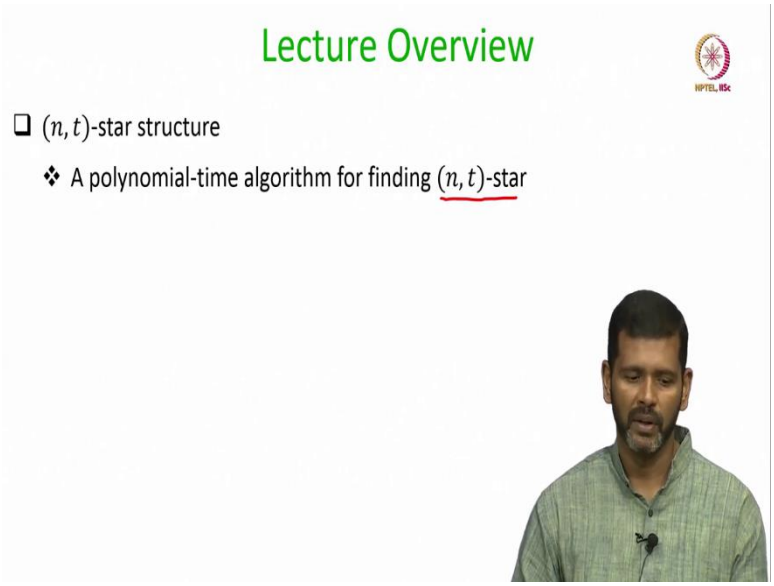
Secure Computation: Part II
Prof. Ashish Choudhury
Department of Computer Science and Engineering
Indian Institute of Science, Bengaluru

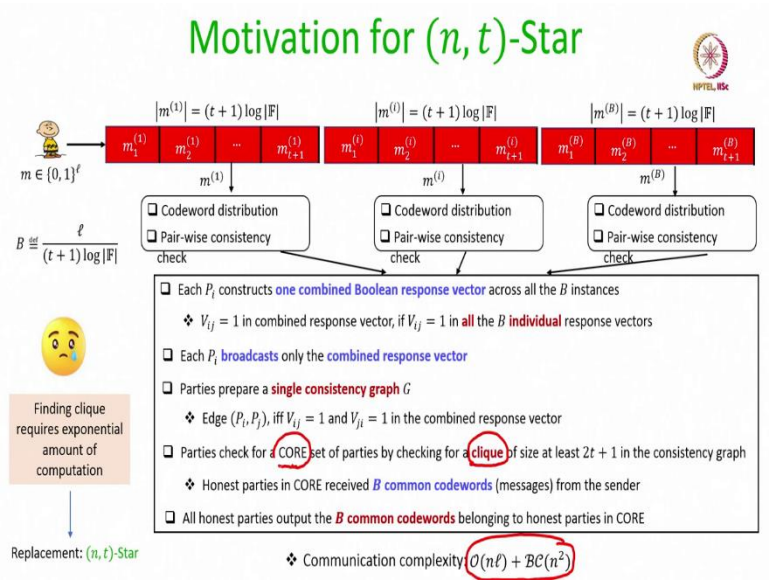
Lecture - 29
(n, t)-star Structure

(Refer Slide Time: 00:25)



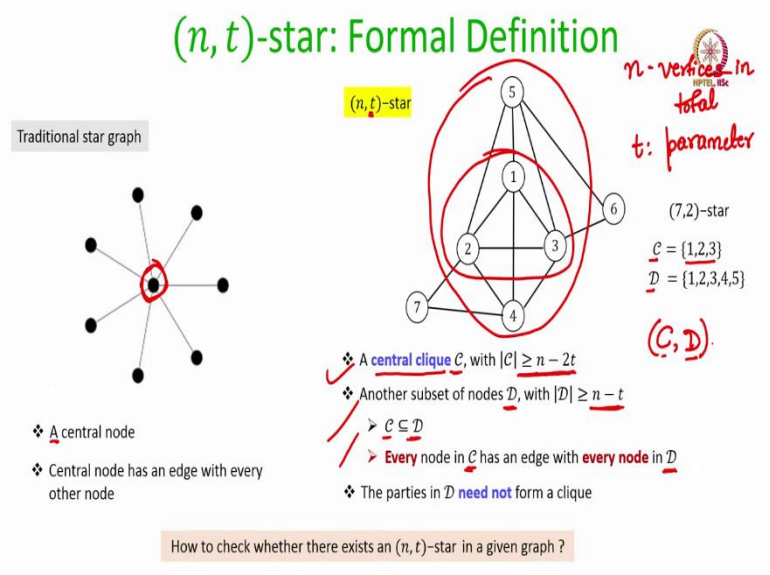
Hello everyone. Welcome to this lecture. So, in this lecture we will see (n, t)-star structure and a polynomial time algorithm for finding (n, t)-star structure in a graph. The main reason for studying (n, t)-star structure is that later we will use it to get to get an to get a domain extension protocol with polynomial computation cost.

(Refer Slide Time: 00:48)



So, recall the domain extension protocol that we had discussed in the last lecture with asymptotically communication cost of order of $n\ell$ bits. There the parties need to check whether there exist a clique of size at least $2t + 1$ in the consistency graph and that step requires the parties to perform exponential amount of computation. The main motivation for studying (n, t)-star is that we will instead check whether a core exists or not by checking whether an (n, t)-star is present in the graph or not.

(Refer Slide Time: 01:34)



So, let us first formally define what do we mean by (n, t)-star structure. The traditionally in a graph a star is defined as follows. We have a central node, a single central node to be more specific and that central node has an edge with every other node in the graph. The other nodes, the non-central nodes they may or may not have an edge among themselves; that is not required, but the central node has an edge with every other non-central node.

Now, when we say (n, t)-star structure in a graph then we are basically talking about 2 sets, a set $\mathcal{C}$ and a set $\mathcal{D}$ and they should have the following properties. The set $\mathcal{C}$ should have at least $n - 2t$ vertices, where t is a parameter here for the (n, t)-star. You have n vertices in total and t is a parameter and we say that a pair of nodes $\mathcal{C}$, $\mathcal{D}$ constitute an (n, t)-star if the following conditions hold.

The set $\mathcal{C}$ should have at least $n - 2t$ vertices, and it should constitute a clique. Namely, all the vertices within the subset $\mathcal{C}$ should have an edge among themselves. So, it is like saying the following, in traditional graph we have a single central node, but now we have $n - 2t$ central nodes.

All of them have an edge among themselves and then we have non-central nodes denoted by the set $\mathcal{D}$, whose cardinality should be at least $n - t$, such that the subset of nodes in $\mathcal{C}$ should be a subset of nodes in $\mathcal{D}$.

And, every node in $\mathcal{C}$ should have an edge with every node in $\mathcal{D}$. However, it is not required, it is not necessary that the parties in $\mathcal{D}$ should have an edge among themselves; that means, there could be few parties in $\mathcal{D}$ who are not in $\mathcal{C}$ and they have no edge among themselves, that is fine. The only requirement is that every party in $\mathcal{C}$ or every node in $\mathcal{C}$ to be more specific, every node in $\mathcal{C}$ should have an edge with every node in $\mathcal{D}$.

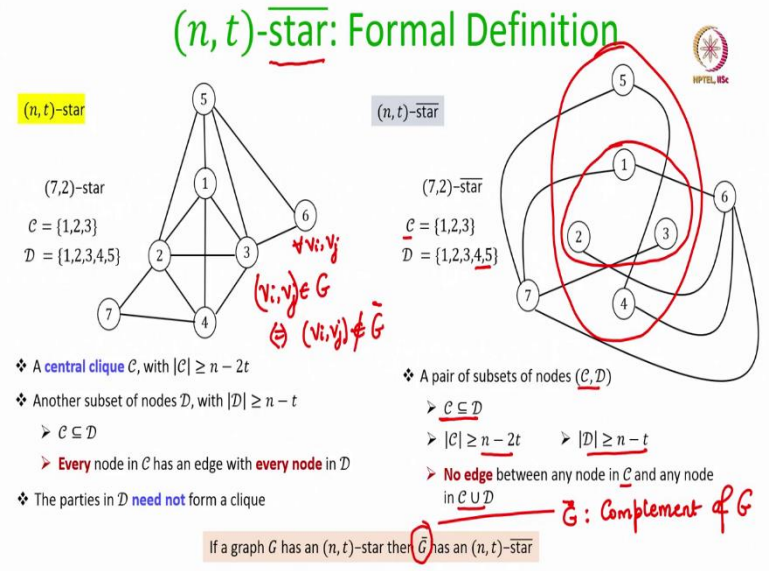
So, as an example here we have a (7, 2)-star. We have 7 nodes and you can see that I can take this triangle {1, 2, 3} as the central clique and the subset {1, 2, 3, 4, 5} as the bigger subset $\mathcal{D}$. So, every node within the subset $\mathcal{C}$ has an edge with every other node in the subset $\mathcal{C}$; that means, they constitute a clique. And, you see between every node in $\mathcal{C}$ and every node in $\mathcal{D}$ there is an edge. However, not all the nodes in $\mathcal{D}$ have an edge among themselves.

So, for instance there is no edge between the node 4 and 5, even though they are part of $\mathcal{D}$. So, that is the definition of an (n, t)-star. It is not necessary that every graph has an (n, t)-

star. The graph may or may not have an (n, t)-star. Now, the next question is how do we check whether a given graph has an (n, t)-star or not and that too in polynomial amount of time? Of course, we can have an exponential time algorithm to check whether a given graph as an (n, t)-star or not.

Check every candidate $\mathcal{C}$ and $\mathcal{D}$ subsets of size $n - 2t$ and $n - t$ and check whether these conditions are satisfied or not. If at all a graph has an (n, t)-star then for 1 candidate $\mathcal{C}$ and 1 candidate D, you will get that all these conditions are satisfied and you will be able to find your (n, t)-star t. But, going over all candidate subsets $\mathcal{C}$ and $\mathcal{D}$ will require exponential amount of computation. We are not interested in that algorithm, we want a polynomial time algorithm.

(Refer Slide Time: 06:19)



So, before going into the polynomial time algorithm we will see a related structure in the graph; this is complement of (n, t)-star. So, a complement of (n, t)-star will also be a pair of subsets of nodes $\mathcal{C}$ and D. The cardinality of $\mathcal{C}$ should be at least $n - 2t$. The cardinality of $\mathcal{D}$ should be at least $n - t$. The subset C, the set $\mathcal{C}$ should be a subset of D. But, now there should be no edge between any node in $\mathcal{C}$ and any node in $\mathcal{C} \cup \mathcal{D}$. So, an example of complement of (7, 2)-star is this graph.

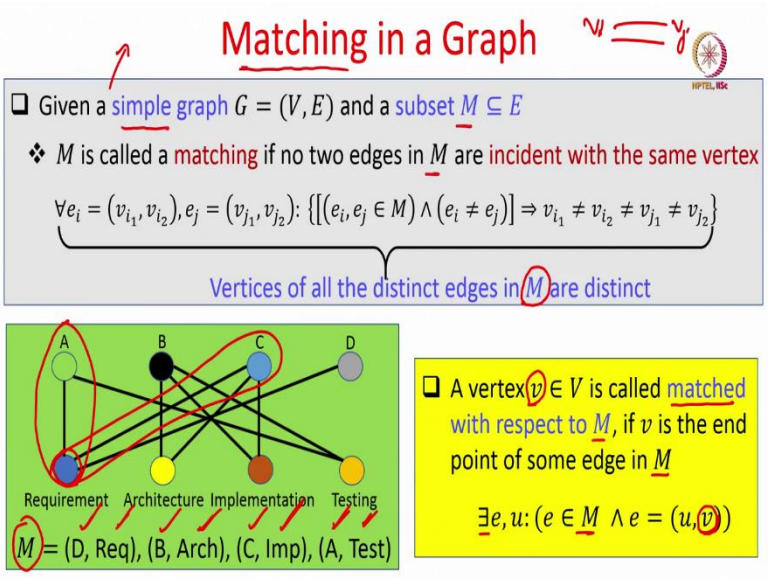
So, I can take the set $\mathcal{C}$ to be {1, 2, 3}. So, you can see this triangle graph {1, 2, 3} has no edge; well it is not a triangle graph. If I take the subset {1, 2, 3} then it there is no edge between 1 and 2, 2 and 3 and 1 and 3 and I can take {1, 2, 3, 4, 5} as the subset D. So,

there is no edge between any node in $\mathcal{C}$ and any node in $\mathcal{C} \cup \mathcal{D}$. Of course, there might be now edges between a pair of nodes in $\mathcal{D}$, who do not constitute a part of $\mathcal{C}$.

So, for instance 4 and 5, they are constitute a part of $\mathcal{D}$, but not a part of $\mathcal{C}$ and there is an edge between them. So, if you see closely the definition of (n, t)-star and complement of (n, t)-star; we can clearly say that if your graph G has an (n, t)-star then in the complementary graph G, you will have a complement of (n, t)-star.

So, this G complement denoted by this $\bar{G}$ denotes the complement of graph G. And complement of graph G means, if you have the edge between v_i and v_j in graph G, then the edge (v_i, v_j) will not be there in $\bar{G}$ and this is an if and only if. That means, whichever edges for every distinct, for every distinct (v_i, v_j); if there is an edge between those 2 nodes in the graph G then that edge will be missing in $\bar{G}$ and vice versa. So, it is like your set complement.

(Refer Slide Time: 09:05)



Now, before going into the algorithm for finding the (n, t)-star, we will also go through the concept of matching in a graph which will be required in the algorithm. So, what is the matching? So, you are given a simple graph here and for simplicity assume the graph is undirected. So, simple graph means between every pair of nodes you have at most one edge. You do not have multiple edges between v_i and v_j in the graph.

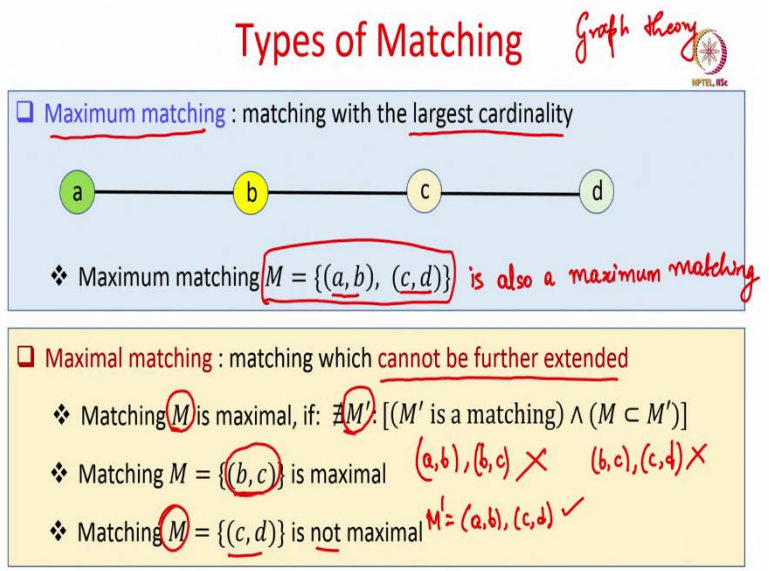
You cannot have more than one edge, that is not allowed in a simple graph. So, assume you are given a simple graph and a subset of edges M. Then, that subset of edges M is called a matching if node two edges in M are incident with the same vertex; that means, node two edge in M share a common vertex or in other words vertices of all the edges in M are distinct. So, let us see some examples of matching.

So, suppose this is a simple graph, then I can take this collection of edges; I can consider this collection of edges as a matching. The edge D requirement, B architecture, C implementation and A testing right. And, if you see the end points of this collection of edges they are all distinct. Whereas, if I would have taken the edges A and requirement and requirement and C, then that does not constitute a matching because there are two edges sharing a common vertex.

Now, if you are given a matching, then with respect to that matching a vertex in the graph is called matched, if it is the end point of some edge in that matching. Namely, there should exist some edge present in that matching such that one of the end points of that edge is the node v. If such an edge e exists, then we say that the vertex v is matched; otherwise we say that the vertex v is not matched.

So, for instance if I take this matching, then D is matched, requirement is matched, B is matched, architecture is matched, C is matched, implementation is matched, A is matched and testing; that means, all the nodes are matched here actually. But, that need not be always the case, that depends upon your graph and matching that you are considering.

(Refer Slide Time: 11:50)



Now, there are different types of matching. Maximum matching means a matching with the largest cardinality. So, for instance if I take this simple graph, then if I take the edges (a, b) and (c, d) then it constitutes a maximum matching. Whereas a maximal matching means a matching which cannot be further extended. What does that mean? That means, I cannot add furthermore edges in this matching M; so, that the resultant collection M' is also a matching. In that sense M is maximal; that means, it cannot be expanded further.

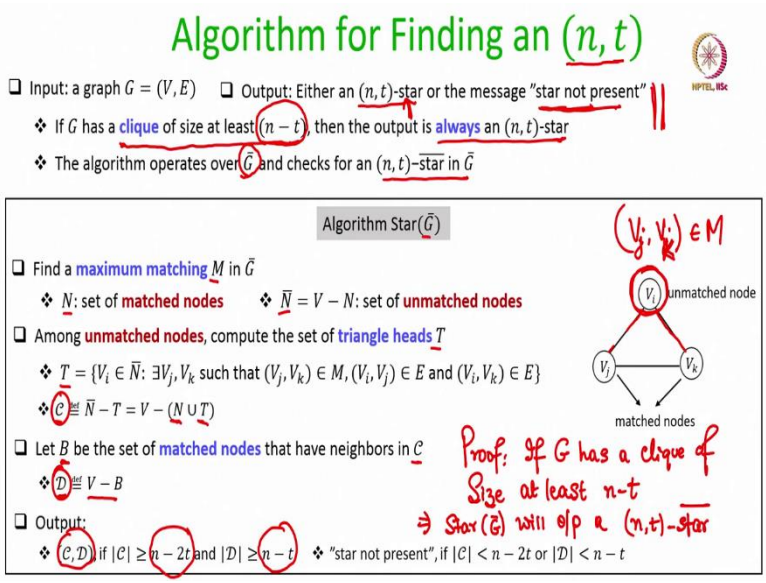
So, for instance in the same graph if I take just the edge b, c it is maximal because to b, c I cannot add the edge (a, b). If I add the edge (a, b) along with b, c it does not constitute a matching. And, in the same way to the edge b, c I cannot add the edge (c, d) because that does not constitute a matching. So, in that sense the edge b, c is maximal, but that is not maximum.

In the same way the edge (c, d) is not maximal because if I include the edge (a, b) in this collection of edges which is just the edge (c, d) then I still get a matching whose size is more than this M. So, that is why the edge (c, d) is not a maximal matching. So, now, we can simply say that every maximum matching is maximal, but other way around need not be true. For instance, this matching is also a maximum matching whereas, this collection of edges is a maximal matching, but not maximum.

Now, there are well known polynomial time algorithms to find out the maximum matching in a graph, a maximal matching in a graph and so on. This course is not about graph theory.

So, you are referred to any standard text on graph theory to know more about those algorithms. We will assume that we have polynomial time algorithms for finding maximal and maximum matching in a given graph.

(Refer Slide Time: 14:38)



Now, coming to our main goal, we want to find out whether there exists an (n,t)-star in my given graph or not. So, the algorithm is as follows. It takes as input a graph and it outputs either an (n,t)-star or the message star is not present in the graph and the algorithm ensures the following.

It ensures that if in your graph it is guaranteed that a clique of size at least $n-t$ is present, then the output is always an (n,t)-star, that does not mean the following. It could be possible that there is an (n,t)-star in the graph, but your graph does not have a clique of size $n-t$. In that case, it might be possible that you obtain the message star not present. So that means, the negative answer, star not present cannot be trusted always.

It could be possible that even though a star is present in the graph, the algorithm may end up saying that a star is not present. However, you can trust or you are guaranteed that if in your graph a clique of size at least $n-t$ is guaranteed, then the output will always be a star. You will not get the error message. And, looking ahead when we will use this algorithm in our domain extension protocol, polynomial time domain extension protocol, this property is sufficient.

We do not always want to ensure that the party should always obtain a star if it is present in the graph. It is fine if they do not get the message star, if they get the message star not present when the graph does not have a clique of size $n - t$. Looking ahead, it will be ensured in the domain extension protocol, that if the sender is honest in the broadcast protocol, then in the consistency graph a clique of size at least $n - t$ will be present.

And, now when the parties run this star finding algorithm, they will always obtain one (n, t)-star. They will not get the message that star is not present and they will not discard the sender. So, that is why this algorithm is not a full-fledged algorithm for finding an (n, t)-star. It will give you an (n, t)-star provided it is guaranteed that the graph has a clique of size at least $n - t$. However, it could be possible that the graph does not have a clique of size $n - t$, but it has a star and the algorithm fail to find that, that is quite possible.

So, how the algorithm operates? The algorithm operates over the complement of the graph and checks whether there is a (n, t)-star complement in $\bar{G}$. If there is a (n, t)-star complement in $\bar{G}$, that automatically implies that the same (n, t)-star complement constitutes a (n, t)-star in the original graph G. So, the algorithm operates over the complement of your graph. And it computes few sets and based on that it tries to find out whether a (n, t)-star complement is present in the graph or not.

So, the first step is it finds a maximum matching in the graph using any polynomial time algorithm. It finds a maximum matching in the $\bar{G}$ graph. Now, based on this matching M, there will be a subset of matched nodes and a subset of unmatched nodes. So, let N denote a subset of matched nodes and N complements denotes the unmatched nodes. Now, among the unmatched nodes we further identify triangle heads.

Namely, a triangle head is an unmatched node v_i which has edges with the end points of an edge present in your matching, maximum matching; that means, imagine (v_j, v_k) is a part of the maximum matching; that means, both are matched nodes. And, v_i is an unmatched node, but it has an edge with v_j and it has an edge with v_k. So, that is why we are calling it as a triangle head. It is a part of a triangle; it is a head of a triangle.

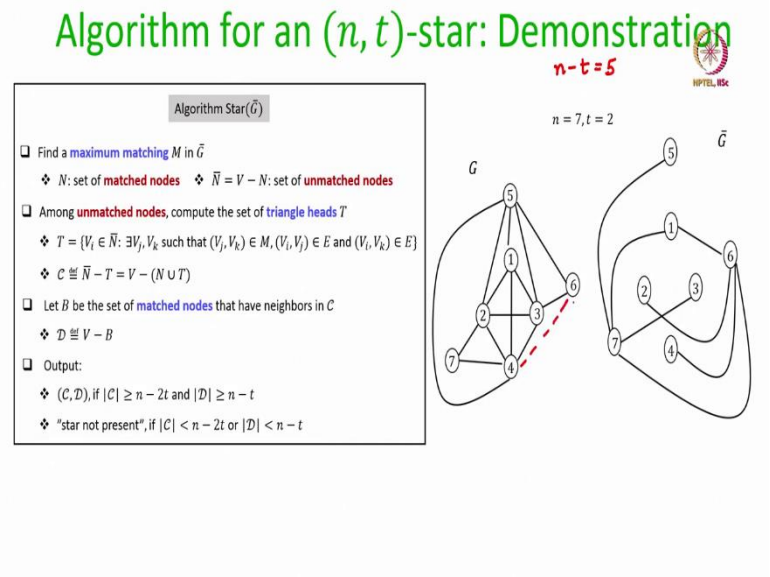
So, the collection of such triangle heads we are denoting by T. Now, we set our $\mathcal{C}$ set to be the set of all nodes except the matched nodes and the triangle heads; that means, you ignore

the matched nodes and the triangle heads, remaining all other nodes you include it in your set $\mathcal{C}$.

And, now let B be the set of nodes, B be the set of matched nodes to be more specific, it B be the set of matched nodes that have neighbors in $\mathcal{C}$. Now, once you have computed your set B, you set your set $\mathcal{D}$ to be the set $V \setminus B$; that means, you exclude all the nodes from the set B. So, now, you got your set $\mathcal{C}$, you got your set $\mathcal{D}$; you check whether the $\mathcal{C}$ is of size at least $n - 2t$ and $\mathcal{D}$ is of size at least t.

If both these conditions are satisfied, then you output your $\mathcal{C}$ and $\mathcal{D}$, otherwise you output the message that star is not present. And, then it can be proved, i if your graph G has a clique of size at least $n - t$, then this algorithm star on the $\bar{G}$ graph will output a (n, t)-star complement. And this (n, t)-star complement will be an (n, t)-star in the graph G, we can prove this. I will not go into the proof, I will share the reference where you can find the proof. The proof is very easy.

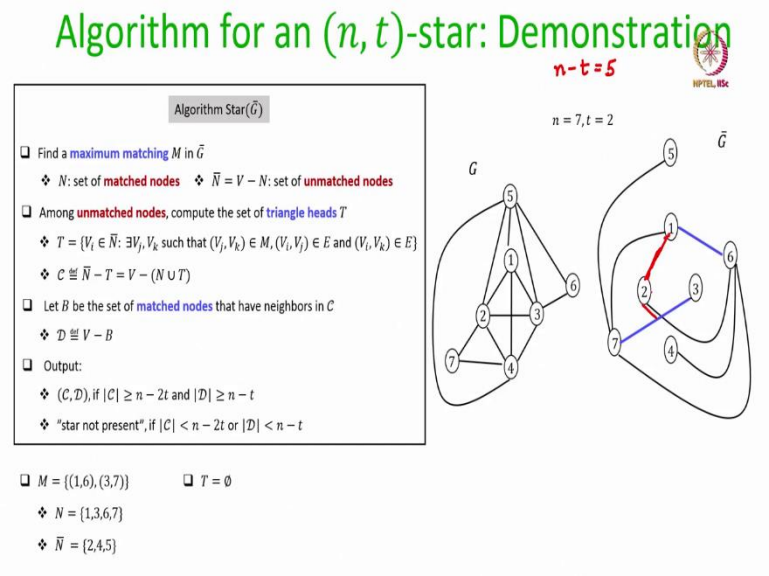
(Refer Slide Time: 22:05)



Now, let us see a demonstration for this algorithm. This is the graph G, which I am considering. We have 7 vertices and we have 2 nodes, we have $t = 2$. So, now, $n - t$ is 5. Does this graph have a clique of size 5? And the answer is yes. If I focus on this collection of nodes 1, 2, 3, 4, 5 then it is a complete graph of 5 nodes. You have an edge between 1 and 2, 1 and 3, 1 and 4, 1 and 5, 2 and 3, 2 and 4, 2 and 5, 3 and 4, 3 and 5 and between 4 and 5.

So, now let us see how this algorithm star $\bar{G}$ will work on this graph. So, the $\bar{G}$ graph will be the graph on your right-hand side. So, you can see there is no edge involving the nodes 1, 2, 3, 4, 5; because all those edges were present in graph G. So, that is why they will be now absent in $\bar{G}$ and there was no edge between 4 and 6 in the graph G, that is why there is an edge between 4 and 6 and so on.

(Refer Slide Time: 23:28)



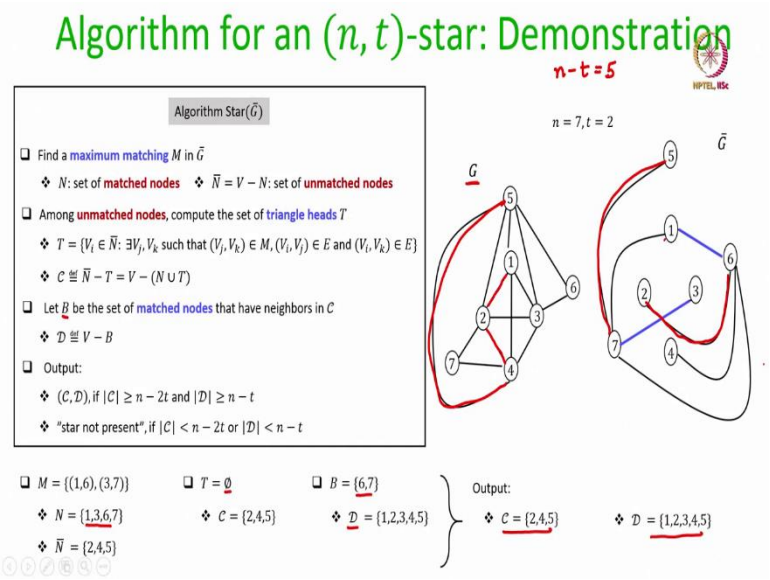
Now, what is the first step? The first step is to find the maximum matching. So, there are many maximum matchings in this graph. Suppose, the algorithms outputs this maximum matching the edge (1, 6) and (3, 7). And, it is easy to see that this is the maximum matching. To this collection M, you cannot add any more edge; so, that the expanded set still constitutes some matching.

Now, with respect to this maximum matching what are the matched nodes? The end points are 1, 3, 6, 7 and the unmatched nodes are 2, 4, 5. Now, let us see whether we have any triangle head. So, triangle head means all the unmatched nodes which have an edge with the end points of a matching in the maximum matching; that means, which have neighbors in the collection N. In this case, we do not have any triangle head.

So, for instance 2 is not a triangle head, because 2 does not have an edge with 3, 2 does not have an edge with 7, 2 does have an edge with 2 does have an edge with 6, but 2 does not have an edge with 1. So, that is why 2 does not constitute a triangle head.

If you would have if there would have been an edge between 2 and 1 and then 2 and 6 along with the edge, blue colored edge 1, 6 that would have labeled 2 as a triangle node, triangle head, but that is not the case here. So, 2 is not a triangle head, in the same way 4 is not a triangle head, 5 is not a triangle head. So, the subset of triangle head is empty.

(Refer Slide Time: 25:17)



So, now $\mathcal{C}$ will be $\bar{N} \setminus T$. Namely, you ignore all the vertices from N and triangle head. So, there is no vertices to be ignored from the triangle head, but you have to ignore all the matched nodes. So, you are left with 2, 4, 5. Now, let us see whether we have any element present in the set B.

So, the set B will have all the matched nodes which have neighbors with the nodes in $\mathcal{C}$. So, what are the matched nodes? 1, 3, 6, 7. So, 1 does not have a neighbor with 2, 4, 5, 3 does not have a neighbor with 2, 4, 5, but 6 have a neighbor in 2, 4, 5. Namely, 6 has an edge with the node 2; so, 6 is a part of the set B and 7 also have a neighbor in 2, 4 5.

So, this edge is present. So, that is why 7 is a part of subset B. Now based on this, the value of the subset $\mathcal{D}$ will be 1, 2, 3, 4, 5. And, output of the algorithm will be a star, where the $\mathcal{C}$ component of the star is 2, 4, 5 and the $\mathcal{D}$ component is 1, 2, 3, 4, 5.

So, now you can check that in the graph G, indeed the collection 2 to 4 and 4 to 5, it is a triangle. So, it is a complete graph of 3 nodes and 2, 4, 5 is a part of the bigger set 1, 2, 3,

4, 5. So, 1, 2, 3, 4, 5 constitutes the $\mathcal{D}$ set and there is an edge between every node in $\mathcal{C}$ and every node in $\mathcal{D}$. So, that is the way the star algorithm will operate.

(Refer Slide Time: 27:20)

So, as I studied I have as I have discussed I have not given you the full proof that indeed the output of the algorithm will be an (n, t)-star; if the graph is guaranteed to have a clique of size at least $n - t$. The exact proof is available in this PhD thesis.

Thank you.