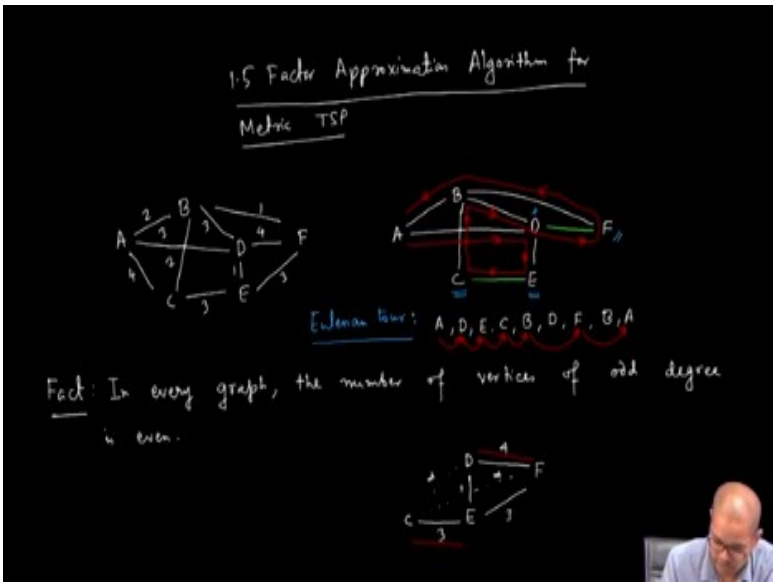
Selected Topics in Algorithm
Prof. Palash Dey
Department of Computer Science and Engineering
Indian Institute of Technology- Kharagpur

Lecture - 44
1.5 - Factor Approximation Algorithm for Metric TSP

Welcome. So in the last lecture, we have started looking at a constant factor approximation algorithm for metric TSP. We have seen a 2 - factor approximation algorithm and we have also described a 1.5 - factor approximation algorithm. So in today's class we will prove and we will analyze and see a diagram of 1.5 - factor approximation algorithm for metric TSP.

(Refer Slide Time: 00:55)



So 1.5 factor approximation algorithm for metric TSP, okay? So let us see an example be A, B, C, D, E, F. Let us put some weights, okay. So now let us compute the minimum or minimum spanning tree and let us use Kruskal's algorithm. So D is the shortest edge, shortest weightage and B F is also another shortest weightage. These two we add. Then A B, B C and then A D and then E F.

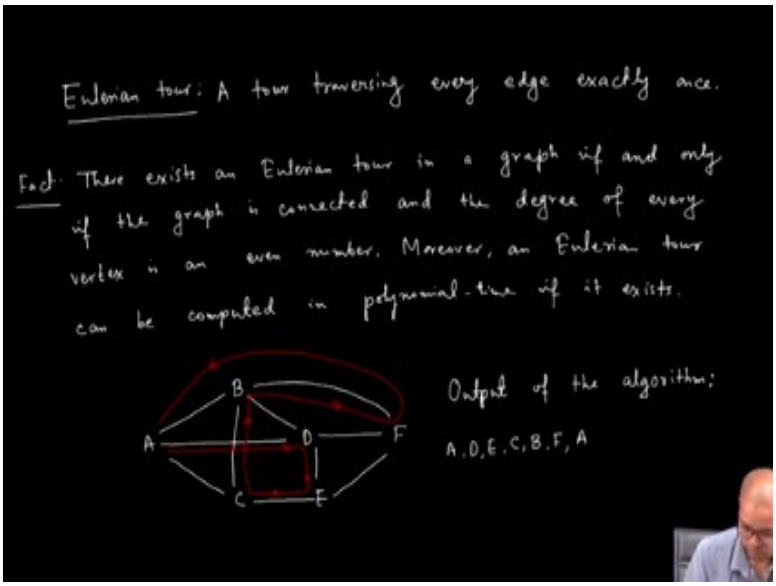
This one minimum spanning tree of course, there are other minimum spanning trees. So what we do is that we first highlight the or maybe let us take this. This will be more useful for showing the example algorithm. The other aim is to we will also work, but this will elaborate the use of this algorithm more elaborately. What we do is that we first highlight the vertices which are of odd degree.

So C is of odd degree. E is of odd degree. F is of odd degree and D, these are the four vertices. So here is a graph theoretic result which is not difficult to prove. It is quite easy to prove, but let us not prove it. Use it as a fact that in every graph the number of vertices of odd degree is even, okay. So here we have four vertices of odd degree. Next what I do is that I look at the induced graph on these four vertices.

So here are the things D E C F okay and these are the edges. So this weight is 1, this weight is 3, this weight is 3, between this is 4. And of course the missing edges we assume they are edges of weight infinity. So in this graph I compute because there are, it is a complete graph and I have even number of, even number of vertices. There always exist a perfect matching.

What I pick is a perfect matching of minimum weight. So that will be, a minimum weight perfect matching for this graph will be C E and D F, okay? So next what I do, I add these edges in this graph, in this tree. So I add the edge D F and the edge C E. Now what I have after adding the edges you see that the degree of the vertices of odd degree has increased by 1 and hence after adding all the vertices we will have even degree, okay?

(Refer Slide Time: 06:45)



And next what we will use next is what is called an Eulerian tour. What is it? A tour traversing every edge exactly once, okay? This is an Eulerian tour and here are a

couple of facts. There exists an Eulerian tour in a graph if and only if the graph is connected and the degree of every vertex is an even integer, even number. Moreover an Eulerian tour can be computed in polynomial time if it exists.

That is, the graph is connected and the degree of every vertex is an even number then Eulerian tour can be computed in polynomial time. And this is what we use next. In this graph, which is obtained by superimposing or matching on odd vertices with a spanning tree it is a connected graph because it contains a spanning tree as a subgraph.

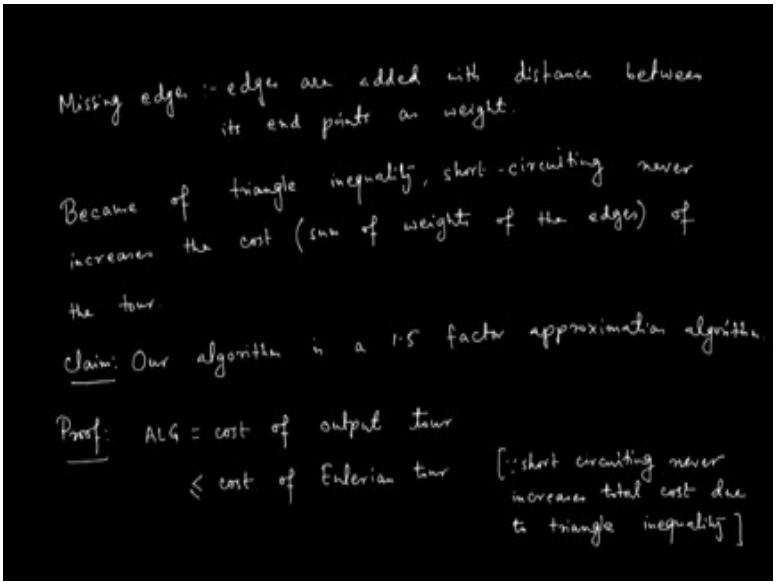
And it is a, all the, the degree of every vertex is even. So there is a there exist a Eulerian tour. And let us compute the Eulerian tour. And it is easy. So from A let us go to D. From D let us go to E. From E let us go to C. From C let us go to B. From B let us go to O. From O let us go to F. From F let us go to B and then A, come back to A okay. So what is the Eulerian tour here?

A to D, E, C, B, D, F, B and then A. Now what we do here, we do a short circuiting. For example from A I go to D because D is unvisited. From D I go to E because E is unvisited. From E I go to C because C is unvisited. From C I go to B because B is still unvisited. But from B I do not need to go to D because D is already visited. So I directly go to F, okay?

And then from A I do not need to go to B because B is already visited. I directly go to A. And because of triangle inequality, because of metric property, this short circuiting does not increase the cost of the tour. So what is the tour here? Let us redraw the graph A, B, C, D, E, F. So here are the here are all the edges. Now what is the tour? From A I go to D and then I go to E. A D E C.

So from A I go to D, D to E, E to C. Then C to B, B to F. C to B, B to F. And then return to A, okay? So this is the output. So output of the algorithm A, D, E, C, B, F, A.

(Refer Slide Time: 14:44)

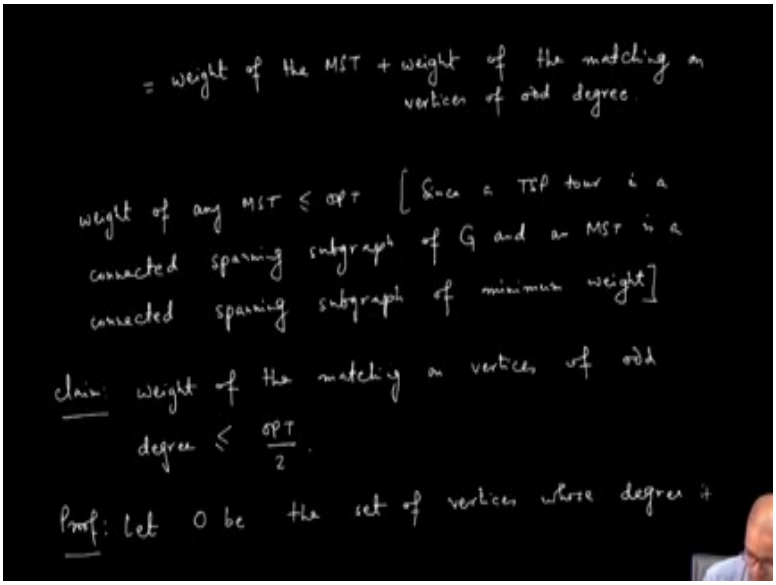


And so and because of triangle inequality so missing edges so here is the preprocessing. For missing edges, edges are added with distance between its endpoint as weight, okay? That is what we do and also note that where are we using the metric property, short circuiting. Because of metric property or because of triangle inequality, short circuiting never increases the cost of the tour of this sum of weights of the edges, okay?

Now we claim that our algorithm is a 1.5 factor approximation algorithm, okay? So first observe that even without short circuiting, the cost of our edge is bounded above by this Eulerian tour. So the ALG is the cost of output tour, this is bounded above by cost of Eulerian tour since short circuiting never increases total cost due to triangle inequality. But what is the Eulerian tour consists of?

It is the, it traverses all the edges exactly once and there are two kinds of edges. One is spanning tree edges and other is, another is this matching edges.

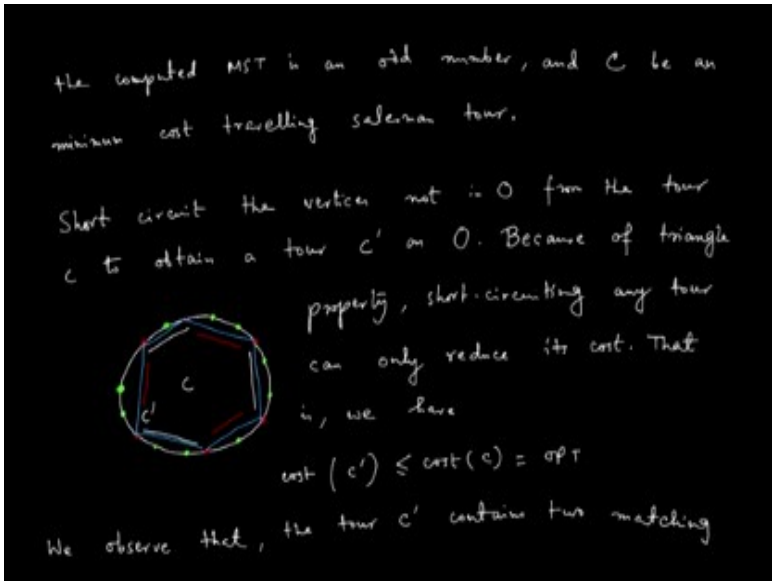
(Refer Slide Time: 19:53)



So this is weight of the minimum spanning tree plus weight of the matching on odd, on vertices of odd degree, okay? Now we have observed in our analysis of two factor approximation also that weight of an MST, weight of any minimum spanning tree is less than equal to OPT. Why, since a traveling salesperson tour, TSP tour is connected spanning subgraph of G input graph and an MST is a connected spanning subgraph of minimum weight, okay?

Next what we claim is weight of the matching on the vertices of odd degree is at most twice OPT, half OPT, at most half OPT. Matching on vertices of odd degree is less than equal to OPT by 2. Proof.

(Refer Slide Time: 24:10)



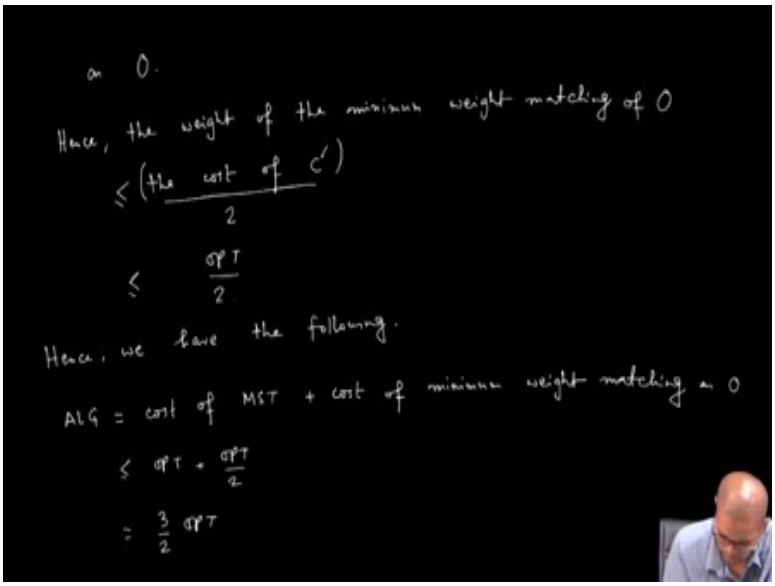
Let O be the set of vertices whose degree in the MST, in the computed MST, is an odd number and C be an optimal or minimum cost traveling salesman tour. The first step is to short circuit the other vertices and have a shorter tour from C which is on O. So short circuit the vertices not in O from the tour C to obtain a tour on O, a tour let us give it a name, tour say C' on O. So what is it? So suppose here is C.

This is an optimal traveling salesman tour. So and where are the odd vertices? Suppose these are the odd vertices, the vertices whose degree is odd. And the other are even vertices, maybe these are even vertices. Their degrees are even. So what we are saying is that from C we short circuit C and visit only odd vertices, okay? And this one we are calling it C', so this is C'.

So again because of triangle property, short circuiting any vertex, short circuiting any tour can only reduce its cost. That is we have cost of O, cost of C' is less than equal to cost of C which is the OPT, okay. Now you see that any circuit, this circuit C' contains two matchings on this red vertices, on this vertices of odd degree. One matching is this, this white edges and other is this, no this red edges.

So let us write. We observe that the tour C' contains two matchings on O.

(Refer Slide Time: 30:02)

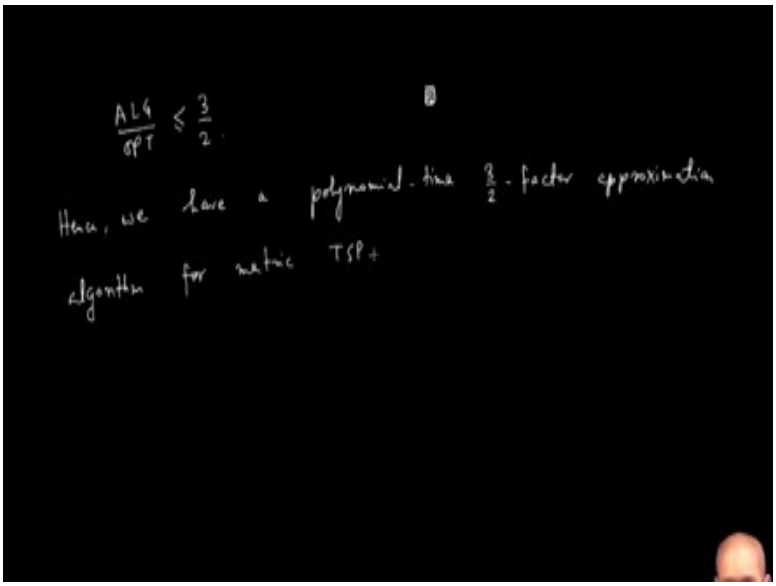


And which edges we have added? We have added a min cost matching on O. So hence the weight of the, hence the weight of the min cost matching, minimum weight

matching of O is less than equal to the cost of C'. Because simply because C' consists of two matchings on O and we have added this thing. So this is this cost of C' and it contains two matchings, so this by 2.

But this is at most OPT by 2. So hence we have the following. ALG is cost of MST plus cost of matching, minimum weight matching on O. Cost of MST is less than equal to OPT and cost of minimum weight matching on O is less than equal to OPT by 2, which is $\frac{3}{2}$ OPT.

(Refer Slide Time: 32:48)



Hence ALG by OPT is less than equal to $\frac{3}{2}$, which concludes the proof. Hence we have polynomial time $\frac{3}{2}$ factor approximation algorithm for metric TSP. Okay, so let us stop here today.