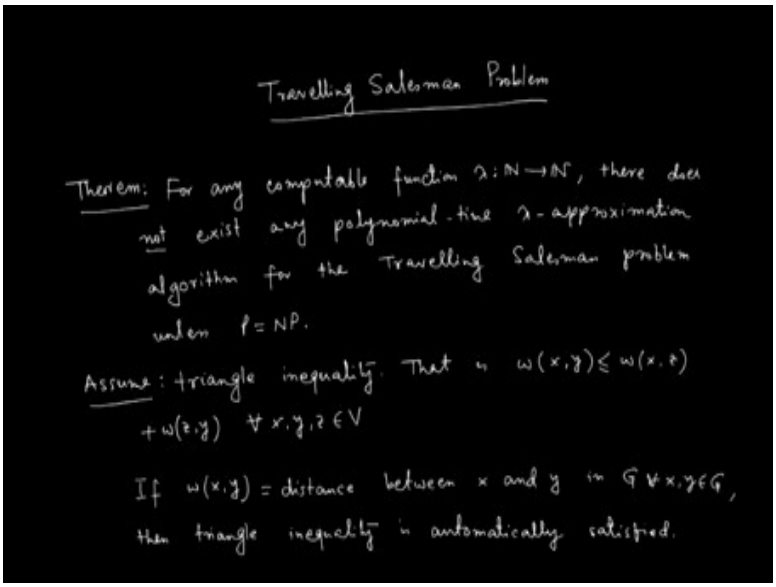
Selected Topics in Algorithm
Prof. Palash Dey
Department of Computer Science and Engineering
Indian Institute of Technology- Kharagpur

Lecture - 43
2 - Factor Approximation Algorithm for Metric TSP

Welcome. So in the last couple of lectures, we have been seeing approximation algorithms. And in the last class we have started looking at traveling salesman problem. So we will continue studying that problem in this class.

(Refer Slide Time: 00:45)



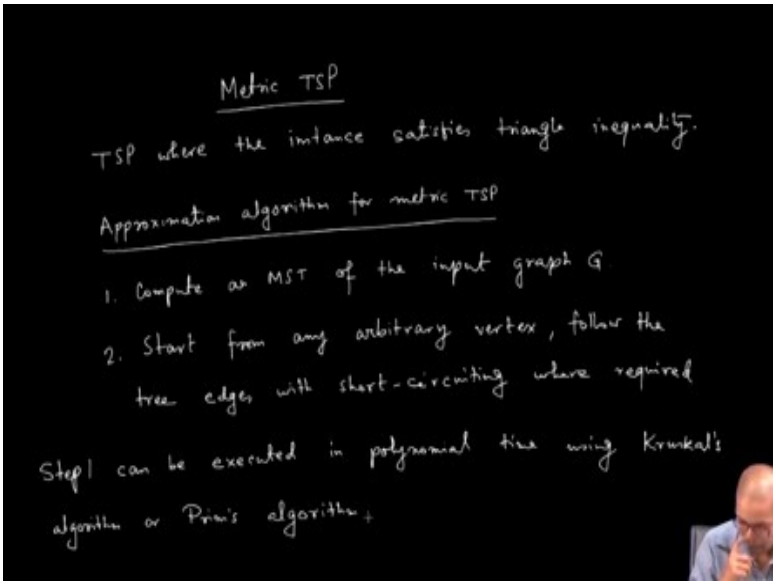
So traveling salesman problem. In the last class, we have seen that this theorem that we have proved this was the corollary of our NP completeness proof of traveling salesman problem that for any computable function lambda from natural numbers to natural numbers there does not exist any polynomial time lambda approximation algorithm for the traveling salesman problem unless P = NP.

So what we will do in today's class is that we will make some natural assumption on the traveling salesman problem and we will see that assumption allows us to have a polynomial time constant factor approximation algorithm for traveling salesman problem. So assume triangle inequality.

That is weight of x, y, weight of this edge between x and y is less than or equal to weight of x, z weight of the edge between x and z plus weight of the edge between z

and y they should hold for all vertices x, y and z in v, okay? And this natural if the weight of the edges are the distance between x and y. Now if weight of x, y is the distance between x and y in G for all x, y in G then triangle inequality is automatically satisfied, okay. So this version of traveling salesman problem is called metric TSP.

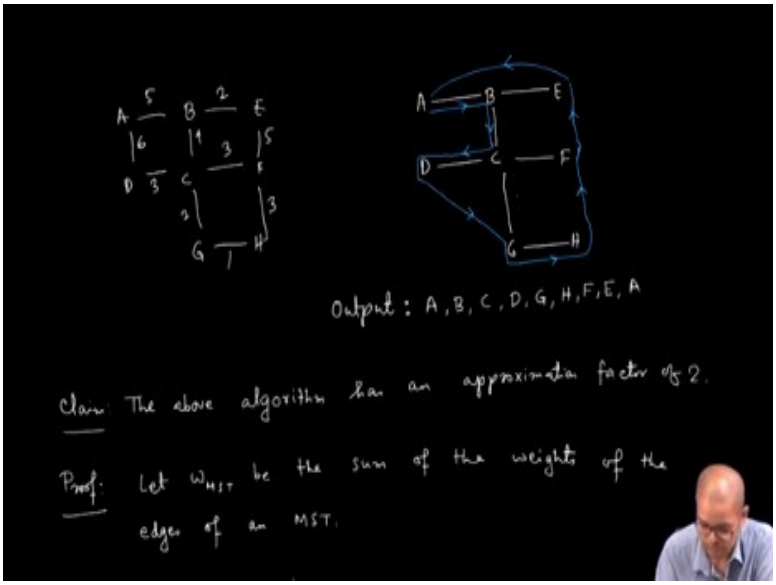
(Refer Slide Time: 05:53)



This is TSP traveling salesman problem where the instance satisfies triangle inequality, okay? So for this there is a very simple algorithm, approximation algorithm and here is the algorithm, approximation algorithm for metric TSP. So first compute a MST, minimum spanning tree, of the input graph G. Start from any arbitrary vertex. Follow the tree edges with short circuiting where required. So that is the algorithm.

So let us explain step two. Step one can be computed in polynomial time, step one can be done in polynomial time. So let me write. Using there are many algorithms, many polynomial time algorithms for finding a minimum spanning tree of a graph for example Kruskal's algorithm or Prim's algorithm, okay. Now what is step two? Let us take an example and explain.

(Refer Slide Time: 10:20)



So let us take a, suppose this is my input graph. The first step is to compute a minimum spanning tree. Let us compute it using Kruskal's algorithm. Kruskal's algorithm picks the edges in non-decreasing order of their weights and adding to the solution unless it creates cycle with the existing set of edges. So the edge with the shortest or smallest weight is G H is 1 and then the second shortest is between C and G that I add.

The third one is between B and D that I add. Then the fourth one is between C and F and C and D and also F and H but I do not add F and H because that will create a cycle and between B and C is the four. And the next is E and F, that I do not add and I add A and B. So this is sort of one minimum spanning tree.

Now what is what I mean by starting from a vertex and following as close follow the tree with short circuiting whenever required. So let us start at A maybe and I follow the tree H. I go to B and I see it is two neighbors, which what is unvisited. So both its neighbors are unvisited. So from B, I can either go to E or go to C. So let us pick any and let us go to C.

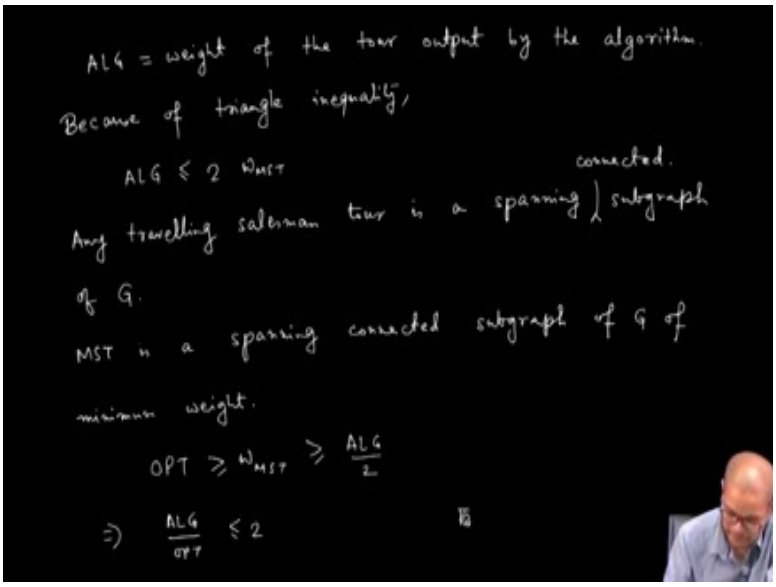
Then it has three neighbors B, F and G, all are unvisited. So I can pick any. Let us go to D. So from D it has only one neighbor C, but it is visited. So let us sort of virtually travel to see. And then what is, I go to C and from C what I do is that I have two unvisited vertices, neighbors F and G, so let us go anywhere. So from C I can go to say let us get G. But now I do a short circuiting.

Instead of from D go to G via C. So instead of going to G via C I take a shortcut and go to G. And because of triangle inequality, the cost of going to G directly from D is at most the cost of going to G from D via C. So from G again I go to H. Then again from H coming back, I sort of see its neighbor. It is I am looking for unvisited node. G is visited, it has one neighbor.

C is also visited and C has one neighbor A, it is unvisited. So I do a short circuiting and from H I go to F. Now from F again I do the same analysis. F has only one neighbor C. From C I go to B. It is also visited and I go to E. E is not visited, so I go to E and then similarly I return to A. So the output is these two. A then B then C. Let us do an arrow. A, B, C, D, G, H, F, E then A again.

Now we will show that a claim, the above algorithm has an approximation factor, has an approximation factor of 2. Proof. Very easy. So let w MST be the sum of the weights of the edges of an MST. So what is our, what is the value, what is the sum of the weights of the edges of the two output?

(Refer Slide Time: 17:25)

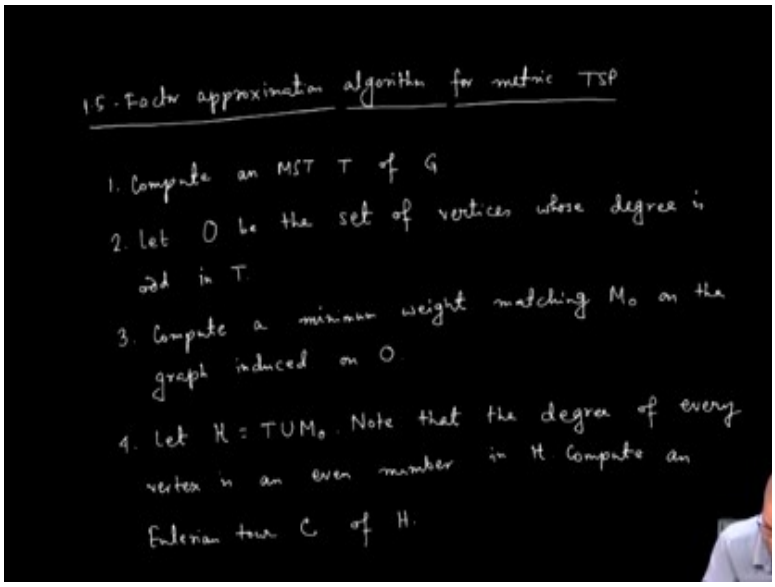


So that is ALG. ALG is the weight of the tour output by the algorithm. Then can see that now so this tour is less than this red tour which simply follows the tree does not do the short circuiting and visits every edge of this MST exactly twice. Because our algorithm does the short circuiting and because of triangle inequality, because of triangle inequality ALG is less than equal to twice w MST.

On other hand, if you look at any tour and delete the last edge, so it visits all the, in any tool it visits all the vertices and then come back. If you delete the last page you know it is a MST. So any tour, any traveling salesman tour is a spanning subgraph, spanning subgraph of G. That means it is a subgraph of G and it has all the vertices in it spanning connected subgraph.

And MST is a spanning connected subgraph of G of minimum weight. Hence we have OPT is greater than equal to weight of MST which is greater than equal to, weight of MST is greater than equal to ALG by 2. Hence $\frac{ALG}{OPT}$ is less than equal to 2 which concludes the proof. Next what I do is improve this to a 1.5 factor approximation and the idea is same. So let me write down an algorithm.

(Refer Slide Time: 21:56)

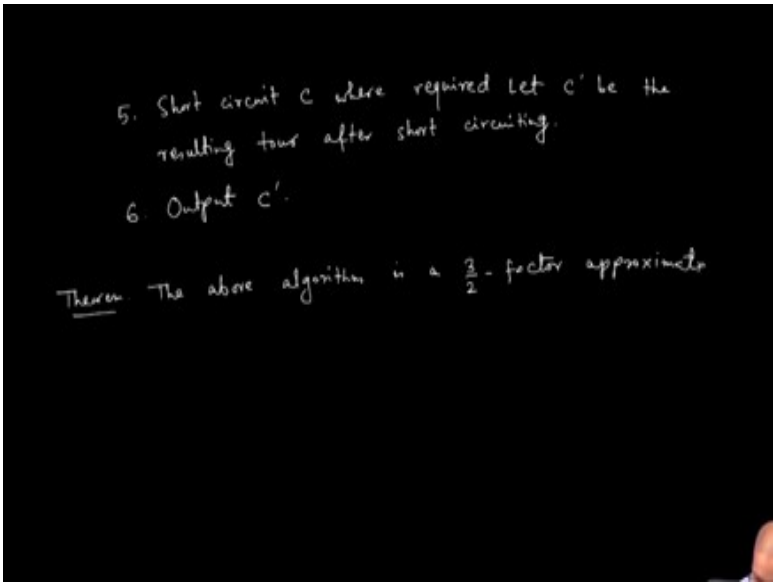


1.5 factor approximation algorithm for metric TSP. So what is the algorithm? The first step is again same. So compute an MST T of G. Then look at the set of vertices in T which are of odd degree in T. Let O be the set of vertices whose degree is odd in T. Compute a min cost matching, minimum weight matching $M(O)$ on the graph induced on O, okay. Now add those matching edges on T.

So if I add the matching edges on T all the vertices will have even degree. So let H be $T \cup M(O)$. It is those on the tree, those vertices are there, the trees edges are there and I have also added a matching edges. So note that the degree of every vertex is an even

number in H. Now it is a graph theoretic result that in a graph where the degree of all vertices are even and if it is connected, then there exist a Eulerian tour. a tour which visits all the edges of the graph exactly once.

(Refer Slide Time: 26:50)



So compute an Eulerian tour C of H with short circuit C where required. Let $C^{'}$ be the resulting tour after short circuiting. Then output is $2C^{'}$. So we claim, so here is the theorem, the above algorithm is a $\frac{3}{2}$ factor approximation algorithm.