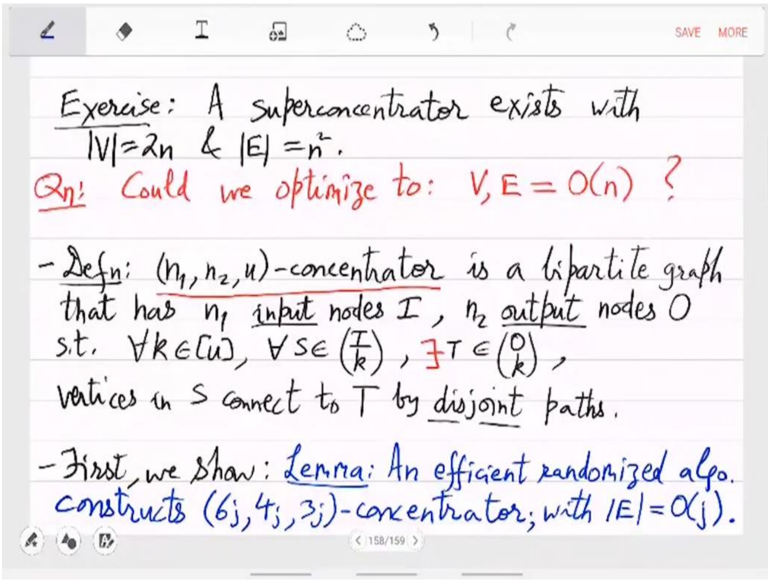
Probability for Computer Science
Prof. Nitin Saxena
Department of Computer Science and Engineering
Indian Institute of Technology - Kanpur

Module - 8
Lecture - 30
Super Concentrators

So, the construction, in fact, even the existence that we are interested in of a super concentrator, it should be a graph family; so, n tends to infinity. And for growing n, you want the vertices and the edges, both of them to be order n. And remember that there should be this very strong connectivity property via disjoint paths.

(Refer Slide Time: 00:41)



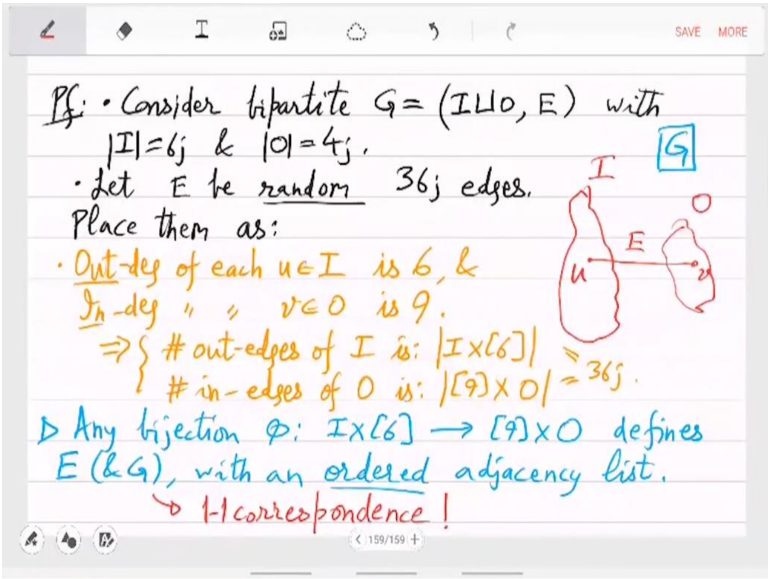
So, we will give this construction by probabilistic methods and we will define first a weaker object, just a concentrator. So, n 1, n 2, u concentrator; this is a bipartite graph that has n 1 input nodes; let us call them I as before; n 2 output nodes; let us call them O; such that for all k up to u; so, you can think of u vertices of I and that of O. So, for all S, I k; k vertices. And similarly, k vertices of O. Again the same thing; vertices in S connect to T by k disjoint paths.

And the key thing here is that the quantifier will be flipped. So, it is not for all; this is, there exists. So, this is the only change in concentrator. So, up to this bound u, upper bound u, you can take k vertices, call them S. And for those k vertices, there is some subset T, such that S to T, there are disjoint connections, disjoint paths. So, obviously, this, there exist is much weaker than for all.

So, for every S, there is a different T possibly, and other T's may not work. So, this is a weaker notion. We will first construct this. First we show this lemma that concentrators do exist, and with very few edges and vertices. So, an efficient randomised algorithm constructs a 6j, 4j, 3j concentrator of size linear in j. So, this I union O, the vertices, that is only 10j. And the number of edges will also be some constant times j.

So, this is really what we want for super concentrators, but this is the first algorithm to actually construct concentrators. Then we will use that to construct super concentrator. So, not only do concentrators exist, we will actually optimally find them. So, let us do that.

(Refer Slide Time: 05:16)



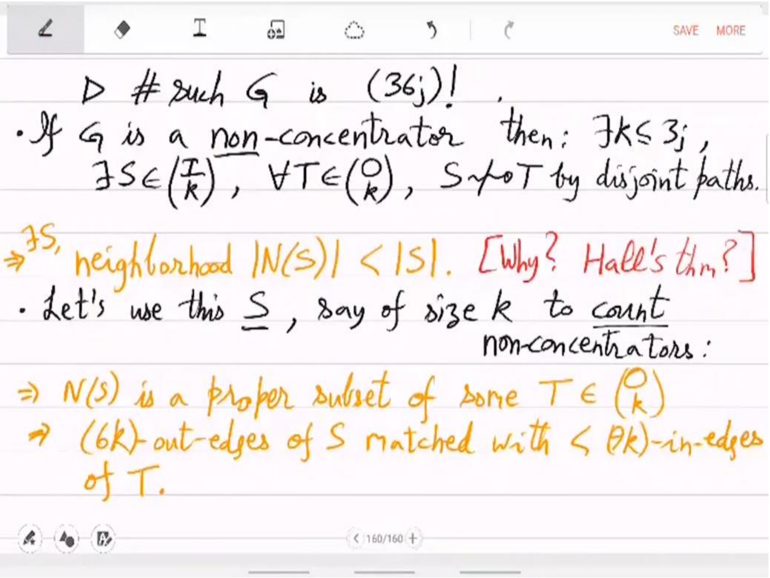
Let us do the construction and it will also be an algorithm, randomised polynomial time algorithm. So, consider a bipartite graph G, which is I disjoint union O. So, vertices are partitioned into input and output nodes. And then there are edges across I O, with I 6j and O 4j. Let E be randomly chosen 36j edges. So, think of this as a larger I and a smaller O. And then, the edges are like this.

So, the edges are only across; there are no edges inside I or inside O, because we want a bipartite graph. So, of course, between 6j and 4j, there could have been 24j square edges, but we may do with 36j only. So, pick them in a random way. Place them as; so, these are the conditions, special conditions. So, out-degree of each u in I is 6, and in-degree of each v in O is 9. So, if you look at the edges out of I, so, number of out-edges of I is I times 6.

And number of in-edges of O is 9 Cartesian product O. And both these are, as you can see, 36j. So, representation of this graph G, we will think of as adjacency list representation where the neighbours are in order. So, it is an ordered adjacency list. So, we can first look at the vertices in I and place the 36j edges. And the same thing we can do symmetrically on O. So, what you get is, any bijection from the I side to the O side, it defines the graph with an ordered adjacency list.

So, that is the one-to-one correspondence between the graph and the bijections. So, one-to-one correspondence is the given, this way. Now, how many bijections are there? That is just like permutations; so, 36j factorial.

(Refer Slide Time: 10:44)



So, number of such G is 36j factorial. So, out of these, now, we have to see whether there is a concentrator. How do we check that? So, for that, we have to see whether vertices up to 3j on the left and on the right; on the left if you choose, then, whether there are disjoint paths to some T in O. So, we have to check that. So, let us instead count the number of non-concentrators. So, let us do that.

So, if G is a non-concentrator, then there will be an S which will not have a T. So, then, there exists an S in I, such that for all t on the right-hand side, what happens? Something bad happens, right? This disjoint path connection is not there. S is not connected to T by disjoint path. And there exists a k smaller than 3j. So, there is an S which is bad, which means for all T, disjoint path connection is not there.

So, from this, what you can deduce is that the neighbourhood of S, this size is less than k. This is by something called Hall's theorem. So, read about this. This is a theorem about bipartite graphs especially. So, it basically says that if for all S, the neighbourhood was at least k in size, then there would have been a matching. So, then the above thing would not have happened.

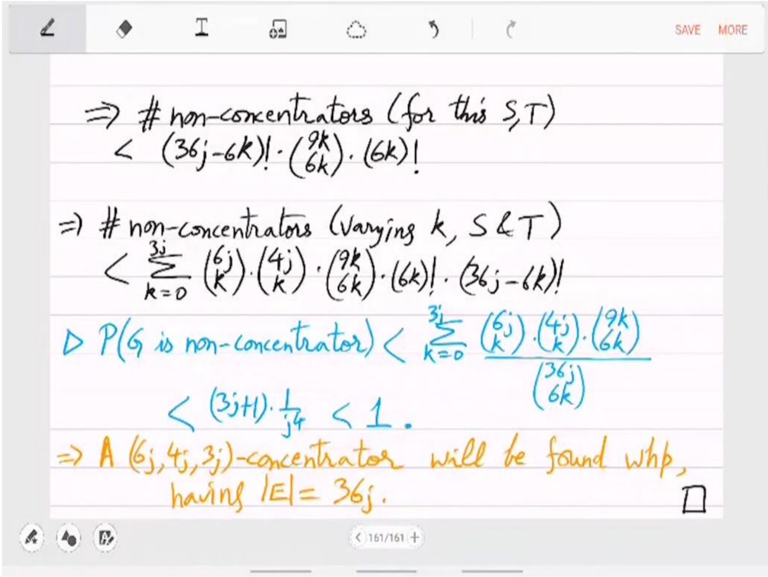
So, since it is happening, then it means that there is an S which has actually, even the neighbourhood is small. And let me put here that in fact there is an S with small neighbourhood. It may not be the same S as before, but there will be an S of size k, some size such that the neighbours are fewer than itself. So, now let us use this S, say of size k, to count. So, what will you count? We will count the number of non-concentrators.

So, basically, as you are picking these random edges, what is the chance that you hit a non-concentrator? Is the probability 1. So, we will actually show that the probability is not 1. So, let us do that to count non-concentrators. So, first thing, the way we read this orange statement is that, neighbourhood of S is a proper subset of some k sized T. So, N S is a proper subset of some k set on the RHS, which means that these 6k out-edges of S, they are matched with less than equal to 9k in-edges of T.

So, look at S; the edges which go out of S, they are 6k many, and they are all going inside a T which is of size less than k. And the number of in-edges to them is less than equal to 9k. I can even put less than 9k here. And so, these two have been matched in the way we are building this graph, the bisection that we had before, the one-to-one correspondence between the edges and the permutations.

So, basically, the 6k are being matched by these very specific 9k. And this already will be enough to give you the result, if you do the count carefully. So, let us do that. All these will give you non-concentrators.

(Refer Slide Time: 17:40)



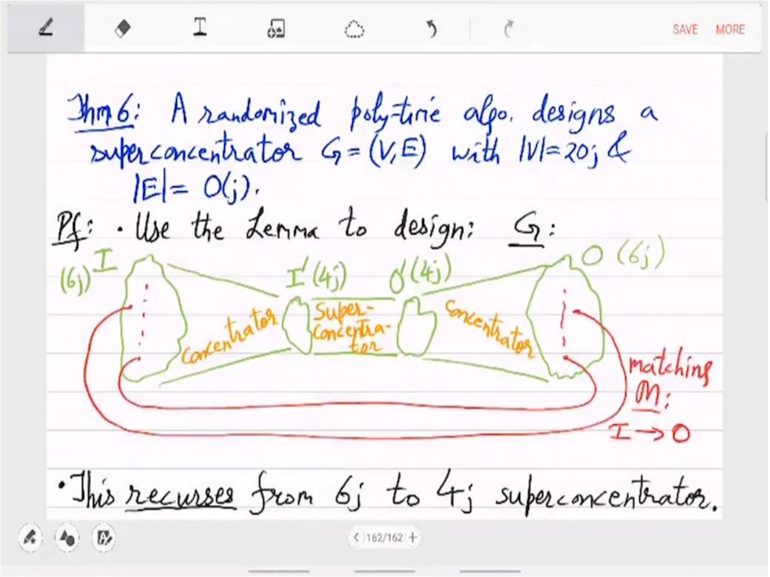
So, let us do that calculation. So, this implies that number of non-concentrators for this S and T is less than equal to; so, other than these 6k, there is no condition. So, 36j - 6k, you can permute as you wish, only for this 6k you have to map them to the 9k ones. So, that is how many options? So, out of 9k choose 6k, and then you can permute them; so, 6k factorial. This very lazy count; actually get strictly smaller than; so, which means that number of non-concentrators.

Now, you vary k, S and T. So, if you vary that, then what do you get? Then you have to sum up the above value. So, that is k equal to 0 to 3j, because k can go with maximum to 3j. And for that, how many S and T? 6j choose k times 4j choose k, times 9k choose 6k, then 6k factorial and 36j - 6k factorial. That is the number of non-concentrators, upper bound; so, which means that the probability of hitting a non-concentrator, that G is non-concentrator; it is less than the above thing divided by 36j factorial.

So, you get sum of all these k's 6j choose k 4j choose k and 9k choose 6k divided by 36j factorial; and with these remaining terms, you actually get 36j choose 6k. So, that is the probability. And with some effort, basically look at the value k = 0 and the n value k = 3j, and from that you can deduce; the way this function grows, you can deduce that. The sum is actually small.

This is actually smaller than; each of these summons in fact, they are smaller than 1 over j to the 4. So, it is in particular less than 1. So, the probability of hitting a non-concentrator in this process is actually not 1, which means that there is a concentrator. So, this means that a 6j, 4j, 3j concentrator will be found. It will be found with high probability, and it will have edges as we used, only 36j. So, this probability or simply counting a proof does give you concentrator as promised. Now, from concentrator, how do we go to super concentrator?

(Refer Slide Time: 22:28)



That is one more step. So, that will be theorem 6. So, a randomised poly-time algo designs a super concentrator G, vertices V, edges E, with number of vertices 20j, and number of edges, some multiple of j, order j definitely. So, this actually, you can now use concentrator to build this. This is not very hard. Once you have a concentrator, you can use 2 concentrators to get to this. So, this is how we will build it. So, use the lemma to design the following network.

This is I and these are the input nodes, output nodes. How many are there? So, this number is 6j and this number is also 6j, because of the 6j in concentrator. So, from I, you first use the concentrator to go to I prime, which will be 4j vertices. So, that is the concentrator business. And similarly, here there is a concentrator. So, I to I prime is the concentrator we built in the lemma. And then, O to O prime is also the same concentrator, but placed kind of in this reverse way.

And I prime to O prime, you basically recurse down and use a super concentrator. So, this part is a super concentrator. So, this is almost the final construction. We used 2 concentrators or 2 copies of the same concentrator and 1 smaller super concentrator, and also have matchings; finally, have a designated vertex. So, these are actually matchings I to O. So, just I and O have the same number of vertices. You basically create a bisection and call that M.

We will call that the matching M. And the claim is that this graph is a super concentrator from I to O. It is a recursive construction. So, design this. So, this recurses from 6j to 4j super concentrator. So, this is the recursive construction. And in every step, you need 2 copies of the concentrator. So, why would this work? We will analyse this next time.