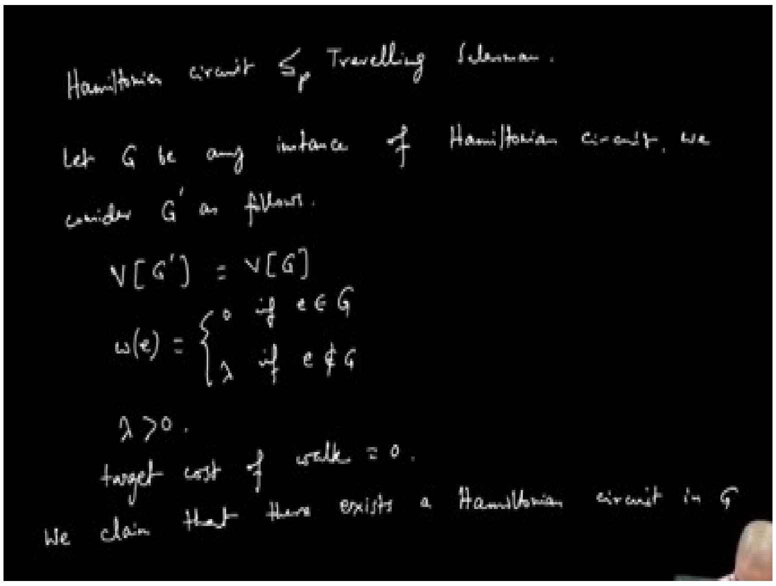
Proof: Clearly the problem belongs to NP.
To prove NP-hardness, we show '

So the decision version of the problem can be shown to be NP complete by reducing from what is called Hamiltonian circuit problem. What is the input? Unweighted graph output. It is a division version. Let us, the problem itself is a division version, decision problem. Does there exist, does there exist a cycle containing all the vertices exactly one. We will use the fact that Hamiltonian circuit is NP complete.

And using this we will prove that traveling salesman problem is also NP complete. So let me write, traveling salesman problem is of course the decision version where it does not make sense for optimization version of a problem to talk about NP completeness, being NP complete. So what is the decision version?

That means, given a complete edge weighted graph and a target of target sum of weight say t does there exist a walk covering all vertices such that the sum of the weights of the edges is at most t. So proof. So of course, the decision version, okay. So clearly I let you check that this is this belongs to NP. Clearly the problem belongs to NP. To show to prove NP hardness we show Hamiltonian circuit.

(Refer Slide Time: 20:03)


Hamiltonian circuit $\leq_p$ Travelling Salesman.

Let G be any instance of Hamiltonian circuit, we consider G' as follows.

$V[G'] = V[G]$

$$w(e) = \begin{cases} 0 & \text{if } e \in G \\ \lambda & \text{if } e \notin G \end{cases}$$

$\lambda > 0$.

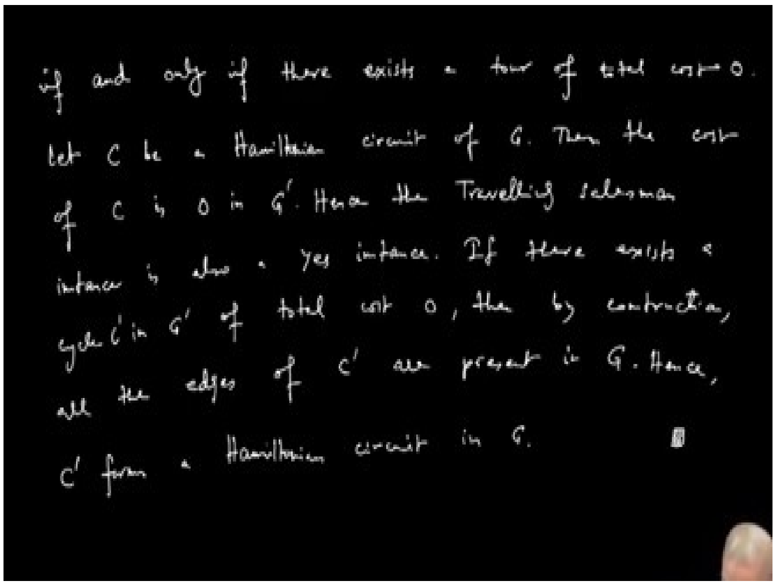
target cost of walk $= 0$.

We claim that there exists a Hamiltonian circuit in G

We show let me write, Hamiltonian circuit many to one polynomial time curve produces to traveling salesman, okay. So what is the thing? So let G be any instance of Hamiltonian circuit. We consider $G^{'}$ as follows. The vertex set is same and in $G^{'}$ we have, $G^{'}$ must be a complete graph and only weights we need to decide. Weight of an edge e is 0 if this edge e is present in G and say w for some positive number w if e does not belong to G.

Let me write some instead of w because w we are using for edges, let us call it λ. λ is greater than 0, okay? And for the decision version of the traveling salesman problem we need to give a target cost. So target cost of walk is 0.

(Refer Slide Time: 23:09)

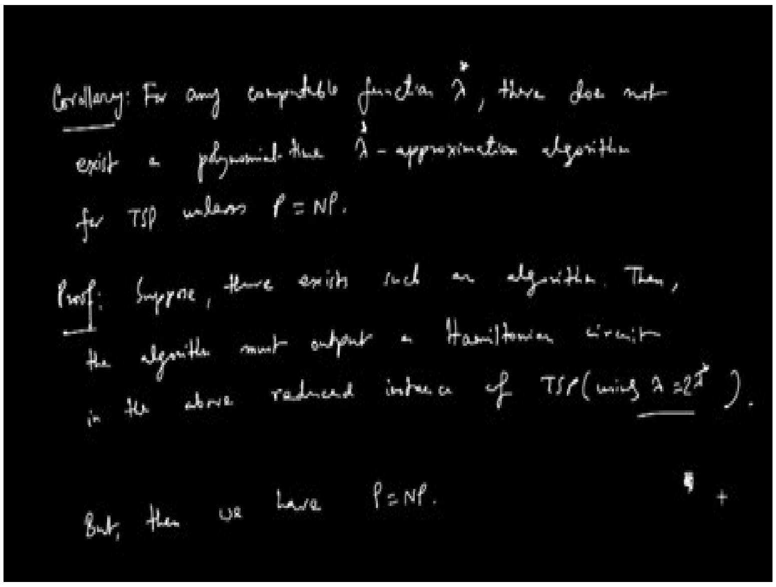

if and only if there exists a tour of total cost 0.

Let C be a Hamiltonian circuit of G. Then the cost of C is 0 in G'. Hence the Travelling Salesman instance is also a yes instance. If there exists a cycle C' in G' of total cost 0, then by construction, all the edges of C' are present in G. Hence, C' form a Hamiltonian circuit in G. ■

So we claim that there exists a Hamiltonian circuit in G if and only if there exists a tool of total cost 0. A walk containing all the vertices exactly once, okay. So of course let c be a Hamiltonian circuit of G. Then the cost of c is 0 in $G^{'}$. Hence the traveling salesman instance is also a yes instance.

On the other hand, if there exists a tour, there exists a cycle in $G^{'}$ of total cost cycle $c^{'}$ in $G^{'}$ of total cost 0, then by construction all the edges of $c^{'}$ are present in G. Hence $c^{'}$ forms a Hamiltonian circuit in G. Hence the Hamiltonian circuit instance is also yes instance. Now that is fine. So this shows that that traveling salesman problem is incomplete, but it shows more.

(Refer Slide Time: 27:05)



<u>Corollary</u>: For any computable function λ^*, there does not exist a polynomial time λ^* - approximation algorithm for TSP unless P = NP.

<u>Proof</u>: Suppose, there exists such an algorithm. Then, the algorithm must output a Hamiltonian circuit in the above reduced instance of TSP (using $\underline{\lambda = 2\lambda^*}$).

But, then we have P = NP.

It shows that, so here is a very important corollary. For any computable function say λ, so λ cannot be a constant. It could be $n, \log n, n^2, 2^n$, 2 to the power you know any computable function λ there does not exist a polynomial time λ approximation algorithm for TSP, is a popular short form of traveling salesman problem unless P equal to NP.

Proof. So suppose there exists such an algorithm. Suppose, there exists such an algorithm, then we can use this algorithm. So then in the reduced instance then the algorithm must output a Hamiltonian circuit in the above reduced instance, reduced instance of TSP using λ. So let us use some other notation λ^* , λ equal to $\frac{\lambda^*}{2}$.

Why? Because what is the value of the instance if value of the optimal solution is 0 if there exist a Hamiltonian circuit? So this we are talking about this reduced instance in this proof. And if the yeah, so it is not λ^* by 2 it is λ^* times 2, two times λ^*. See, if there exist a Hamiltonian circuit, then the value of the optimal solution is 0.

And if it is optimal solution is 0, it actually does not matter. The value of the optimal solution is 0, then ALG because ALG must be less than equal to approximation ratio times opt. Now if opt is 0, that means ALG also must be 0 for this approximation ratio to hold. Hence ALG also must be 0 and this is possible only if the algorithm outputs the Hamiltonian circuit.

But then we have P = NP. So what we have seen is that for TSP, we do not expect to have any kind of approximation algorithm however bad the approximation ratio could be, will not expect to have an approximation algorithm for TSP. So in next lectures what we will see we will make some realistic assumption on TSP and that will allow us to have an approximation algorithm for TSP. Okay. So let us stop here.