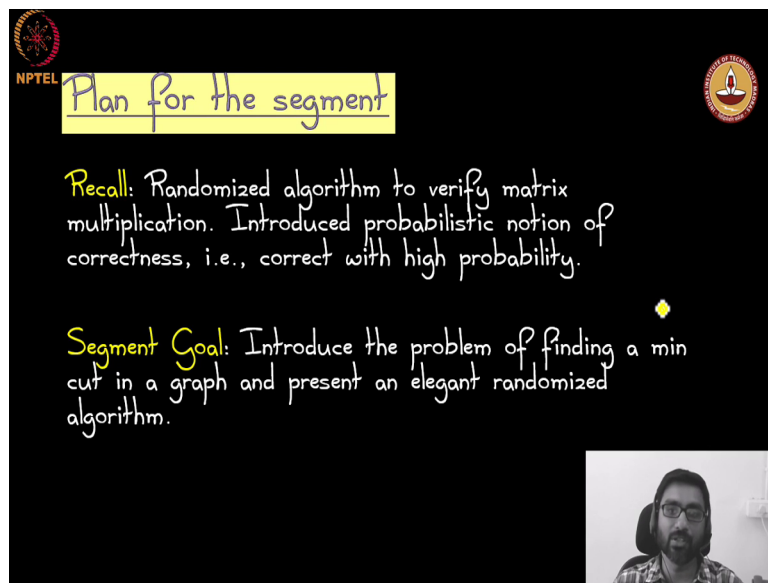


Probability & Computing
Prof. John Augustine
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Module – 01
Introduction to Probability
Lecture - 05
Segment 5: How Strong is your Network?

(Refer Slide Time: 00:19)



Plan for the segment

Recall: Randomized algorithm to verify matrix multiplication. Introduced probabilistic notion of correctness, i.e., correct with high probability.

Segment Goal: Introduce the problem of finding a min cut in a graph and present an elegant randomized algorithm.

We are going to be talking about how strong our network is and what do we; well let us look at the plan for this segment with the recall that in the last segment, we studied a randomized algorithm for matrix multiplication and we introduced the probabilistic notion of correctness.

What do we mean by an algorithm being correct with high probability in and in this current segment, we are going to look at the problem of how strong a network is formalize it in the following way we are going to define something called a min cut in a graph informally speaking it is the fewest number of edges that you need to remove in order to disconnect that graph that if that is very small, then the graphs in some sense is not strong enough and that is the intuition that leads to this formalism and we present a very simple elegant algorithm a randomized algorithm to solve this problem.

(Refer Slide Time: 01:21)

The slide features a black background with the NPTEL logo in the top left and the IIT Bombay logo in the top right. The title "The Mincut problem" is written in white. Below the title, the question "Which of these graphs will be a robust network?" is posed. Two graphs are shown: one on the left with a single yellow edge highlighted, and one on the right with a yellow arc highlighting a set of edges. At the bottom center, the text "Karger's Mincut Algorithm" is visible. A small video inset in the bottom right corner shows a man with glasses speaking.

So, let us look at the motivation as I said, we are trying to figure out; how strong the network is. So, here are 2 examples of networks. The question is which of these graphs networks is robust ok. If you look at the one on the right, it is completely connected. So, let us let us say, we are play the devil's advocate, we want to disconnect this network, you would have to destroy a lot of edges before this network becomes disconnected, even if you want to isolate one node, you will have to destroy all the incident edges and that is a lot on the contrary, if you look at this graph on the left, if this is a network, you only have to delete that one link and you have disconnected this network.

And so, this makes this network very brittle and so, you want to find out how brittle or how easy to break your network is and so, we formalize it in the following ways, we call this the min cut problem ok.

(Refer Slide Time: 02:39)

The Mincut problem

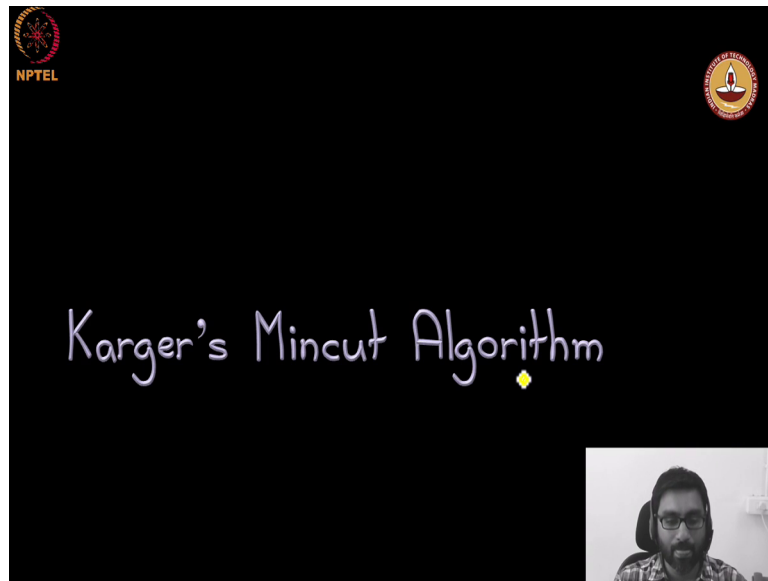
- Consider an undirected, unweighted graph $G = (V, E)$.
- A cut is a partition of V into two sets S and $V \setminus S$.
- The cutset corresponding to this cut is the set of edges with one end in S and the other in $V \setminus S$.
- Our goal is to find the partition such that the corresponding cutset has least cardinality.

Karger's Mincut Algorithm

So, we assume that our input is a graph with vertex set V and vertex E this represents our network. So, typically you can think of the vertices as being the computer nodes and edges is being the ability for a bit that 2 nodes can talk to each other. And, so if you can create a cut in this network; that means, there is a portion of the network that cannot talk to the rest of the network and that is that is not a good situation. We want to avoid that. So, cut is a partition of vertic of the vertex at V into 2 sets S and the remaining V minus S and the cut set corresponding to a particular cut is the set of edges with one end in S and the other end in V minus S .

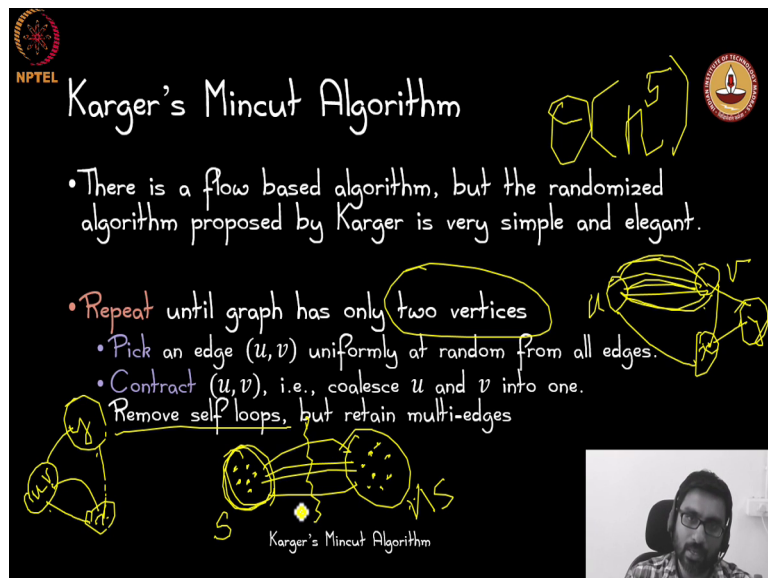
So, all these edges that go across the cut are called the cut set our goal is to find the partition we want to find the set S such that the corresponding cut set has the least cardinality you want to find the smallest cut and that is called the min cut. So, we want to find the min cut.

(Refer Slide Time: 04:13)



So, the algorithm we are going to look at was invented by Karger's. So, call it the Karger's Mincut algorithm.

(Refer Slide Time: 04:21)



Um there is you know we have we have to keep in mind that there is a flow based algorithm that could do quite well as well, but it is still not very efficient the most straightforward application of the flow based algorithm would lead to something of the order of n^5 and we want to do better than that ok. So, that is and moreover the Karger's algorithm is very very elegant and nice. So, you will see this and the algorithm

here is the algorithm right its very simple you just repeat this process of contraction n minus 2 times because each time you contract you will be coalescing two vertices. So, you can do that n minus 2 times for your left which is two vertices.

So, repeat until the graph has only two vertices. Now, let us pick an edge uniformly at random from all the edges, I think note here is the graph the way it will progress you might end up having multiple edges going between vertices. So, when you pick an edge you are not you should be careful not to pick two vertices and then consider the edge connecting them because some edges could have a lot of some vertices in some pairs of vertices could have a lot of edges going between them and others may be much fewer.

So, you really have to be careful to pick the edge uniformly at random over the set of all edges ok. So, these two vertices between which there is a lot of edges and another. So, for example, here say this one has lots of edges going between them and there is another pair of vertices and you only have one edge going between them, the probability this one edge is chosen should be exactly equal to the probability that this other edge of here is chosen ok.

So, pick such an edge uniformly at random and then you look at the two vertices that it is connecting. So, let us say this is u and this is v . Now knowing that this edge is connecting u and v , we have to contract or coalesce u and v into a single vertex. So, you have to contract that edge and when you do that. So, for example, now if you contract u and v and that is add a couple of more edges to illustrate the point, let us say this is our graph when you contract u and v , you have to put u and v together as one vertex.

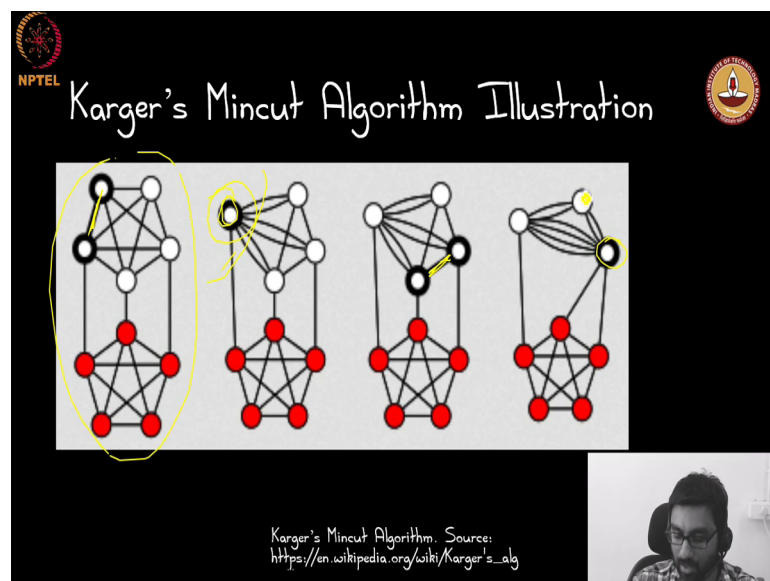
So, this is the coalesced vertex u v , and let us just call these x and y ok. So, all these edges that go between u and v will form self loops when you contract them. So, they are all removed. So, you remove all the self loops just do not consider them, but you need to be careful about the other edges though. So, let us put down x and y and you have let us look at the there is one edge going between u and x now there is no u there is only a u v , but you see that edge still has to be put in over here.

And between v and x there is an edge. So, that has to be a separate edge between V and y there is an edge and between x and y they there previously was an edge. So, this is the new graph that we get ok. So, this is the algorithm you just keep contracting you pick an edge uniformly at random you contract it and create a new graph and this new graph will

have one fewer vertex keep in mind some of the vertices are going to become these sort of compound vertices that contain multiple original vertices in them. So, we do this until we are left with just two vertices and when we are left with two vertices what will picture look like well it is going to look something like this there is going to be 2 sort of giant vertices that include a lot of coalesced original vertices and then. So and then there is going to be edges across them.

And all the vertices in one of these what is the original vertices in one of these what is these sort of coalesced vertices you call them s . So, all the original vertices in the other coalesced vertex is going to be V minus S at least this is going to be our candidate S and V minus S and our hope is that this edge this edge set will be the min cut and this of course, is not guaranteed, but we hope to be able to show that with some reasonable probability indeed that will be the min cut.

(Refer Slide Time: 09:57)



So, let us look at the execution of this algorithm in this example. So, here you have a sequence of graphs ok, what this is the original graph this is the original graph and we have chosen to contract this vertex as.

So, this edge and because of that contraction it is the two vertices have become one vertex and now there are one two three four five six edges going away from these pair of vertices. So, those 6 edges must find there well actually there is a seventh one over here. So, then those seven edges must be here and you notice that is the case there are these

seven edges that are incident on this new coalesced vertex all this must have been oneself loop that was created that was removed. And now you pick one more edge to contract this is the edge that is chosen for contraction and after contraction you get this coalesced vertex and you continue on ok.

(Refer Slide Time: 11:15)

Karger's Mincut Algorithm Illustration

Karger's Mincut Algorithm. Source: https://en.wikipedia.org/wiki/Karger's_algorithm

So, now you have chosen to contract this edge and you get this vertex as the coalesced vertex. Now notice that you are trying to contract this edge and you get this coalesced vertex and you continue on.

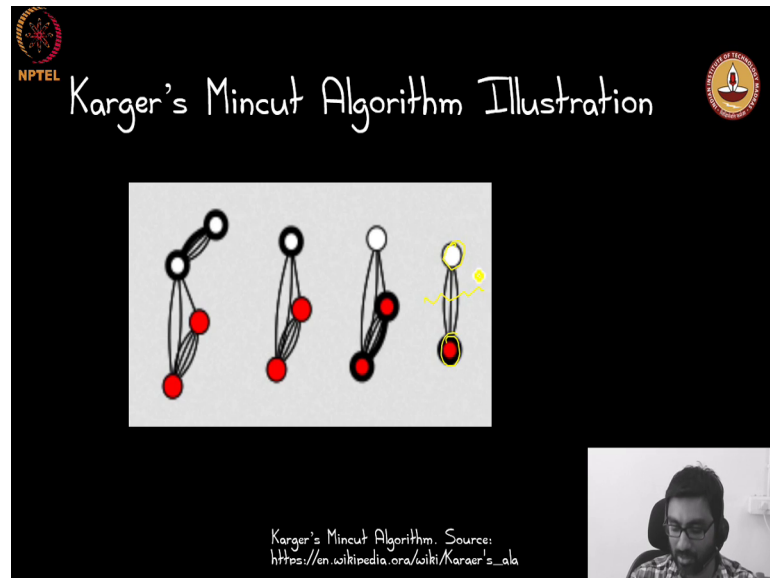
(Refer Slide Time: 11:37)

Karger's Mincut Algorithm Illustration

Karger's Mincut Algorithm. Source: https://en.wikipedia.org/wiki/Karger's_algorithm

So, you are contracting you have chosen to contract this edge you get this coalesced vertex and you continue proceeding.

(Refer Slide Time: 11:46)



This way you are left with two final coalesced vertices and you are left with these 3 edges and incidentally these are all this coalesced vertex in this coalesced vertex respectively contain all these vertices and all these vertices respectively and if you think about it the min cut indeed was this these 3 edges that separated the top from the top 5 vertices from the bottom 5 vertices and. So, this shows an example of a good run of this algorithm because it has actually led to us finding the min cut.

(Refer Slide Time: 12:33)

Analysis

Lemma: Let S be the set of original vertices that lead to one of the final two vertices. With probability at least $\frac{1}{\binom{n}{2}}$, S and $V \setminus S$ induce the smallest cutset.

Proof:

- Let C^* be a cutset of minimum cardinality.
- In order to prove the lemma, we will simply prove that:

$\Pr[\text{cut produced by algorithm is } C^*] \geq \frac{1}{\binom{n}{2}}$

Karger's Min-Cut Algorithm

But of course, that is not at all guaranteed it is a probabilistic thing and you need to be able to show that this happens with some reasonable probability. So, let us try to analyze that ok. So, what is it that we are going to prove let S be the set of original vertices that led to one of the final two vertices.

So, remember the algorithm ends with 2 final vertices with and let S be the set of original vertices that led to one of the final two vertices now with probability at least one over n choose two. So, that is one over some polynomial n S and V minus S induced the smallest cut. So, the cut set produced in this way is going to be the min cut with probability at least one over and choose 2. So, this is this is what we need to show ok. So, I noticed that this probability is fairly small. So, this one over some polynomial n , but bear with bear with me will figure out how to deal with that.

So, how do we prove this lemma ok, actually we do not care a about all possible min cuts you know a graph can have lots of min cuts and good exercise for you would be to come up with graphs that have lots and lots and lots of min cuts. So, now, let us pick one min cut that C^* be a cut set a min cut of minimum cardinality ok. So, instead of proving something strong like the probability of finding any min cut is bla, we are just going to limit show that even finding this one particular min cut this finding the probability of finding this one particular min cut is at least one over n choose 2 and this is actually

good enough if you think about it ok. So, we are going to limit ourselves to this proving this.

(Refer Slide Time: 15:05)

NPTEL Analysis (contd.)

Let e_1, e_2, \dots, e_{n-2} be the sequence of edges contracted by the algorithm. The algorithm succeeds if none of them are in C^* .

$$\begin{aligned} \Pr[\text{cut produced by algorithm is } C^*] &= \Pr(e_{n-2} \notin C^* \cap e_{n-3} \notin C^* \cap \dots \cap e_1 \notin C^*) \\ &= \Pr(e_{n-2} \notin C^* | e_{n-3} \notin C^* \cap \dots \cap e_1 \notin C^*) \times \\ &\quad \Pr(e_{n-3} \notin C^* \cap \dots \cap e_1 \notin C^*) \end{aligned}$$

Karger's Mincut Algorithm

So, let us try to figure out, how to do that and think about how this event may fail to happen what could go wrong as long as the edges that you keep picking are avoiding the min cut, this particular C^* um, let us let us say that you know there is a graph looks something like this and this is the min cut that are this is the C^* that we are talking about as long as the contractions happen over here, in other edges that are not part of the min cut, we are going to be fine the min cut is going to survive till the end the moment we pick a min cut edge we have lost it ok. So, that is the intuition that we want to formalize ok.

So, towards that let e_1, e_2 and so on up to e_{n-2} be the sequence of edges contracted by the algorithm now as I said if one of this e_1 through e_{n-2} is a member of this min cut that is bad ok. So, the other way of saying is that the algorithm succeeds is if none of them are in C^* ok. So, this probability that we care about probability that the cut produced by the algorithm is not C^* is C^* is equal to the probability that the last edge that we contracted was not in C^* the last, but one of the edge that we contracted was not in C^* and so, on up to the first edge that was contracted ok. So, I have just gone in the back reverse order for a reason.

So, now this is the probability of the intersection of several events also. So, all of these things will have to happen in order for the algorithm to succeed. So, a lot of stars have to line up. So, let us what we you know let us see the one thing need, we need to be careful about is these are not independent events, they could there could be dependencies. So, we should be a little bit careful we cannot just multiply their individual probabilities, but given this situation, we can write this intersection of all these probabilities in the following manner you isolate the probability the conditional probability of just the very first event listed over here conditioned on all these other things happening ok. But if you were to do that then you also have to multiply it by the probability of whatever condition you imposed ok.

(Refer Slide Time: 18:27)

NPTEL Analysis (contd.)

$$= \Pr(e_{n-2} \notin C^* | e_{n-3} \notin C^* \cap \dots \cap e_1 \notin C^*) \times \Pr(e_{n-3} \notin C^* \cap \dots \cap e_1 \notin C^*)$$

$$= \Pr(e_{n-2} \notin C^* | e_{n-3} \notin C^* \cap \dots \cap e_1 \notin C^*) \times \Pr(e_{n-3} \notin C^* | e_{n-4} \notin C^* \cap \dots \cap e_1 \notin C^*) \times \dots$$

$$\Pr(e_i \notin C^* | e_{i-1} \notin C^* \cap \dots \cap e_1 \notin C^*) \times \dots$$

$$\Pr(e_2 \notin C^* | e_1 \notin C^*) \times \Pr(e_1 \notin C^*)$$

Karger's Mincut Algorithm

So, that same thing is written over here, but now the same structure holds over here. So, here if you notice it is just the probability of the intersection of several events except $e_{n-2} \notin C^*$ has been removed, but the rest of them over all intersection of intersections of events which means we can repeat the same procedure you isolate $e_{n-3} \notin C^*$ conditioned on the rest of them. But then, if you were to do that you would have to then multiply it by the probability of this condition which means then you can isolate $e_{n-4} \notin C^*$ conditioned on the remaining intersections and so on.

So, when you work that out you are going to get this this sequence of conditional probabilities and the last one of course, is you are going to end with probability of e_1 not in C^* after which you cannot apply this any further. So, this boils down to computing one such conditional probability because really the probability that we want is the product of a sequence of such conditional probabilities. So, by now you may be wondering; what is the intuition ok. So, let us actually look at a picture that gives us the intuition of exactly what happens let us think of the execution of this algorithm.

at the very beginning you are in some state you have still not picked any edge. So, then you pick an edge that edge could be a member of the min cut C^* or not if it is a member of the cut C^* basically what; that means, the very first edge that you picked if that belongs to C^* that is very bad your this execution will not succeed otherwise you something good happens e_1 is not in C^* good great. So, then the execution has survived the first iteration ok, but then again the second edge that was chosen e_2 if it belong to C^* again we are out of luck. So, the execution we had as we had off into a bad situation otherwise its good and this continues on ok.

So, at any point in time, let us look at what is the you know what happens well if you have managed to reach this point that is basically saying let us let us say that there was a sequence of executions this is the i -th iteration how can you say that the i -th iteration will succeed or fail well first of all you should have reached the i -th iteration what does it mean to have reached the i -th iteration all of these where good all of these had to have been good only then you would have even reached here and that is exactly what is captured by this condition all of the previous edges e_{i-1} e_{i-2} and so on up to e_1 , all of those edges must have missed the min cut only then you would have even come to this point and then conditioning having come to this point.

Now, you can ask what is the probability that you would either go down this bad path or you would continue to I am sorry you would continue to be in the good path and that is exactly the intuition here. So, now, that we can come back to the formalism here let us look at this let us look at each of these conditional probabilities right. So, in particular we are going to focus on this one conditional probability right. So, let us 2×2 get the right expression for that when particular probability.



(Refer Slide Time: 23:20)

The slide features a black background with white text and a diagram. At the top left is the NPTEL logo, and at the top right is the IIT Bombay logo. The title 'Minimum Degree and Cutset Size' is written in white. Below the title, there are two bullet points: 'Let $k = |C^*|$. Clearly, the minimum degree of G must be at least k .' and 'Moreover, this holds as an invariant for all intermediate (multi) graphs.' The diagram shows a green circle labeled 'Coalesced node' containing several red dots, connected by orange arrows to a purple cloud labeled 'Rest of the network'. At the bottom center, it says 'Karger's Mincut Algorithm' with a small logo. A small video inset of a person is visible in the bottom right corner.


Let us denote the size the min cut asking, now if you think about it as the graph control I mean as the edge contractions take place the degree of the graph that we obtained is never going to be less than k why is that well if you look at any cut in the graph as the algorithm progresses in the let us see in the i -th iteration any cut in the graph actually corresponds to an original cut in the original graph. So, if that if the number of edges going across the cut that you chose in the i -th iteration is less than k its also going to be less than k in the original graph.

So, k is going to be a lower bound on the min cut as the algorithm progresses. So, all the graphs that we encounter till the very end is going to have a min cut of size at least k which means the degree will also have to be at least k because if the degree is smaller, then you can create a cut you can either do the small the low degree vertex can become one side of the cut and all the other vertices can become a other side of the cut. So, clearly that the minimum degree of the graph G that we have is at least k and as I mentioned this is going to be the case throughout the execution right. So, any look at any coalesced vertex u . So, this has a lot of the original nodes and if you look at all the edges going out of that node it cannot be it cannot become less than k ok.

(Refer Slide Time: 25:23)


$$\Pr(e_i \notin C^* \mid e_{i-1} \notin C^* \cap \dots \cap e_1 \notin C^*)$$

- Just before the i^{th} contraction, there are $n - i + 1$ vertices.
- Recall also that the minimum degree is always at least k .
- The number of edges, therefore, is at least $\frac{(n-i+1)k}{2}$.




So, just before the i -th contraction how many vertices are there well I minus one contractions have taken place. So, you are left with n minus I , I you will be left with n minus I minus one vertices which is n minus I plus 1 and each one of them has degree at least k right. So, if let us if you count and how do you count the number of edges well actually let us count the total degree there are n minus I plus one vertices each of degree at least k and, but if you count the total degree that double counts the number of edges. So, if you divide by 2 you will get a lower bound on the number of edges. So, now, let us go back to this probability remember this probability term is what we want to compute ok.

We know that the total number of we know that the number of edges is at least add this quantity and we have reached the i -th the execution without destroying the min cut C^* how do we ensure.

(Refer Slide Time: 26:45)

NPTEL Thus,

$$\begin{aligned} & \Pr[\text{cut produced by algorithm is } C^*] \\ &= \Pr(e_{n-2} \notin C^* \mid e_{n-3} \notin C^* \cap \dots \cap e_1 \notin C^*) \times \\ & \quad \Pr(e_{n-3} \notin C^* \mid e_{n-4} \notin C^* \cap \dots \cap e_1 \notin C^*) \times \\ & \quad \vdots \\ & \quad \Pr(e_i \notin C^* \mid e_{i-1} \notin C^* \cap \dots \cap e_1 \notin C^*) \times \\ & \quad \vdots \\ & \quad \Pr(e_2 \notin C^* \mid e_1 \notin C^*) \times \Pr(e_1 \notin C^*) \\ &= \frac{n-2}{n} \times \frac{n-3}{n-1} \times \frac{n-4}{n-2} \times \dots \times \frac{2}{6} \times \frac{3}{5} \times \frac{1}{3} = \frac{2}{n(n-1)} \end{aligned}$$



So, what is the probability?

(Refer Slide Time: 26:46).

NPTEL

$$\Pr(e_i \notin C^* \mid e_{i-1} \notin C^* \cap \dots \cap e_1 \notin C^*)$$

- Recall: $e_{i-1} \notin C^* \cap \dots \cap e_1 \notin C^*$, so all k edges in C^* are intact.
- Therefore,

$$\begin{aligned} & \Pr(e_i \notin C^* \mid e_{i-1} \notin C^* \cap \dots \cap e_1 \notin C^*) \\ & \geq 1 - \frac{\binom{k}{(n-i+1)k}}{2} \\ & = \frac{n-i-1}{n-i+1} \end{aligned}$$


That i -th edge that is chosen also avoids the min cut well that is going to be at least the probability that i -th edge is part of the min cut. So, the back this is the probability of a good event this is a good event that i -th edge did not belong to the min cut that is going to be 1 minus the probability of the bad event which is that i -th edge is a member of the min cut C^* . So, this is going to be the probability that i -th edge is a member of the min cut that is what we need to put here.

So, how do we get this expression here well there are k edges in the min cut if you choose one of them which you will do with probability k over the set of the total number of edges ok , if you this is going to be the probability with which you are going to choose one of these min cut edges. So, this is going to be the probability of the bad event. So, if you do 1 minus the probability of the bad event you will get the probability of the good event right and keep in mind that this is just the lower bound on the on the size of the on the number of edges.

So, when you replace the total number of edges by a lower bound this whole quantity can become a little bit larger the probability of the bear even could become large potentially a little bit larger. So, 1 minus of this therefore, you could only be a little bit smaller and as a result you will have this inequality here its greater than are equal to this whole thing on the right hand side ok.

So, now, if you work that out its going to come out as n minus i minus 1 divided by n minus i plus 1 this is the expression we get for this one probability term. So, now, going back to this original probability that the cut produced by the algorithm is C star remember we wrote that as the product of multiple conditional probabilities and for each one we have n minus i minus 1 divided by n minus i plus 1 .

So, if we plug that n we get this telescopic product and nice thing is there is going to be a lot of cancellations this n minus 2 will cancel out with this one there will be an n minus 3 cancel out cancelling out with the next n minus 3 over here and so on. So, here if you see the 3 will cancel out over here four will cancel out over here and so on. So, what you will be left with is these 2 terms the denominator and this 2 here in the numerator which is exactly what we want. So, with that we have proven the lemma ok.

(Refer Slide Time: 30:01)

Boosting the Probability of Success

- To improve our success probability, we can (independently) repeat the algorithm, say, $c \binom{n}{2} \log_e n$ times (for any $c > 0$) and report the smallest cutset.
- Probability that all these repetitions will fail is at most $(1 - \frac{1}{\binom{n}{2}})^{c \binom{n}{2} \log_e n} \leq e^{-\frac{c \binom{n}{2} \log_e n}{\binom{n}{2}}} = \frac{1}{n^c}$.
 (Since $1 - x \leq e^{-x}$)

Thus, algorithm will succeed WHP.

Karger's Mincut Algorithm

So, now, we need to figure out a way to boost the probability of success remember now we only have a probability of success that is something like 2 over n into n minus one and that is pretty low ok. So, let us figure out what let us see what happens when we repeat this some number of times.

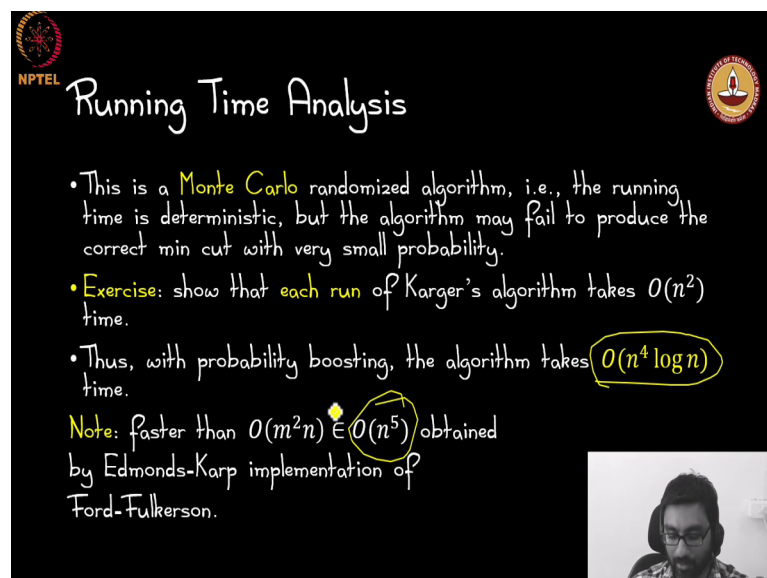
So, let us in particular we are going to repeat it this many times let us see what happens what is the probability that all of these repetitions these C times n choose 2 times log n number of repetition what is the probability that all of them will fail well the probability that one of them will fail is 1 is at most 1 minus 1 over n choose 2 the probability that all of them will fail and these are all independent repetitions is this a product of this many times. So, you can just raise it to the power C C n choose 2 times log n and we have this inequality 1 minus x is at most E to the minus x is a commonly used inequality. So, this 1 minus this quantity can be written as E to the minus 1 over n choose 2, but then you also have this in the exponent.

So, the new E to the you get overall you get E to the minus this exponent the nice thing is is this n choose 2 over here and then choose 2 over here, they cancel out and you can take the C as log in the whole raised to the power C and so, you finally, this works out to 1 over n choose n to the power C ok. So, the nice thing is now what we have shown is that the probability that all of these repetitions will fail is at most one over some arbitrarily large polynomial n ok. So, this leads us to the algorithm that will succeed with

high probability you keep repeating the algorithm this contraction algorithm that we have discussed you repeat that this many times C times n choose 2 times $\log n$ times and each time you repeat you see what is the min cut that you get if it is better than all the min cuts that you have seen before updated ok.

So, at the end you would have the best min cut that you have seen is in all these many repetitions and that min cut is going after that that candidate min cut actually will actually be the true min cut with probability at least $1 - \frac{1}{n^C}$ and that is that is exactly what we mean by with high probability.

(Refer Slide Time: 33:19)



The slide is titled "Running Time Analysis" and features the NPTEL logo in the top left and a circular logo in the top right. The content is handwritten in white and yellow on a black background. It includes a list of bullet points and a note. A small video inset in the bottom right corner shows a person wearing headphones.

Running Time Analysis

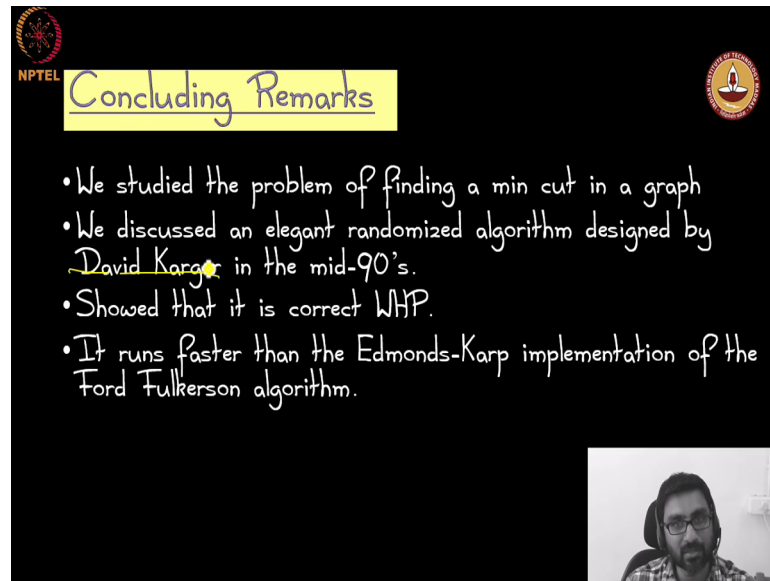
- This is a **Monte Carlo** randomized algorithm, i.e., the running time is deterministic, but the algorithm may fail to produce the correct min cut with very small probability.
- **Exercise:** show that each run of Karger's algorithm takes $O(n^2)$ time.
- Thus, with probability boosting, the algorithm takes $O(n^4 \log n)$ time.

Note: faster than $O(m^2 n) \in O(n^5)$ obtained by Edmonds-Karp implementation of Ford-Fulkerson.

So that is good. So, we have got the algorithm working correctly what we need to do is make sure that we have not designed a very bad algorithm in terms of running time um, but keep in mind that this is a Monte Carlo algorithm, we are not guaranteed to have found the min cut we have we guarantee that the min cut that we produce is going to be the cut as a cut that we claim as a min cut is indeed going to be the min cut with high probability, but not guarantee and this Karger's algorithm this contraction algorithm can be carefully designed to run in O of n squared time.

So, the overall running time because remember we at also repeat this is another n choose 2 times $\log n$ number of times. So, the overall running time is going to be O of n to the four $\log n$ and that is going to be better than the flow based algorithm.

(Refer Slide Time: 34:19)



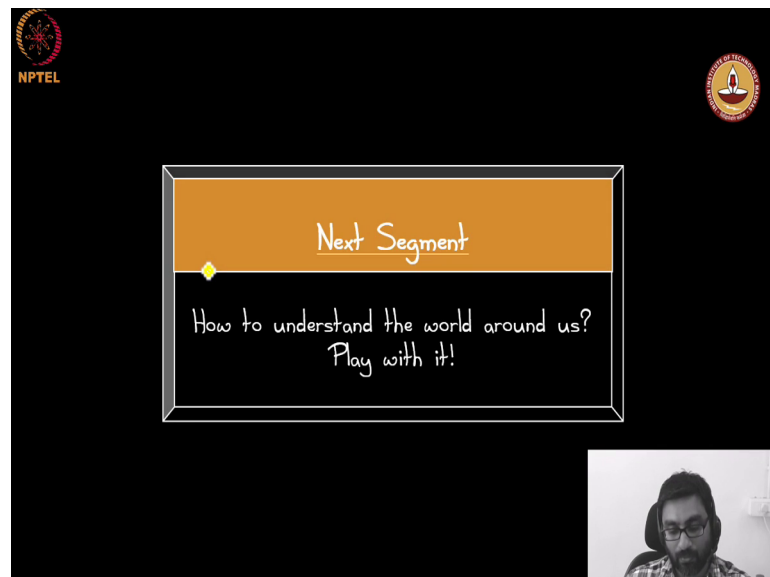
The slide features a black background with a yellow title box at the top center containing the text "Concluding Remarks". In the top left corner, there is a small circular logo with "NPTEL" written below it. In the top right corner, there is another circular logo with a lamp icon and the text "UNIVERSITY OF TECHNOLOGY" and "MUMBAI". The main content consists of four bullet points written in white text:

- We studied the problem of finding a min cut in a graph
- We discussed an elegant randomized algorithm designed by David Karger in the mid-90's.
- Showed that it is correct WHP.
- It runs faster than the Edmonds-Karp implementation of the Ford Fulkerson algorithm.

In the bottom right corner, there is a small rectangular video inset showing a man with glasses and a beard, wearing a headset, speaking.

So let us conclude this segment, we have studied the problem of finding a min cut in a graph we discussed an elegant randomized algorithm invented by David Karger and show that if you repeat it appropriately it is going to be correct with high probability which is faster than the usual deterministic algorithm that we think about in this context.

(Refer Slide Time: 34:50)



The slide features a black background with a white-bordered box in the center. Inside the box, the text "Next Segment" is written in white on an orange background. Below this, the text "How to understand the world around us?" and "Play with it!" is written in white. In the top left corner, there is a small circular logo with "NPTEL" written below it. In the top right corner, there is another circular logo with a lamp icon and the text "UNIVERSITY OF TECHNOLOGY" and "MUMBAI". In the bottom right corner, there is a small rectangular video inset showing the same man with glasses and a beard, wearing a headset, speaking.

In the next segment, we are going to study base law and we will see it via an example where we need to understand something about the world around us, we will repeat some experiments. And, as we run these experiments we will get a better and better

understanding of the world around us and that is what we mean by. I mean by this you know this prompt for the next segment how to understand the world around this play with it run some experiments figure out update your understanding of the world.

So, with that we end this segment.