

Arithmetic Circuit Complexity
Prof. Nitin Saxena
Department of Computer Science and Engineering
Indian Institute of Technology-Kanpur

Lecture - 05

Last time we saw that ABP can be expressed as a determinant.

(Refer Slide Time: 00:17)

Corollary: $\text{per}(X_{n \times n}) = \sum_{C \in \text{cyclecover}(G)} \text{wt}(C)$

— Lemma: If f has width- w depth- d ABP, then it has a $O(wdn)$ -size determinant.

Proof: • Firstly, make the edge-weights literals (i.e. $\{F, U, \bar{x}\}$).
→ width becomes $O(wn)$, depth is $O(d)$.

• Let G be the directed graph underlying this ABP.
 Modify it to G' :

- Add a $\text{wt}=1$ edge from t to s .
- On all other vertices add self-loops of $\text{wt}=1$.

▷ $C \in \text{cyclecover}(G')$ uniquely specifies a path $s \rightarrow t$ in G , of the same sign ($\text{wt}G=1$)

▷ Converse is true.

$\Rightarrow \det(A') = \det(A)$ $\Rightarrow \#V(G') = O(wn) \times d$ 22/46

So, width w depth d the ABP this can be written as a determinant of a matrix of size of the dimension wdn and is the number of variables in f . As the proof was very simple it was just converting. You are given an ABP with source vertex s , destination vertex t . Every path from s to t that contribution you get by a cycle. So give a reverse edge from t to s of weight 1 and that will make a path into a cycle.

So every path becomes a cycle. And together with that cycle, you can also use the self-loops to look at it as a cycle cover. So every path actually corresponds to a cycle cover and that gives you the determinant by this combinatorial interpretation of determinant. Any questions? Now we will come to the more interesting part which is the converse.

(Refer Slide Time: 01:38)

- More surprising result is:
Theorem (Mahajan, Vinay '97): \det_n has a width- $O(n^2)$ depth- $O(n)$ ABP (over any commutative ring).
 $\Rightarrow \det_n \in VP(\text{depth-}O(\log n))$ (P-uniform) (unbounded fanin/fanout)
 - The main tool in the proof is a relaxation of disjoint cycles to closed walks (while still computing \det_n).
 - Defn: let G be a graph on $V(G) = [n]$.
 A clow of G is a closed walk of length, say l , such as $C = (v_1, v_2, \dots, v_l, v_1)$ with v_1 being unique minimum.
 Call head(C) := v_1 . (head doesn't repeat)
 A clow sequence is a clow-tuple (C_1, \dots, C_k) with increasing heads, i.e. $\text{head}(C_1) < \text{head}(C_2) < \dots < \text{head}(C_k)$.

So we now want to convert determinant into an ABP. And this will actually improve algorithms of determinant. That we have seen till now. So the determinant, we will show, has a width n^2 depth or length n ABP, okay. And all we will use is commutativity of the entries in the matrix. So what will be the algorithmic implication of this?

Well this will actually mean that determinant is again in VP as I mean which we have already shown, but the depth will be better. So depth will only be $\log n$ now. Depth will be $\log n$ and this will be a completely uniform way. You will get a fast algorithm to also compute this representation given an n , okay. So the advantage is that the depth is now $\log n$ in comparison to in contrast to $\log^2 n$.

That we saw using the other method, using Newton identities. One point of difference is that now the fanin will be unbounded. So depth will be $\log n$, but in contrast to the other the older circuit where depth was $\log^2 n$ and fanin fanout was 2. Here we are making the fanin fanout arbitrary. So it will be bounded by the size of the circuit only and reducing the depth to $\log n$.

And this cannot be improved any further. We do not expect this to be improved at all. So this really gives you an optimal result for determinant computation using parallel algorithms. If you want a fast parallel method to compute determinant, this is the

implementation to be used, okay, this is a unique result. We do not have anything better than this. No, so every addition, multiplication gate should be seen as a chip or it is simply a microprocessor. So if you have these, let us say size, if the size of the circuit is n^4 , so for size to solve to compute, size n determinant, you pay the price of manufacturing these n 's, n^4 many microprocessors. And then whatever input matrix is given $n \times n$ matrix, the time to compute the determinant is only $\log(n)$. The parallel complexity, time complexity is only $\log(n)$ for this $n \times n$ determinant.

All of them work simultaneously. So they are working simultaneously. The microprocessors are actually working in parallel, but their time will not add up. The physical time which you will see will only be the length of the path. Theoretically this is a nice result because it shows now equivalence of ABP model with the symbolic determinant model. These two models are the same.

So determinant is something unarguably it is fundamental and we are getting this model to be equivalent to this new model which we had defined ABP and this we have defined in a completely different way. Still they are equivalent? So that is a nice theoretical result. So how do you show this? How do you convert determinant into an ABP? So determinant is defined, we saw the combinatorial interpretation using cycle covers.

So you would want to convert or implement cycle cover as a path now and this does not seem easy or even possible. It will be a non-trivial, it will be a complicated proof. So what we will do is we will actually prove something more about determinant, more properties of determinants. The main tool in the proof is a relaxation of disjoint cycles which appear in a cyclic cover to closed walks while still computing determinant.

So we had, in a cycle cover we use disjoint cycles of course. We want to relax it to closed walks. So the difference would be that now you will be allowed to actually repeat vertices. Okay, these will not be simple cycles and moreover across so in the

case of cyclic cover across cycles, the vertices were disjoint. That will also be false now. So across these closed walks, vertices may actually be repeating okay.

“Professor - student conversation starts” Same cycle won’t repeat again? Yeah, so all kinds of repetitions are allowed now. **“Professor - student conversation ends”**. Within a closed walk you can repeat a vertex and across these, across closed walks also you can repeat vertices. So there will be only one thing which you would not be able to repeat, and that is called head of the walk.

So we would want heads to be distinct. So let us now define this formally. So let G be a graph on vertex set $[n]$. So what we will call clow, closed walk. So a closed walk or a clow of G is a closed walk of length l , say $C = (v_1, v_2, \dots, v_l, v_1)$. So it is a closed walk but the vertices other than v_1 maybe repeating with v_1 being unique and minimum.

So it is the unique minimum here, okay. So we are thinking of the vertex set as numbers $1, 2, 3, \dots, n$. So there is an ordering and a closed walk is just you are basically walking in a sequence of vertices. So obviously, v_1 to v_2 there should be an edge and then v_2 to v_3 there should be an edge. So you are taking, you are picking these edges walking along them, but everything is allowed except you should not fall below v_1 , okay.

So v_2 to v_l all of these should be strictly bigger than v_1 . So that is the only constraint. Everything else you are allowed to do. So this is this designated vertex is called head. So, $\text{head}(C) := v_1$. So head does not repeat, okay. That is the thing to remember. It does not repeat as in it does not repeat in the middle. When you are repeating it the closed walk ends, the walk closes.

Okay, so the walk gets closed at the point you come back to the head. Any questions? So that is one closed walk. So this is a relaxation of a cycle. Now to get a cover you want to look at a sequence of closed walks with increasing heads, okay. So a clow

sequence is a clow tuple. So a tuple of clows or a collection of clows C_1 to C_r with increasing heads. That is $\text{head}(C_1) < \text{head}(C_2) < \dots < \text{head}(C_r)$.

They are distinct and in increasing order. Note that here we are not putting the extra condition of covering all the vertices. They may be missing some vertices, but it would not be important. For the final result this covering will not be important. So just think of this clow sequence as a set of closed walks, closed heads are strictly increasing. Within a clow you are allowed to do whatever you want. Just respect the head. Any questions?

“Professor - student conversation starts” So, when we were doing the reason we were able to find that connection was because cycle cover that has very close link with permutation. Sure. Is there something. **Professor:** Yeah sure. This will also give you the determinant. That is the observation by, well invention and observation by Mahajan, Vinay. **Student:** And right now is there any link between clow and Professor: yes a cycle cover is a closed sequence. That is the connection. Student: Okay, but it is not the other way round. **Professor:** Well, so for we give it a sign, okay. **“Professor - student conversation ends”**. So for that we first define length.

(Refer Slide Time: 14:18)

- The length of a clow sequence is the sum of lengths of underlying clows. (length = n)
 - The sign of a clow sequence is $(-1)^{\# \text{even-clows}}$.
 - The weight is product of weights of underlying edges.
- ▷ A cycle cover is a clow sequence of the same wt. & sign.
- The surprise is:
- lemma [MV'97]: If A is the adjacency matrix of G , then
- $$\det(A) = \sum_{C \in \text{clowSeq}(G)} \text{sgn}(C) \cdot \text{wt}(C)$$
- Proof: Key idea: The contributions of clowSeq, that are not cycle covers, cancel out! (Join/Break clows.)

So the length of a closed sequence is the sum of the length of the underlying clows. So naturally it is just these vertices which you are seeing. So in a clow you will see vertices v_1 back to v_1 and then sum it up over all the clows in the sequence. And then the sign of a clow sequence should be, what do you want to define the sign as minus so parity of some number which should be the length.

Well okay not the length. So you have to again you have to define it analogous to cyclic covers. So in the case when you are looking at a cycle when your clow sequence is a cycle cover what is the sign? Number of even cycles.

The sign of a clow sequence is $(-1)^{\# \text{even-clows}}$.

This is inspired from the sign of cycle covers. Otherwise, there is really no motivation why you should look at the parity of even clows, number of even clows just to be consistent with cycle cover, we define it like this. And now what will happen is that these clows, clow sequences which are not cycle covers in the sum we want them to cancel out and this is what we will show that, they indeed cancel out with the sign. Okay, that is it.

When once we have shown that we will have the connection with determinant. So let us just recall also the weight definition. So the weight of clow sequence this should be the product of weights of the underlying edges. So length of a clow I think is then number of edges you see, not the vertices. It is the number of edges, which was also the case in cycle cover.

And weight is just you, you take the product of these edge labels. Okay that would be the weight. So every clow gives you a signed weight. Every clow sequence you will just multiply these to get a sign weight for the clow sequence and, oh, no you repeat it. If an edge repeats you have to multiply. The idea is that it will not matter because there will be a clow sequence of opposite sign with the same weight.

So they will cancel anyways. Okay cancellation is the magic here. **“Professor - student conversation starts”** It is not bounded. The length of a clow sequence? Yeah, that is a good point, yes. **“Professor - student conversation ends”**. So is it bounded? The length of a, the number of clows is bounded because the heads have to increase and within a clows, within a clow does the definition allow doing this infinitely many times?

The definition as it stands seems to suggest that you can do this infinitely many times. So I guess no, in the end, I think we will just put a bound on this. So do not, we have to see in the implementation, but there will be a bound. So we will not be doing this infinitely many times. Within a clow they are just a natural bound say more than n , say n^2 is what is where we will stop.

Within a clow sequence the number of heads cannot be more than n and within a clow also you do not have to look at all these arbitrarily long clows. It will be clear from the implementation when we want to compute it. But for now let me just continue with more observations. So first simple observation is that a cycle cover is obviously a clow sequence of the same weight and sign. That is by design.

So this should be clear just compare the definitions. And so the surprising thing is that this sum will also give you determinant. So Mahajan, Vinay[MV'87] that paper shows this lemma: If A is the adjacency matrix of a graph G , then,

$$\det(A) = \sum_{c \in \text{clowSeq}(G)} \text{sgn}(c) \cdot \text{wt}(c)$$

So the attraction here is this equality, . You know that determinant is equal to the sum if you went over all the c 's that are cycle covers, that we had seen before. Now we are saying that even if you relax cycle cover to a clow sequence the sum is the same. It does not add anything. In other words, so to prove this it is equivalent to proving that C 's that are not cycle covers, they will cancel out.

“Professor - student conversation starts” So again like he pointed out if we did not put that bound, this bound will be a finite. ? yes. **Professor :** Exactly yeah. So once

you have a cycle you can just keep repeating it. Sure. **“Professor - student conversation ends”**. So we will see that in the implementation. So for now just think of a very large upper bound. We will exactly see what that is. I think it will simply be n .

The ABP which will construct will be of depth n and it will just use n edges, not more than that. So the idea is that you should be able to cover subsume all the cyclic covers. And if you think of a cycle cover there how many edges do you see? You see only n edges. So you restrict your clow sequence, clows to be just n . In fact the clow sequence you restrict to be n .

You do not need to go above that, because within that, you will be able to subsume any possible cycle cover and the rest of the things will be produced with in different ways with different sign and they will cancel out. So n will be the correct bound. So maybe I mention it here. So $length \leq n$. Just we are putting that restriction but you can take it to be any bound because then it does not matter.

Just for finiteness we put that bound. So the thing we will show is exactly that. We will show cancellation of bad clows. Key idea is the contributions of clow sequences that are not cycle cover, they cancel out. This cancellation we have to show. Usually showing this cancellation is I mean in a complicated computational model showing cancellation is highly non trivial.

Here it will be made easy because we will actually show that if you look at a cycle cover with a sign that is, you look at a clow sequence with a sign that is not a cycle cover, then by a simple manipulation you can get another clow sequence with opposite sign. Which is in this one? it will be in this sum, because it will be a very simple manipulation. We will just take two clows.

Either we will join them or we will break a clow into two clows, okay. So either join or break. So this will change the parity.

“Professor - student conversation starts” Joining with length restriction is a bit tricky, right? **Professor:** No, but those edges are already there. Joining means, so the length is not changing. **Student:** So it is just like a union. **Professor:** The length is not changing but the sign is. The sign will change. **Student 2:** But the sign depends upon the length right? **Professor:** No, it depends on the number of even clows. **Student 1:** Still it's a bit tricky. **Professor:** So we will see it in the proof and then we will also see in the implementation. **Student 1 :** Edges are allowed to repeat if we have a length less than n but still only got say length n but you only use half the edges, You have another clow sequence that also have length n , that only uses half the edges. **Professor:** That is fine. **Student 1 :** Suppose you have to take the union of them to cancel but only other are sequence of s to n . **Professor:** Sure. But that will be the job in the proof. So the operations will be join and break. Okay.

(Refer Slide Time: 26:56)

- Consider a clow sequence $C = (C_1, \dots, C_k)$ of length $l = n$.
If C is not a cycle cover, then some vertex repeats.
- Let $i \in [k]$ be the largest st. $C_i = (v_1, v_2, \dots, v_k, v_1)$ has a vertex that repeats. $v_1 \in V(G) = [n]$
- $(\Rightarrow (C_{i+1}, \dots, C_k)$ are disj. cycles, but (C_i, \dots, C_k) are not " " .)
- This can happen in two ways:
case 1: $\exists j' < j \in [k], v_{j'} = v_j$ (C_i not a cycle)
case 2: $\exists j \in [k], v_j$ occurs in C_{i+1}, \dots, C_k .
- Over the cases, pick the least j .

So consider a clow sequence $C = (C_1, C_2, \dots, C_l)$ increasing heads of length l . So you see l edges exactly in this and they may be repeating all that is possible. So if C is not a cycle cover what does it mean? This happens if and only if the vertex repeats so that is possible in two ways. So then some vertex repeats. Yes, that is true. So shall we fix it l to be n . Okay. Let us do that. So length is exactly n .

So length of a clow sequence you fix it to be n and so in C you see you see n edges, but this is not a cyclic cover it means that some vertex is repeating okay. Let i be the

largest such vertex. Well okay let i be first the largest clow. So let $i \in [r]$, be the largest such that this clow C_i which so now we go inside this largest clow that is bad. Well not largest clow but clow C_i for the max i .

Let us go inside this. So say $C_i = (v_1, v_2, \dots, v_k, v_1)$. Okay this is the clow. Has a vertex that repeats. so either there is a vertex. No, the C_i in this which is not a so C_{i+1} to C_r that is a partial cycle cover. C_i with C_r it is not.

“Professor - student conversation starts” If C_i is not C_1 we probably won't be able to write it like this because the clow sequence the heads are increasing. Okay. So C_i cannot have v_1 . **Professor:** Why not? v_1 is not 1. v_1 is an arbitrary label. v_1 is just a variable. **“Professor - student conversation ends”**.

The vertices are labeled by $1, \dots, m$. v_1 can be anything in that set. Why? No. It is v_1 , v_1 is a variable. It is a vertex variable. $v_1 \in V(G) = [n]$.

So this in particular tells you that (C_{i+1}, \dots, C_r) are disjoint cycles, but when you look at (C_i, \dots, C_r) that is not. They are not disjoint cycles. This is clear? So why are they not, why does including C_i violate the disjointedness of cycles? Well, either because C_i is not a cycle or it is a cycle and something here overlaps with some other vertex in C_{i+1}, \dots, C_r . There are two ways. So this can happen in two ways.

Case 1: $\exists j' < j \in [k], v_{j'} = v_j$. So v_1 to v_k are already repeating. So within so which means that C_i is not a cycle. It is not a simple cycle.

Case 2: $\exists j \in [k], v_j$ occurs in C_{i+1}, \dots, C_r

So note that both the I mean these cases are not exclusive. Both the things may be happening. So we can set the convention that first you check case one and if it fails only then you go to case two. So in case two when you come then I mean I do not think it will help. You could have we could have assumed that C_i is a cycle. So all

these C_i 's to C_r are cycles, but there is a vertex in C_i that overlap. So they are not disjoint. But they are cycle. So that you can think of case two as that.

And actually no. So we, this would not be the convention. So convention will be over these cases, pick the least j . We will go with the least. Okay. So look at both the cases and pick the vertex that is the that has the least subscript index, j . Okay, so but still, what could happen is yes.

“Professor - student conversation starts” Student 1: I have a question. We said C_{i+1} to .. **Student 2:** We can make it a single case. v_j occurs in C_i to C_r . **Professor:** Yeah but I do not want to do that. In one case I will do join clows. In other case I will do break clows, break a clow or break into clows. So I will join clows or I will break into clows depending on which case I am in. Well, so you can only guess. At this point, so we have suppose you are in case one which is C_i is not a cycle, what do you do? Can you modify this and get another clow sequence? **Student:** you take away. **Professor:** Exactly, you break it. So you can already guess that case one is friendly with the breaking operation. So you break C_i and get two get a different clow where the number of clows has increased. No, so we will only break into two, we can fix that. **Student:** And there is only one way to break it even if it occurs multiple times. **Professor:** Yeah, right. **“Professor - student conversation ends”.**

And case 2, v_j actually overlaps with say C_i and C_{i+1} . So in that case, you are seeing a connection between C_i and C_{i+1} . So this is a good case for joining. So you join the two with respect to v_j . So you get a get an associated clow sequence. So we have set up an association amongst clows, clow sequences that are bad, that are not cycle covers.

So this thing you cannot do with cycle covers, but with clow sequences, you can do this and we so what remains to be shown is that this is actually, this partitions the bad

clow sequences into two parts and sets up a bijection with signs opposite. So you can exactly see how things are cancelling out with the sign. So that is what remains.

“Professor - student conversation starts” So how do we show that when we are clubbing these two sequences the head., **Professor:** Yeah so we will see that in the now in the analysis. **“Professor - student conversation ends”**.

So now we analyze the break and the join. So let us go to the next page. Just remember these definitions.

(Refer Slide Time: 37:47)

In case 1, vertices v_{j+1}, \dots, v_j are all distinct. So, this gives a (sub-)cycle.
 • Define a new clow sequence C' by breaking C_i into $\text{clow1} = (v_1, \dots, v_{j'}, v_{j+1}, \dots, v_k, v_1)$
 $\text{clow2} = (v_j, v_{j+1}, \dots, v_j)$ is a cycle.
 [Heads distinct? $\text{Head}(\text{clow1}) = v_1 < \text{Head}(\text{clow2})$.
 cycle clow2 is disjoint from C_{i+1}, \dots, C_r]
 [\Rightarrow We can order the $(r+1)$ -heads.]
 $\triangleright \text{wt}(C') = \text{wt}(C)$.
 $\triangleright \text{sgn}(C') = (-1)^{e+r+1} \leftarrow \text{sgn}(C) = (-1)^{e+r}$.
 Pf: $\text{sgn}(C) = (-1)^{\# \text{even-clows} = e}$; $\# \text{odd-clows} = r - e$.
 • We have $e = \sum_{i=1}^r |C_i| \equiv_2 r - e \equiv_2 r + e \Rightarrow e + r \equiv_2 e$. \square

So in case one, vertices $v_{j+1} \dots v_j$ are all distinct because they pick the least j . So before j nothing repeats. So we assume, let us point that out. So we assume the $v_{j'}$ and v_j to be the same. And since j is the least, between j' and j , you do not see a repetition. Okay, so this gives a cycle.

So let us say a subcycle and we can use that this subcycle to define a new clow sequence (C') by breaking C_i into $\text{clow1} = (v_1, \dots, v_{j'}, v_{j+1}, \dots, v_k, v_1)$ And $\text{clow2} = (v_j, v_{j+1}, \dots, v_j)$. Should I say v_j prime comma, should I add v_j prime also in this? So that is how my notation was. So, $\text{clow2} = (v_{j'}, v_{j+1}, \dots, v_j)$.

So clow1 is v_1 to v_1 with the subcycle removed and clow2 is the subcycle and everything else C_{i+1}, \dots, C_r we are not changing. Okay, so what do we gain with this? Well, first we have to check whether this is a clow sequence. So let us check that. Are the heads distinct? So what is the head of clow1? That remains the same because we have only removed higher vertices. So v_1 remains the main, it is unique.

In clow2 what is the main? Is that the main? It should be j' . We said that $j' < j$. So that is the index. So it should be equal to, so how do you show that there is nothing smaller in the middle? No so I do not think I want to say that v_j is the head here. No, no the value, the vertex value should be minimum and non-repeating. Should be unique-minimum.

Then it could be anything. So suppose the minimum in clow2 repeats. I think clow 2 is just a cycle, it is a simple cycle. That is what it should be. That we have already gotten. They are distinct. So actually this is a simple cycle. So something, the minimum would be unique. I do not know what it is, it is something.

All I want is that the head of clow1 should be less than that of clow 2.

That is true because $\text{head}(\text{clow1}) = v_1 < \text{head}(\text{clow2})$. No I do not want equal to, it should be strictly. So v_1 is the unique minimum in C . So because of that. So head of clow 2 is increasing and well but there are also the C_{i+1}, \dots, C_r .

“Professor - student conversation starts” But they can be the same. **Professor:**

They can be the same? **Student:** The head for clow 2 can be in those C_{i+1} .

Professor: Yeah, so we have to show that they are at least distinct, even if not ordered. So this cycle is disjoint from the rest. That you would need. So clow 2, cycle clow 2 is disjoint from C_{i+1}, \dots, C_r . Do you believe in this? **Student :** I mean if it

occurs, it occurs in exactly one of them. **Professor:** What occurs? **Student:** The head

of clow 2 occurs in. **Professor:** No, I want to prove something even stronger. That

clow 2 is actually disjoint from the next. **Student :** And also if it does intersect if we

try to go the other way, okay it does intersect. Then I mean if poly intersection is only one of them right? **Professor:** No I want to show it does not intersect. How do I deduce that? I think here I have to use then if it does intersect then you will have to go to case 2 probably. I think I have to switch between the cases. **Student:** What was Case 2? **Professor:** Case 2 was exactly this that C_i intersects with something else, something next, C_{i+1}, \dots, C_r . There exist a v_j that occurs in, I mean, v_j is something that occurs in C_i and also somewhere else. **Student:** Then we can keep the cases separately? I mean case 1, if case 2 does not happen. **Professor:** Yeah. So we have to switch between the cases Yeah. So that switching will be needed. **“Professor - student conversation ends”**.

So here we can assume that clow 2 is that actually clow 2 is the subcycle is disjoint from the rest. This is logical consistency and it has to be carefully checked. But let me just proceed. So I am basing my deductions on this that clow 2 is disjoint from the rest. And hence, so in the beginning you had C_i, \dots, C_r with heads distinct.

And when you broke C_i then you got vertices that are different from v_1 and they are different from you got a head actually, you got a head of clow 2 that is different from v_1 and it is also different from the heads of C_{i+1}, \dots, C_r . So hence the heads are disjoint, the heads are distinct and you can order them. Look at that clow sequence. So we can order the heads, order the $r + 1$ heads.

Okay, and what else do I need? So now more important conclusions from this: that $wt(C') = wt(C)$, that is obvious, because the product would not change when you do these, do this break and reordering. What about the sign of C' ? So I want to make this claim that $sgn(C') = (-1)^{l+r+1}$ and $sgn(C) = (-1)^{l+r}$. Let us focus on this.

So if you look at the original clow sequence, the parity of the number of even clows matches with the parity of $l + r$. l is the number of all the edges you see, actually it is

fixed to be $n +$ the number of clows. Why is that? It is a simple calculation. By definition sign of C is $(-1)^{\#even-clows := e}$ and any ideas?

And let us say odd, so number of odd is then $r - e$, okay. So if you look at this,

$$l = \sum_{i=1}^r |C_i| \equiv_2 r - e \equiv_2 r + e \Rightarrow l + r \equiv_2 e$$

That is the proof.

So, $sgn(C)$ was total number of edges plus the number of clows and in the same spirit $sgn(C')$ is total number of edges plus the number of clows which has increased. So the parity changes, it flips. Is that clear? So that is a good process.

“Professor - student conversation starts” So this clow 1 into clow 2, so does it not make the potential case for case 2. The j prime is on clow 2.

Professor: What do you mean? **Student :** So is it not making into clow 1 and clow 2? **Professor:** No, so we are done with this process. We are now looking at C' which has one more with the same amount of edges. **“Professor - student conversation ends”**.

So we said the convention here that if both cases are applicable then use case 2, okay. This is just a convention to break the tie.

So since we are breaking the tie we can assume here that the disjointedness because otherwise we will be using case 2. So there is some logical stuff you have to check. So that we are not in a vicious cycle but I think it is fine. So let us now quickly do the joint operation in case 2.

(Refer Slide Time: 52:56)

• In case 2, $v_j \in C_i \cap C_{i'}$ for $i < i' \in [r]$.

Here we join the clows C_i & $C_{i'}$ at the vertex v_j to get clow $C_{i'} := (v_1 \rightarrow v_{j-1}, \underbrace{C_{i'}}_{(v_j \rightarrow v_j)}, v_{j+1} \rightarrow v_k, v_1)$

Call the new clow sequence C' .

[$\text{head}(C_i) = v_1$, heads in C' are in the same order as C]

$\triangleright \text{wt}(C') = \text{wt}(C)$

$\triangleright \text{sgn}(C') = (-1)^{i+i-1}$ & $\text{sgn}(C) = (-1)^{i+i}$

\triangleright He above gives a map τ from $\text{ClowSeq}(G) \setminus \text{CycleSeq}(G)$ to itself st.:

- τ has no fixed point.
- flips the signed-wt.
- τ is invertible.

τ is an involution

Case 2 we have this $v_j \in C_i \cap C_{i'}$ for $i < i' \in [r]$ and this i' is unique. Why is that? No it is not because of that, it is because $C_{i+1} \dots C_r$ are cycles. They are disjoint cycles. Yes. Oh you meant for the i . So the way i was defined was largest i such that you get a bad clow. So after this you only have good clows which are all cycles and they are also disjoint.

So vertex v_j of C_i reappears, it can reappear only in one place. So that is $C_{i'}$. So now, since C_i and $C_{i'}$ have this connection, we will just use it to join them. So in this case the number of clows will fall by one with the same effect on the sign. No, do not break. So here we join the clows at the vertex v_j to get clow $C_{i'}$. So this we have to write as a sequence of vertices.

So let us so we are filling inside C_i . So C_i go up to v_j , v_1 to v_j and then you can take a detour inside $C_{i'}$. So follow the map inside the cycle $C_{i'}$. So that would be $C_{i'} - \{v_j\}$. So break that cycle by removing v_j . Put all those vertices in that order here in this place. And why v_{j+1} ? Well because we have removed the first v_i in $C_{i'}$.

So it will come back to v_j and then from that you can proceed to $v_{j+1} \dots v_k, v_1$. That is the description. Sorry. So say it removes from the first because that we have included or we can just write v_{j-1} yes, that is true. So from after v_{j-1} you go to v_j in $C_{i'}$.

Think of $C_{i'}$ as headed with v_j . So v_j is the first then you will then it will complete the cycle come back to v_j and then move to v_{j+1} ..

That is the order. So think of this representation:

$$C_{i'} = (v_1, \dots, v_{j-1}, C_{i'}, v_{j+1}, \dots, v_k, v_1) \text{ where } C_{i'} = (v_j, \dots, v_j)$$

So all the edges are available and you can do this close walk, starts and ends at v_1 . Okay, and with this call the new sequence C' that is the new clow sequence which has now one less clow. We have to now do those checks. Is the head what is the head of $C_{i'}$? head $(C_{i'}) = v_1$. That is because the $C_{i'}$ that we have put inside the $C_{i'}$ that we have put inside the C_i that these are all higher vertices.

So v_1 remains the unique minimum. That is good. And the heads are in the same increasing order. Because we just deleted $C_{i'}$. So delete $C_{i'}$ and then the head sequence is the same. So heads are distinct. So this means that it is a valid clow sequence. So that proves that it is we can call it a clow sequence. And then we can talk about weight and sign.

Yeah, so $wt(C') = wt(C)$ because we have not lost any edge. Neither have we added a new edge. Sign calculation is the same as before. $sgn(C') = (-1)^{l+r+1}$ and $sgn(C) = (-1)^{l+r}$. Is this clear? So the sign has flipped weight remains the same.

So it is now fair to say that the above description gives a map τ from $ClowSeq(G) \setminus CycleCover(G)$ to itself. So this is defined only on non cycle covers. Because otherwise you would not be able to get into case 1 or case 2. And from such a clow, from a bad clow sequence, you are only going to a bad clow sequence. You cannot go to a good, you cannot go to a cycle cover.

So it is a map to itself and additional properties are τ has no fixed point. Well obviously because it is changing the number of clows. So there is no fixed point. It does not fix anything in this set. It moves everything. It flips the sign. Flips the signed

weight. It only flips it. It does not change it by any other way. And but you need one more thing

You need for bijection, it should be invertible. So why is that? Well, because it is a finite set and we have defined τ on everything. So with these three properties on the map τ is actually partitioning the set into two halves and moving anything from the left to something in the right. And when you apply it again you come back to that okay. So these are called involutions. So τ is an involution. No, exactly n.

So it is an involution and it is moving everything. Or you can just directly show that τ^2 is identity. So by finiteness, I think it is, it is already there, but also explicitly, you can see that if you apply τ again two times. So basically you, you break and then , so then it will be breaking and joining and get to the same thing. So it is actually on every instance it is an involution you can check that.

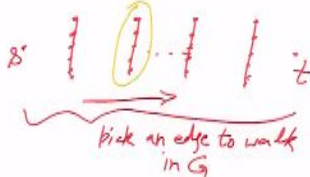
So no such loophole is there. So this is actually leading to a cancellation, okay. This leads to a genuine cancellation term by term. So one term gets canceled with its negative. It is the simplest form of cancellation.

(Refer Slide Time: 1:04:37)

$$\Rightarrow \sum_{C \in \text{Clos Seq}(G)} \text{sgn}(C) \cdot \text{wt}(C) = \sum_{C \in \text{Cyc. Cover}(G)} \text{sgn}(C) \cdot \text{wt}(C) = \det(A) \quad \square$$

(1)

Lemma (MV'97): Expr. (1) has a width- n^2 , depth- $(n+1)$ ABP, where $n := |V(G)|$.



So overall, we get this property that if you look at the signs, weight of cycle covers, oh sorry, cflow sequences of a graph then it is the same as all the bad things cancel and

you are left with cycle covers the strict subset, proper subset and that is by almost by definition, it is determinant of the A , adjacency matrix of the graph, A . So that finishes the proof. Is that clear? That finishes our theorems lemma.

This lemma is shown that determinant is equal to sum over cycle sequences and now the question arises. So this is a mathematical fact. But how will this be turned into an algorithm. So from one exponential sum, we have moved to a bigger sum. Now how can this ever help you in life if you make your sum even bigger? So the term is dynamic programming.

So using dynamic programming, you can actually implement this huge sum on cycle cover on cycle sequences. So going over permutations is impossible in an efficient way. But magically going over the cycle sequences because cycle sequences are so relaxed, that this thing you can actually implement using dynamic programming. I do not think we have enough time. This we will do in the next class. It will have a short proof.

Well dynamic programming itself would not have been enough for our purposes. We will actually do it systematically in the form of just matrix multiplication or ABP. We will just directly write down the ABP which computes, every path of which computes a cycle sequence. Well let me write down the statement first. So ultimately what you will get will be very powerful. So this expression has a nice implementation.

Lemma [MV'97]: So expression 1 has a width- n^2 depth- $(n+1)$ ABP where $n := |V(G)|$.

So you can think about this implementation why this thing, why expression one can be written as oh sorry. I am pointing to the wrong thing, cycle sequence. Yes. So the definition of a cycle sequence is so relaxed that you can walk in a nearly memoryless way. So all you have to remember is the head.

So you just remember what is the head above which you want, you are allowed to visit vertices and just continue to walk on that. At some point you decide that you want to close the walk. So you close it and in the next step you move to a bigger head creating a new clow okay. So this is what a path in an ABP will do. So every level will correspond to in every level of the ABP if you think of the levels, there is s vertex there is t vertex.

So and these will be the vertices. So the depth will basically represent these edges which you are picking to walk, to walk forward. So that will be, as you go along, pick edges. Pick an edge to walk on. And to implement that in a level, you will need vertices around n^2 . So this will basically n^2 because you want to remember the head and you want to remember the current vertex where you are sitting.

So these two things, it is basically a tuple , head comma current vertex, it is a tuple. So the values that this tuple can take is n^2 . So this within a level the number of values is n^2 . So basically this is what is your memory requirement. and then based on this we will implement. So the number of edges gives you the depth and the number of possibilities of head comma current gives you the width.

So it will be an $n^2 \times n$ ABP, okay. So we will finish this tomorrow.