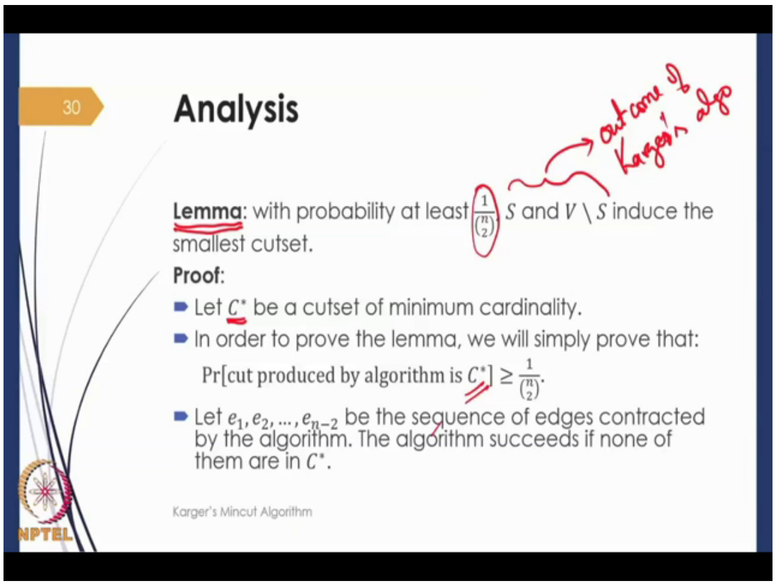
# Algorithms for Big Data

**Prof. John Ebenezer Augustine**
**Department of Computer Science and Engineering**
**Indian Institute of Technology, Madras**

## Lecture – 05
## Analysis of Karger's Mincut Algorithm

In this last segment, in today's lecture, we will analyze the Mincut Algorithm proposed by Karger.

(Refer Slide Time: 00:29)



### Analysis

30
out come of Karger's algo
Karger's Mincut Algorithm
NPTEL

**Lemma**: 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:

  $\Pr[\text{cut produced by algorithm is } C^*] \geq \frac{1}{\binom{n}{2}}$.
- Let $e_1, e_2, \ldots, e_{n-2}$ be the sequence of edges contracted by the algorithm. The algorithm succeeds if none of them are in $C^*$.

And just to recollect, this is the lemma that we are going to prove with probability at least 1 over n choose 2, this sets S and V minus S. These are the sets that come out as an outcome of Karger's algorithm. This two sets, this partition induces the smallest cut, this is the claim that we want to prove. And to prove that we will focus on one of the minimum cardinality cut sets. There could be many minimum cardinality cut sets. What we are going to do is pick one of them. So, C star I call it and we will prove something slightly stronger than it is required. We will prove that the probability of the cut produced by the algorithm, this S and V minus S is actually corresponding to the cut set C star. That probability is itself greater than 1 over n choose 2.

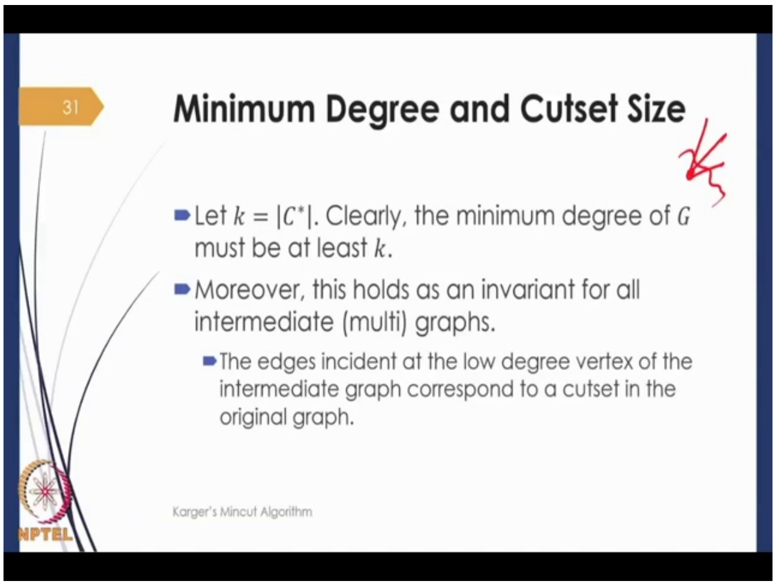
Now if there are other cut sets, naturally the probability with which the algorithm will fines a one such minimum cut set is only going to be increased, so therefore being conservative in our claim here. Now, let us proceed with proving this claim and towards that, let’s look at the execution of Karger’s mincut algorithm.

In each step one of the edges was picked and contracted. Let us call these edges e 1, e 2, and so on up to e (n minus 2). Of course, we only would contract n minus 2 edges, because of that time we may left with these two vertices and therefore we would stop.

What is the probability that the algorithm succeeds? Well, if all of these edges that were picked for contraction, missed C star then what we will be left with at the end is going to be C star. Another way of stating this is, as the algorithm progresses if we pick an edge and it happens to be an edge in this cut sets C star then the algorithm is going to fail. However, if all the edges that we pick as the algorithm progresses are outside of the set C star, the cut set C star then at after the n minus 2 edge contractions takes place what we will be left with is exactly the edge set C star.

Of course, if that were not the case meaning if e 1 through e (n minus 2) were not part of C star, but then of you were left with something other than C star then C star cannot be the minimum cut. This is going to be the criteria for success of our algorithm. So, none of the edges that we picked are in the element, set, in the cut set C star.

(Refer Slide Time: 04:38)



31

## Minimum Degree and Cutset Size

$k$

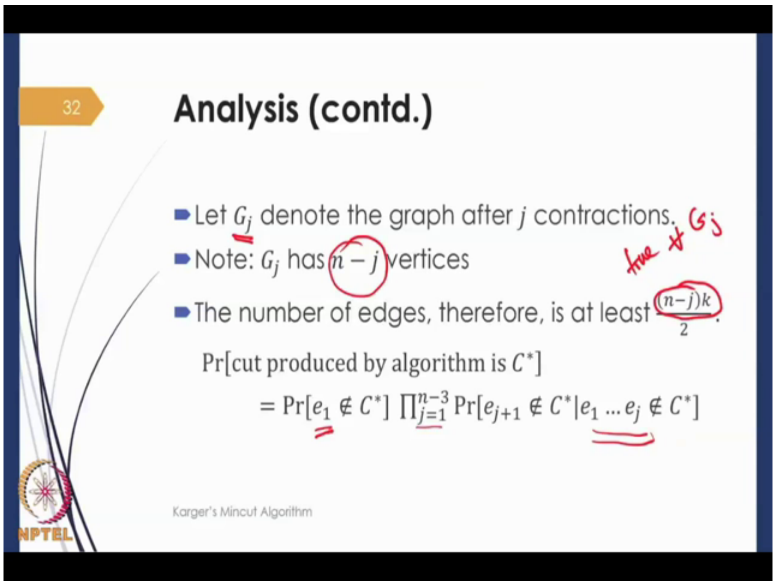
- Let $k = |C^*|$. Clearly, the minimum degree of $G$ must be at least $k$.
- Moreover, this holds as an invariant for all intermediate (multi) graphs.
  - The edges incident at the low degree vertex of the intermediate graph correspond to a cutset in the original graph.

Karger's Mincut Algorithm

Let us say the cardinality of C star is some k. Clearly, the minimum degree of G must be at least k. Why? Because if the minimum degree was something less than k then you have a smaller cut. So, for example, if k is 5 but if one of the edges has the degree just 3 well that is the contradiction because you have a cut just size 3. So, minimum degree of G must be at least k. And this continues to hold as an invariant for all intermediate multi graphs that are produced because at any point in time, if the degree goes below k, then we have a cut set that is corresponding to the original graph, that cut set that can be obtained in this manner.

The vertex with super note with degree less than k can be put on one side and all other vertices on the other side and the cut set that induces will be of cardinality less than k and that will contradict assumption that we have that k is equal to the cardinality of minimum cut set.

(Refer Slide Time: 06:13)



## Analysis (contd.)

- Let $G_j$ denote the graph after $j$ contractions. $G_j$
- Note: $G_j$ has $n-j$ vertices true $\forall$
- The number of edges, therefore, is at least $\frac{(n-j)k}{2}$.

Pr[cut produced by algorithm is $C^*$]

$$= \Pr[e_1 \notin C^*] \prod_{j=1}^{n-3} \Pr[e_{j+1} \notin C^* | e_1 \dots e_j \notin C^*]$$

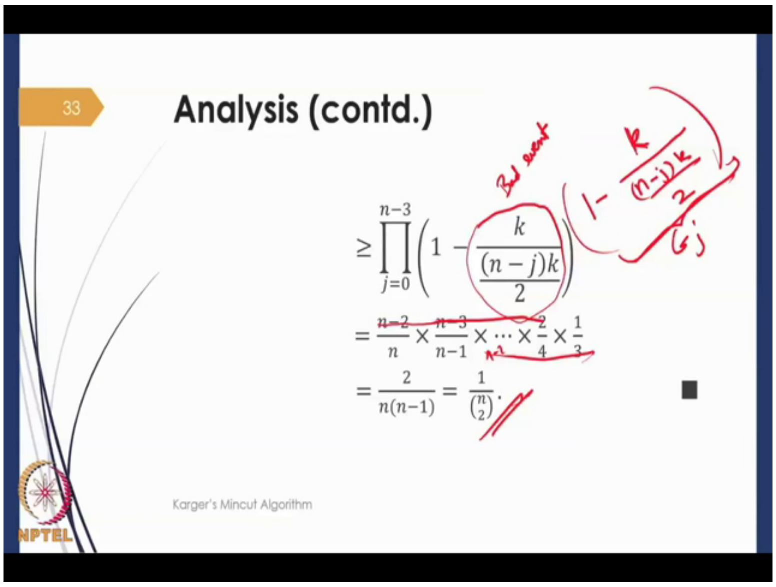
Karger's Mincut Algorithm

One more notation - let $G_j$ denote the graph after j constructions. And here you notice that $G_j$ has n minus j vertices, while G 0 will have all the n vertices, G 1 will have after one contraction will have just n minus 1 vertex and so on. The number of edges, you have to recall two things; $G_j$ has n minus j vertices and it is of degree at least k, minimum degree at least k. So, the total number of edges therefore must be at least n minus j times k divided by 2. Now the degree has to be shared, so each edge contributes two to the degree. If you look at the total degree, the total degrees n minus j times k, so the number of edges will have to be at least n minus j times k divided by 2. What is that mean? Now we can start, and this is true for all $G_j$.

So, with that now we have the mathematics needed to continue with the proof. And what we want to do is bound this probability that the cut produce by the algorithm is in fact to the specific cut C star. So, what is that? That is, this event is the probability that the first edge, in fact this is nothing but the probability that none of the edges that were chosen by the algorithm intersected with C star. Let us write the outcome carefully. That is the probability that e 1 does not belong to C star times, now you are going from j equal to 1 to n minus 3, so probability that e 2 does not belong to C star, probability that e 3 does not belong to C star and so on up to probability that e n minus 2 does not belong to C star.

But each one of the subsequent probabilities notice required that the previous edges also do not belong to C star, because if any of the previous elements edges belong to C star then the algorithm is already not going to lead us to C star.

(Refer Slide Time: 09:29)



33

## Analysis (contd.)

Bad event

$$\geq \prod_{j=0}^{n-3}\left(1-\frac{k}{\frac{(n-j)k}{2}}\right) \quad \left(1-\frac{k}{\frac{(n-j)k}{2}}\right) G_j$$

$$= \frac{n-2}{n} \times \frac{n-3}{n-1} \times \cdots \times \frac{2}{4} \times \frac{1}{3}$$

$$= \frac{2}{n(n-1)} = \frac{1}{\binom{n}{2}}.$$

■

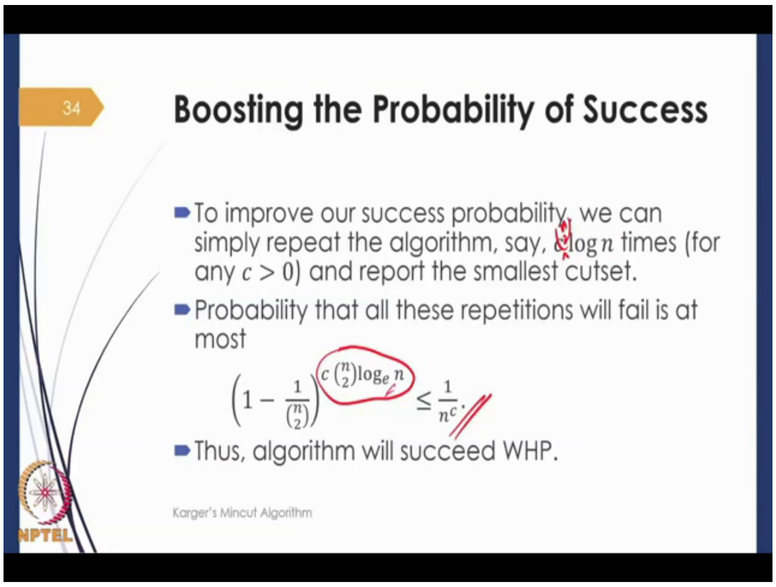
Karger's Mincut Algorithm

 So, just working through that product we get this. What we have here is the probabilities that each of the e j plus ones do not belong to C star. Now, what is the probability of actually picking an element from C star? Well, that probability is going to be at least k, because k divided by the total number of edges. K is the cardinality of C star. So that the bad event would be that we pick an element in C star, so, that's, this is the probability of the bad event which would be that we pick an element from an edge from C star, so there are k edges in C star. This is for particular $G_j$ the total number of edges we have already seen is this quantity. This is the probability of the bad event.

Now we are considering the good event and therefore one minus of that. So, what is the good event? Good event is that the edge that we picked is not actually an element of C star. And this is the good event for one particular $G_j$, and such a good event has to be occurring for every j and therefore we take a product overall j values and this is the expression we get.

Now, solving this expression each one of these is going to be of the form n minus 2 by n, n minus 3 by n times and so on up to 2 by 3 times 1 by 3 and. Of course, this going to be a lot of cancellations will be left with all these terms up to 2 will get canceled, and all these terms up to 3 will get canceled, there will be an n minus 2 over here starting from that you will have cancellations. So, we are left with 2 over n times n minus 1 which is nothing, but 1 over n choose 2 as we required. So, this completes the proof of our lemma.

(Refer Slide Time: 12:34)



## Boosting the Probability of Success

- To improve our success probability, we can simply repeat the algorithm, say, $c\log n$ times (for any $c > 0$) and report the smallest cutset.
- Probability that all these repetitions will fail is at most

$$\left(1 - \frac{1}{\binom{n}{2}}\right)^{c\binom{n}{2}\log_e n} \leq \frac{1}{n^c}.$$

- Thus, algorithm will succeed WHP.

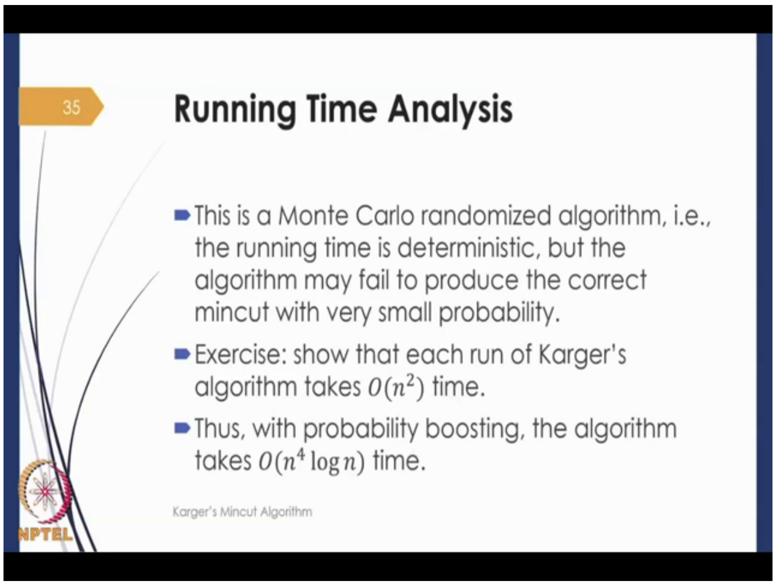
Karger's Mincut Algorithm

What we are left with is that we need to boost the probability of success. 1 over n choose 2 is too smaller probability of success, but it is not very difficult to boost this probability, we simply have to repeat the algorithm. Say some, C log n time n choose 2 number of times this quantity, and so, we have to repeat that many times. Now the probability that all of these reputations will failed is going to be,

$$\left(1 - \frac{1}{\binom{n}{2}}\right)^{c(nC2)(\log_e n)} \leq \frac{1}{n^c}$$

So, here I am using log base e. And this ends up being at most one over n to the C, which is what we need for high probability. This gives us a guarantee with very high probability that the algorithm will indeed find in particular of the cut set C star.

(Refer Slide Time: 13:50)



35

## 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 mincut 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.

Karger's Mincut Algorithm

We are finally left with Running Time Analysis. This particular algorithm is a Monte Carlo algorithm. So, the running time is actually deterministic quantity. Of course, the algorithm may fail to produce the correct mincut with the very small probability. A quick exercise here, you have to show that each run of the Karger's algorithm takes O of n square time. You have to be little bit careful about this, but are not hard at all. And if each execution takes order of n square time and we are going to repeat this, something like C n choose 2 time log n time. Therefore, the total running time ends up being O of n to the four times log n.

So, this is the running time. There are some techniques to improve the running time, but for our purposes this is the, it is still a good polynomial running time, and moreover the algorithm is very elegant and simple and easy to implement.

Thank you very much.