

Randomized Algorithms
Prof. Benny George Kenkireth
Department of Computer Science & Engineering
Indian Institute of Technology Guwahati

Lecture – 20
Cover Time

This lecture we will learn about Cover Time.

(Refer Slide Time: 00:31)

Cover Time

Commute time

$$C_{uv} = h_{uv} + h_{vu} = 2m \times R_{uv}$$

\uparrow hitting time \uparrow # of edges \downarrow effective resistance

$C_x(G) \triangleq \frac{\text{Expected Time taken to visit every vertex at least once starting at } x}{\uparrow \text{ vertex of } G}$

Thm: $C(G) \triangleq \max_x C_x(G)$ is $O(n^3)$

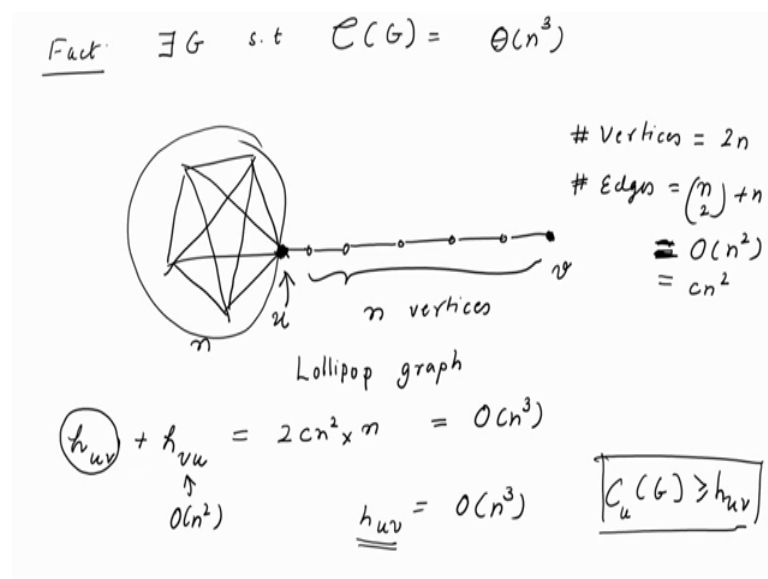
In the previous lecture we learnt an important parameter of random walks namely the Commute time ok. So, commute time was denoted by C_{uv} can be said that this is the sum of 2 parameters h_{uv} plus h_{vu} ok, these were the hitting times both of them were hitting time and h_{uv} is the time taken to start at vertex u and reach vertex v for the first time and h_{vu} is the time taken to start at v and reach back at u . If the sum of these was the commute time and we had argued with the help of electrical networks that this quantity is equal to 2 times number of edges times the effective resistance between u and v ok.

In case this is a number of edges and this is the effective resistance. We will now look at cover time; cover time basically is the time taken to cover the entire graph or visit all the vertices, so we will formally define what these are this is denoted by $C_x(G)$. So, here x is a starting vertex so this is the vertex, this is defined as time taken to visit every vertex at least once starting at x . So, imagine that we are doing random walk on a graph and if the

starting vertex is x the time taken to visit every vertex assuming we start at x that is known as the cover time or the expected time taken.

So, the time taken will be random variable its expectation is called as the cover time, the theorem that we will show or prove is cover time of a graph ok. So, cover time of a graph which is by definition equal to $\max_{x,y} C(x,y)$, so cover time is $O(n^3)$ ok. That means, no matter which vertex you take the expected time to visit all the vertices once at least once is $O(n^3)$.

(Refer Slide Time: 04:07)



Now, we will also show the following fact there exist graphs such that $C(G)$ is equal to $\Theta(n^3)$ ok, so we can construct one such graph. So, let us look at a click here this is the click on $n=5$ vertices, let us say that click on n vertices and there is a line with n vertices. So, total $2n$ vertices and edges is equal to $\binom{n}{2} + n$, so that you can say that is appropriately $O(n^2)$ and this graph the cover time is $\Theta(n^3)$ that is if you start at a vertex of course it depends on which vertex you start at.

If you start at this particular vertex let us call this vertex where the stem meets the cluster, so this graph has the special name this is called as the Lollipop graph. So, if you call this as vertex u and this end as vertex v , if we start at vertex u the time taken to visit all the vertices is $O(n^3)$ ok. So now, couple of things about cover times cover time is not a monotonic property with respect to the edges that is if you increase the number of

edges, the cover time need not decrease and if it is a complete graph the cover time is $n \log n$ approximately $n \log n$.

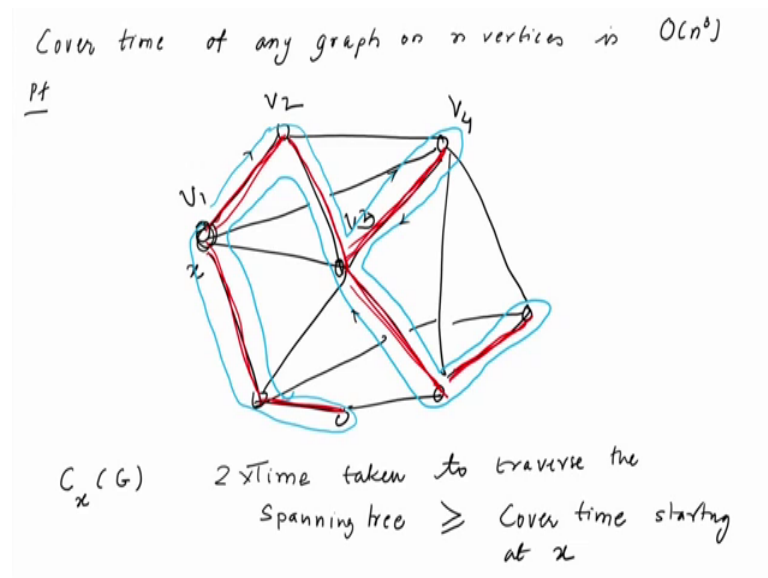
If you take the straight line graph it is n squares and something in between we can have O of n cube ok. So, we will first show that if we start at vertex u and we want to travel it will take O of n cube ok. So in order to show that we will take some results from our previous lecture, if you look at h_{uv} plus h_{vu} that is the time taken to go from u to v and then v to u that is a cover time that we know is equal to so O n square let us write this as c times n square c is some absolute constant.

So, this is equal to 2 times the number of edges which is c n square times the resistance within u and v which is about n . So, this is equal to O n cube so sum of 2 quantities is O and cube means at least one of them should be greater than n cube. In fact, if you start at vertex v and if you want to travel till u ; this is a random walk which we have already explored and we know that the time taken to do that so the first time you reach u will be the time taken to traverse this entire thing and that is roughly. So, h_{vu} this is O n square while looking at the Markovs chain for 2 stand we had explored this and showed that it is O n square.

So, since some of this O n cube h_{uv} should be so this would mean that h_{uv} is O n cube, what we will show in the remaining part of this class is that C_G is Θ n cube. Now h_{uv} is the time taken to go from u to v ; clearly $C_u G$ is going to be greater than or equal to h_{uv} , h_{uv} is the expected time taken to start from u and reach v . Well it could be the case that while moving we have not explored any vertex here, but clearly if we have covered all vertices then the time taken to do that will surely be an upper bound on the time taken to go from just u to v .

So, $C_u G$ is going to be greater than h_{uv} and h_{uv} we know is a O n cube ok. If we take the cover time starting at vertex v this is no more than n cube, what we will show is not just for this graph we will show that for the general graph any connected graph on n vertices it is cover time is bounded by n cube ok.

(Refer Slide Time: 09:24)



So, cover time that is our theorem of any graph on n vertices is $O(n^3)$; proof is fairly simple. So, let us look at cover time little more carefully if this is our graph on which we were doing a random walk ok. So, you start at any vertex and keep on moving to a neighbor and so on ok. So, let us say that we want to compute C_x of G that is if you start at vertex x how much time does it take to visit all vertex let us look at a spanning tree of this particular graph ok. So, the red edges denote a spanning tree of the graph.


So, if it travels all the edges of this particular spanning tree starting at x we are guaranteed that we would have visited all vertexs. So, we can write the following time taken to traverse this spanning tree is going to be surely greater than or equal to cover time starting at x .

So, spanning tree again when we say time taken to traverse a spanning tree we mean time taken to traverse a spanning tree starting at vertex x ok. So, let us just number these vertices in a particular way this is my vertex v_1 x I am going to calling as v_1 this v_2 this is v_3 v_4 someone traverse a spanning tree in a particular way, so from here to here then all the way here before back to here.

If I travel along with blue line that would mean I visit every vertex twice, so done this in a particular way and that would have visited every vertex twice. So, 2 times the time taken to traverse the spanning tree also greater than or equal to the cover time and in I

want to traverse this spanning tree in exact in the exact order as indicated by this particular blue line ok, traverse the spanning tree in this particular order.

(Refer Slide Time: 13:09)



$T = v_0 v_1 v_2 \dots v_{2n}$

time taken

Expected time for traversal along the above mentioned path is an upper bound on the cover time.

$v_0 - v_1$ $h_{v_0 v_1} \leftarrow$ expected time for the first edge of the tree

Expected time $\leq \sum_i h_{v_i v_{i+1}}$ $h_{uv} + h_{vu}$

$$\mathbb{E}[T] = \sum_{\substack{e \in E \\ (u,v)}} (h_{uv} + h_{vu}) = \sum_{e \in E} 2 \cdot m \cdot R_{uv} \leq \sum 2m = (n-1)2m$$

So, let us say this was a spanning tree some graph, I am starting at this particular vertex and they want to traverse along in this in some particular order ok. So, let us say the vertices are v_0 starting vertex $v_1 v_2 v$ and say $2n$, because every vertex is appearing exactly twice. So, this is the order in which vertices are traversed. So, time taken to do this the expected time taken to do this will be an upper bound on the cover time.

So, that we will just write down expected time for traversal along the above mentioned path is an upper bound on the cover time and this time can infant be viewed as. Let us say I am at vertex v_0 and from there I have to go to v_1 may be while I am doing the random walk on the graph this is not the path that is chosen; from v_0 I might go to some other vertex and after a long time I will come back to v_1 . And then from v_1 after some time I will reach v_2 from there after sometime I will reach v_3 ok. Let us look at all these as random variables, so the time taken for the traversal from v_0 to v_1 is the expected time taken is going to be $h_{v_0 v_1}$ ok, this I will think of as the expected time for the first edge of the tree ok.

So, the total expected time if I think of this the time taken for this path as T ok, the expected value of T will be the sum of those random variables. And then by linearity we can just look at the sum of the expectation and each of the individual expectation is h is

of the form $h(v, i) + 1$ ok. So, the expected value of the T or the expected time is going to be less than summation over all these is $h(v, i)$ and note that $h(v, i) + 1$ would appear on let us say this particular this is $i + 1$ and later on or if we had taken this as i and this is $i + 1$ this edge would traverse forward as well as reverse.

So, all the edges are appearing in pair and therefore this random when we look at this random variable, we have terms of the form $h(u, v) + h(v, u)$ such pair for each edge of this spanning tree. So, the total time taken if we denoted by a random variable T ; the total time taken is equal to the sum of these random variables and therefore its expectation is going to be sum over all the edges in E .

So, if this edge e is equal to u, v then this is the sum over terms of the form $h(u, v) + h(v, u)$ ok. So, what we know what we have shown so far is that the expected time taken to traverse along this particular path is going to be sum over all edges $h(u, v) + h(v, u)$ ok. And $h(u, v) + h(v, u)$ from our last class we know that is going to be equal to $2m \cdot R(u, v)$ and $R(u, v)$ note here u and v are 2 edges which are connected.

So, their effective resistance is at most 1 so this is going to be less than or equal to 1 therefore this quantity is less than $2m$ times 1 ok. So, this the number of edges is going to be $n - 1$ because it is spanning tree. So, $n - 1$ into $2m$ is going to be the time taken to traverse the spanning tree along this particular path ok.

(Refer Slide Time: 18:48)

$2m(n-1)$ is an upperbound on $E(T)$

$\frac{n^3}{2}$ $O(n^3)$ $O(n)$

Randomized Algorithm for $s-t$ connectivity $C \cdot \log n$

$s-t$

Randomly walk on G for 2^n steps, starting at s .

{

if t is found say "connected".

}

Say "not connected".

So, $2m(n-1)$ is an upper bound on the cover time. So, if we start at any particular vertex we can do the same process, so this is an upper bound on the cover time starting at any vertex and therefore this is an upper bound on C_G and m is at most n^2 . So, we know that $O(n^3)$ is the an upper bound on the cover time ok. We will use this fact to construct a randomized algorithm for S t connectivity, so we have a special situation here we have a graph which is very large and we want to know whether 2 vertices S and t are connected ok.

So, further we want to do it without using too much space we could of course use dfs and solve this problem but when we use DFS we have to store the entire graph to debug or the data structures are going to take space and that requires $O(n^2)$ space that is it is going to be proportional to the size of the input. Here let us assume that there is a read only input tape on which the entire data; the entire graph is stored, let us say an existence in matrix format and we can query whether some particular edge i and j are connected. So, this query can be done quickly we are not expected to use additional memory, so without using additional memory or let us say we are allowed to use $c \log n$ C is some constant $C \log n$ amount of memory. So, can you do a log space computation for S t connectivity.

That is I want to say I want to have an algorithm which will check if vertices S and t are connected or the algorithm will not use more than say $1000 \log n$ amount of memory. The algorithm is simple we will just start at a particular vertex and keep on going to one of its neighbors ok. So, you can just toss a coin and select one of its neighbors at random and we go to that we do not remember as to which vertex, we were in let us say previous steps we just keep tab on how many vertices have been visited so far. In other words we will do so random walk say randomly walk on G for $2n^3$ steps starting at S . At any stage at any point if you already discovered t ; if t is found say connected if you do not find it in the n th you will say.

So, after $2n^3$ step if you do not find t then you will say that they are not connected ok. Of course, if you are given an input where S and t are not connected this algorithm clearly gives the correct answer, where as if S and t were indeed connected; we want to say that with significantly high probability the algorithm would return the correct answer. Now from the previous analysis we know that $2m(n-1)$ is the amount of time

taken is an upper bound on the cover time, refer in those many steps it would have covered all the expected steps taken to visit all vertices is $2m$ times n minus 1.

So, that is something like n^3 at most n^3 . So, if you just apply Markov inequality; so which would mean that probability that there is some vertex which is not been discovered even after spending twice the expected number of steps ok. So, that probability will be by Markov inequality be less than or equal to half and therefore this algorithm gives the correct answer.

On the S instances with probability at least half you can make this $200n^3$ and then the probability would be significantly higher ok, repeated runs this algorithm can make the algorithm give more accurate answer ok. We will stop here; we will in the next lecture see more properties of random walks we will see properties called as rapidly mixing random walks etcetera.