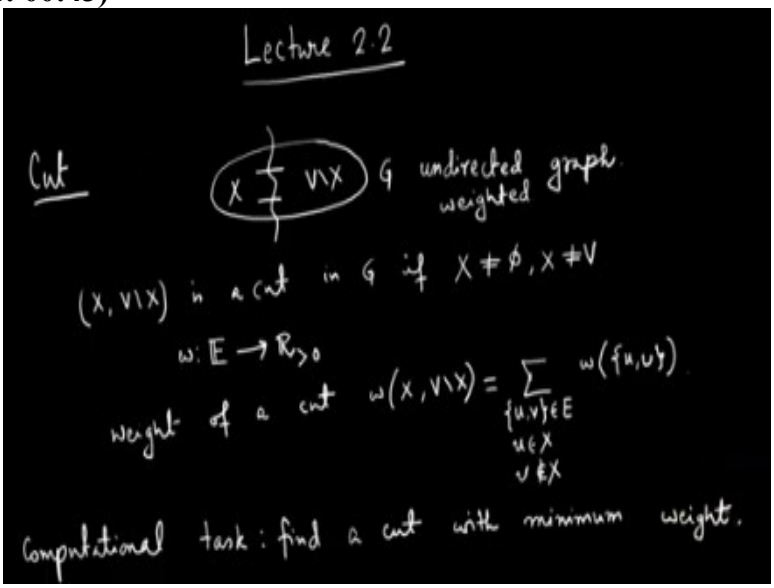
Selected Topics in Algorithm
Prof. Palash Dey
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Module No # 02
Lecture No # 07
Karger's Algorithm

Thank you welcome so in the last class we had seen some applications of maximum flow in finding matching's in Bipartite graphs so let us continue from after that.

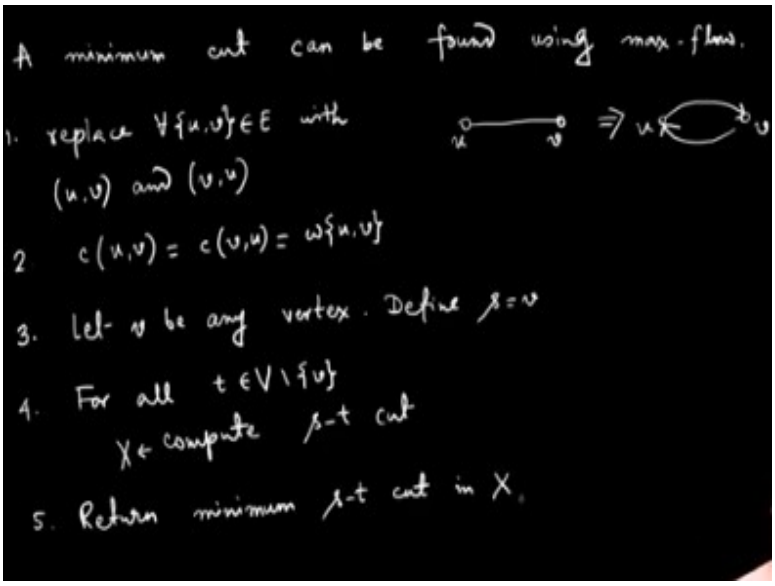
(Refer Slide Time: 00:43)



So in today's class we will see an algorithm for min cut problem. So what is min cut so what is a cut? First let us define what is a cut in a graph? So let G be a graph G an undirected graph and a cut is just a partition of the vertices into 2 non-empty sets so $(X, V \setminus X)$ is a cut in G. If x is not equal to empty set and also x is not equal to the entire. Set so what are undirected and weighted graph that means edges are weighted.

So there is a weight function W from each set to positive real numbers and we define the weight of a cut W of $(X, V \setminus X)$x is the sum of weights of all the edges which crosses this cuts which has 2 endpoints in 2 parts. So this is $(u,v) \in E, u \in X, v \notin X$ a weight of u, v. And the computational task computational question is to find a cut with minimum value, minimum weight. So this problem also can be solved using max flow as follows.

(Refer Slide Time: 04:26)

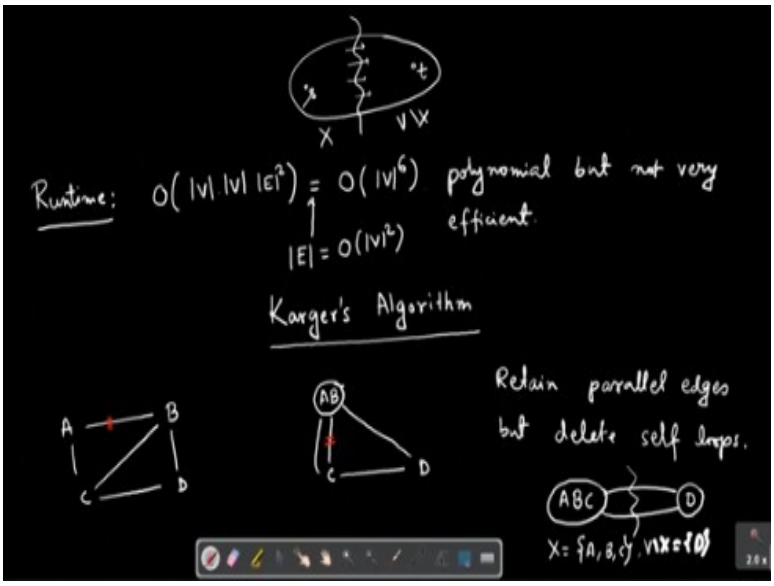


So minimum cut can be found using max flow as follows the first step is for the for max flow to be applicable the graph should be directed. So but in the cut problem we have the undirected graph so what we will do is that we will replace each undirected edge u, v with 2 directed edges. So replace all u, v in e with 2 directed edges u, v and v, u. So these are 2 anti-parallel edges we assume that anti parallel edges are not allowed

In max flow but that assumption if you recall is without loss of generality by introducing new vertices we can get rid of anti-parallel edges. Next is what we do is we make the capacity of both the directed edges to be equal to the weight of the undirected edge now and we make any arbitrary vertex to be source.

So let v b any vertex define s = v and iterate over all other vertices to be t and compute the min s, t cut. So for all $t \in V - \{v\}$ compute s t cut and let us let put all such cuts in a set X and then return minimum s, t cut in X.

(Refer Slide Time: 08:23)



Now the correctness is easy to see because so suppose here is this vertex s and let us look at the min cut suppose this is X this is $V \setminus X$. And let us pick any vertex from $V \setminus X$ to be t and this cut is the size of this cut is the sum of the capacities of this weight of this edges which is the sum of the capacity some of the weights of the undirected edges. So if I iterate over all s, t cuts I get the minimum cut so what is the runtime of this algorithm runtime is we go of I am this while this for loop is run mod V times.

And this many times I need to run the max flow algorithm each execution of max flow algorithm using Edmond curve algorithm of augmenting using un-weighted shortest path takes V times e square time. So this is e could be s_i v_i V to the power 6. So which is not very efficient it is polynomial but not very efficient. This is because e size of E is R of size of $|V|^2$.

Now next what we will see we will see an easy randomized algorithm due to David Karger and that is called Karger's algorithm. So what is this algorithm? This algorithm is very simple what it does is picks a random edge and merges the 2 end points so the algorithm is. So let me first explain this algorithm with an example and then we will write the code pseudo code A, B, C, D what it does it?

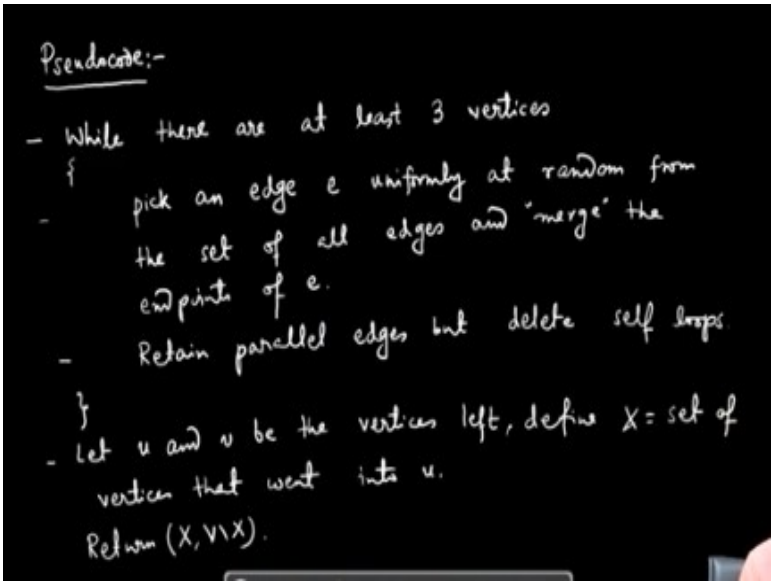
First in the first iteration it picks random edge an H uniformly at random so suppose this edge is picked and then what we will do is? That we will merge or will collapse these 2 the 2 end points so that means that I remove vertex a, and vertex B and introduce a new vertex. Let us call it A B

and F C D and so this C D H is there but the age there is the age from C to B that H I u add with the super vertex there is an edge from B to D I add this Edge from d and this new collapsed vertex and there is an edge 2 a 2 c so I R that is also c2a and there is an edge from A to B so this edge is there is this self-loop. What I do is that you know? I retain parallel edges but delete self-loop no you may keep the self-loose but that is not going to affect the result.

So deleting the self-loop's only make the bookkeeping easy so let us delete the self-loop's. But parallel edges are very important. So now we have 4 edges again the algorithm picks one edge uniformly at random and collapses it so suppose it picks this edge say the straight edge then our new graph. In the new graph we will have a super node A, B, C and this is D and there are 2 parallel edges A B to C and A B to D and there are 2 self-loops because of the 2 edges between A B and C and the self-loops I ignore.

The algorithm stops when we have only 2 vertices and it returns this cut. The one and the corresponding vertices of these 2 nodes are is the cut so X is A B C and of course $V \setminus X$ is D. So returns this cut and in this particular case is indeed a min cut.

(Refer Slide Time: 15:11)

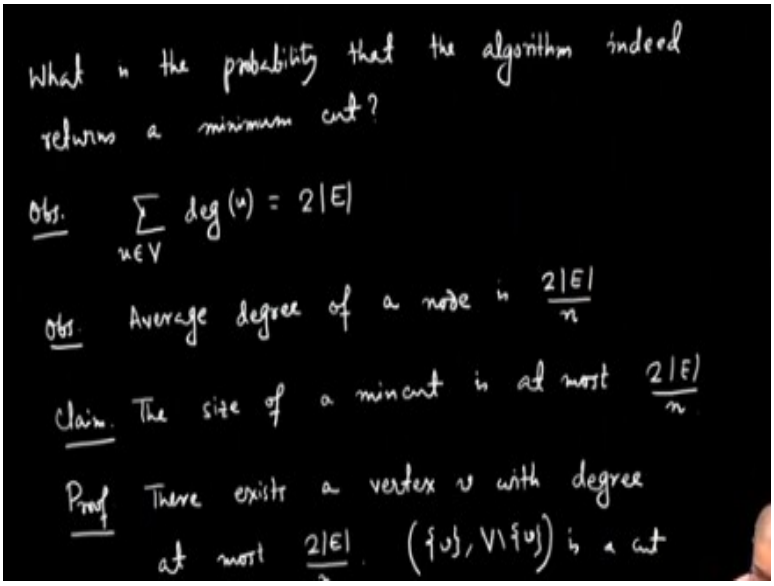


So now let us write this pseudo code of cardinals min cut algorithm. So while there are at least 3 vertices so algorithm terminates when there are 2 vertices algorithm is very simple pick and edge e uniformly at random from the set of all edges and merge the end points of E. We specify explicitly that retain parallel edges but delete self-loops. And let the last 2 vertices let u and v be

the vertices left define say X equal to set of vertices that went into u return $(X, V \setminus X)$ as the min cut.

So it is a randomized algorithm and its runtime is a so picking an edge uniformly tandem and once the edge is picked it can be merged in order n time and this while loop runs at most n minus one times or n - 2 times y while loop runs at more exactly n - 2 times. Because in each iteration of the while loop the number of vertices drops by one so the overall run time of this procedure is $O(n^2)$.

(Refer Slide Time: 19:19)

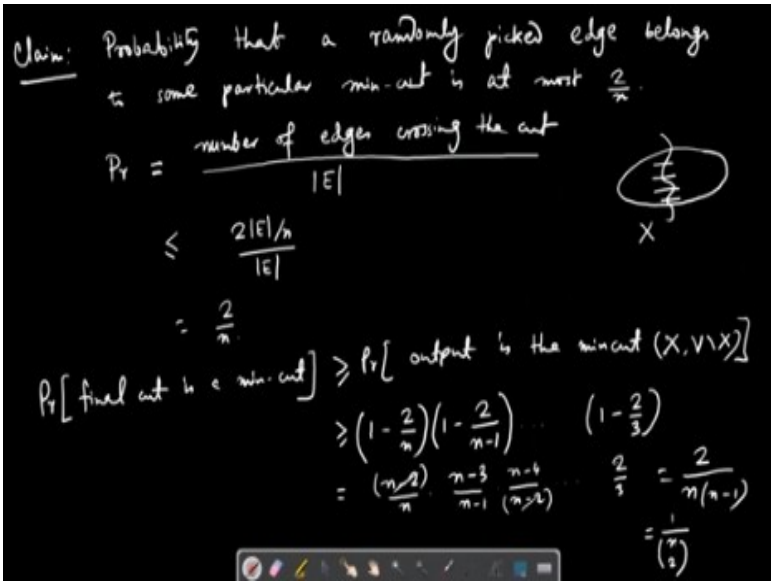


Now what I need to do is what; is the probability that the algorithm indeed returns minimum cut? So towards that let us make some observations so first observation is sum of degrees of all the vertices you in set of vertices is twice the number of edges this is a fundamental result basic result in graph theory and it follows from handshaking limber. So if you hand shaking argument and if you have not seen it I will let you do let you prove it as homework.

So next if the average degree of a node is you know sum of degrees is 2 E and average degree sum of degree by n so the average degree is $2\frac{|E|}{n}$. The next observation or claim the size of a min cut is at most $2\frac{|E|}{n}$. Why because the average degree is at average degrees $2\frac{|E|}{n}$ so let me prove

it. Since average degree is $2\frac{|E|}{n}$ there exists vertex V with degree at most $2\frac{|E|}{n}$. And you know V and set of other vertices is a cut this is a cut of size at most to $\frac{|E|}{n}$ so the size of the minimum cut can only be lower than this.

(Refer Slide Time: 23:19)



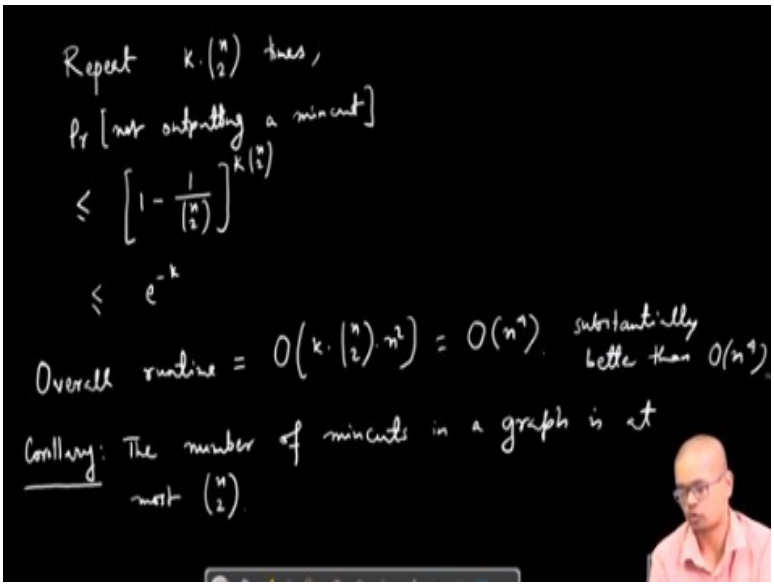
So now next claim probability that randomly picked age belongs to some particular mean cut is at most $\frac{2}{n}$. You know there could be more than 1 min cut so mean cut is not unique so what we will show is that you know let us fix some in cut from in our mind and so these are the edges crossing the cut and for with respect to this particular min cut. Let us ask what is the probability that a randomly picked edge belongs to this cut edge is at the edges which crosses this cut so this probability is at most 2 n simply.

Because this probability is number of edges not crossing the cut belongs to this so number of edges crossing the cut by total number of edges. And the number of edges crossing the cut is at most $2\frac{|E|}{n}$. So probability that the final cut is min cut this is greater than equal to the probability that output is this particular min cut is the mean cut $(X, V \setminus X)$ and what is the probability that this particular mean cut is?

Output is that whenever we pick an edge no cut is picked no h crossing the cut $(X, V \setminus X)$ is picked so what is the probability that the first age that is being picked does not belong to the cut so that is at least $1-\frac{2}{n}$. In the second iteration what is the probability that the age peaked does not belong to this cut this is $1-\frac{2}{n-1}$ because in the second iteration we have n - 1 vertices and so on this the last term is $1-\frac{2}{3}$.

This is that n -2 this cancels and similarly n -3 cancels with next term and so on so what remains is 2 by n times n -1 which is like 2 by in so 1 by n is 2.

(Refer Slide Time: 28:35)



So if we repeat so this error probability is small but if we repeat say K times n choose 2 times then probability of not outputting min cut is that it will fail to find min cut in both in all these iterations so in one iteration it fails with probability $1-\frac{1}{n}$ is 2 and this should happen every time. So this k times inches 2 this is less than equal to e^{-k} so if we pick some good constant. Then this error probability becomes small so what is the overall running time including repetition.

Overall run time is O of k is that if k is a constant $k\,{}^{n}C_{2}$ and one iteration takes n^2 Times so if k is a constant then V is $O(n^4)$. This run time can be further improved by modifying this Karger's algorithm appropriately. And so it is this algorithm has also some one interesting coordinate is

one important corollary which I would let you prove as homework is that the number of min cuts in graph is at most ${}^{n}C_{2}$.

So no graph; cannot have more than inches to number of min cuts that sort of interesting quality and also one comment is that you know this algorithm works only for this analysis is n to the power 4. And this so this is substantially better than you compare it than we go of n to the power 6 which we got by naive application with max flow so we will stop here today.