Social Networks
Prof. S. R. S. Iyengar
Department of Computer Science
Indian Institute of Technology, Ropar

How to go Viral on Web
Lecture - 156
Myopic Search

(Refer Slide Time: 00:07)

So, we have seen that how we can write a code for making a small world network in 1 dimension. What is the next aim is, we want to implement the decentralized search or myopic search on this network and look at how does this search confirm. So, what I am going to do in this code is, we look at how do we do myopic search on this network. So, for doing myopic search we remember what we have to do is we take a node and then we look at all of its neighbors. And, then we look at the distance of all these neighbors from the target and do you remember when we look at this distance, we assume that we have no knowledge about the long range context of these neighbors.

So, we have to look at the distance of these nodes across the cycle. So, while looking at that distance we assume that there is no long link in the network and then we find their distance.

(Refer Slide Time: 01:07)

So, for that purpose what I am going to do is the graph G here and then I have added the edges here. So, before adding any long link in this network I will take a copy of this graph here H equals to G dot copy. And, why I have taken this copy here is actually very simple, the reason is very clear what is the reason when I am going to do myopic search. So, in that myopic search I have to find the distance of my neighbors from the target and by finding the distance, I do not have to see the long range ties in the network.
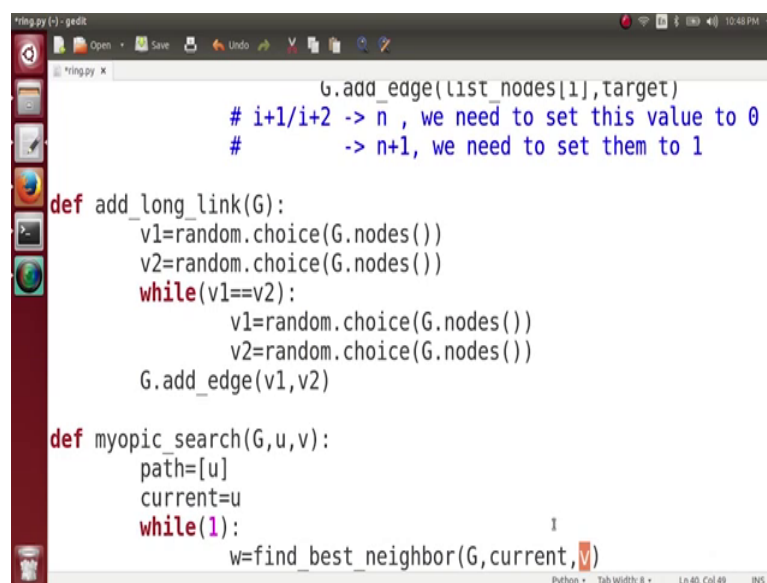
(Refer Slide Time: 01:45)

So, when I have to find the distance I look in this network edge, this network edge is this network edge. This network edge here is simply my cycle, the cycle which we have formed which consists of only the ties based on homophily and has no long range link. And, then this network G over here, G is the network which consists of the homophily based links as well as the weak ties. So, whenever I do my myopic search I will performing the myopic search on my network G, but when finding the distance of my neighbors from the target; I will be using this network edge over here ok. So, how do we proceed is I define a function myopic search over here.

(Refer Slide Time: 02:37)
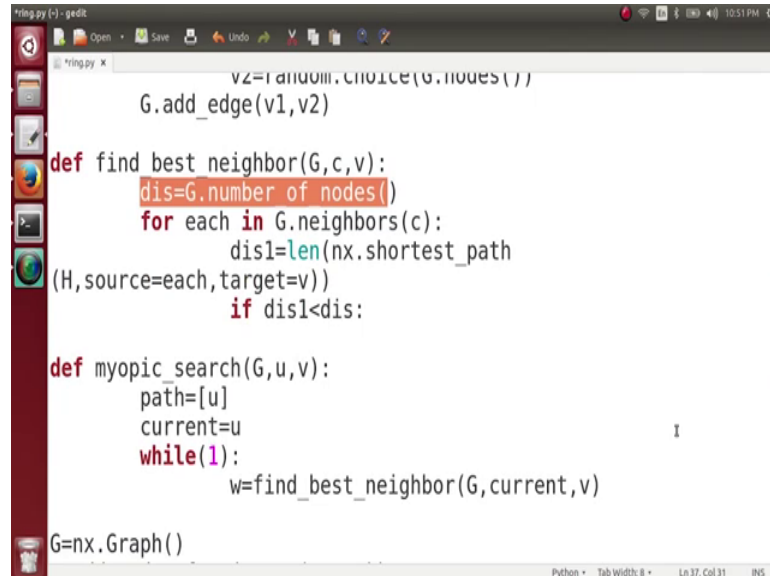

Define myopic search, I am these needs are graph the source vertex and a target vertex. So, u is my source and v is my target and we have to find the path from source and target. I take this path as a list and this path initially consists of only the node u, which is the source, current is my variable which takes care of where I am standing currently on the network. So, currently we are standing on u and while 1 what we are going to do is I take a loop over here, what we do is we have to find the next best neighbor where we have to move on.

So, I call a function here find best neighbor find best neighbor G current where, have to find the best neighbor from current and for finding the best neighbor I will be needing the target. So, what I am going to do is for finding the best neighbor this current node will be looking at all of its neighbors and we choose the one which is closest to the target

in the graph H, not in the graph G. In the graph H where there are no long range ties. So, before proceeding further let me define this function here.
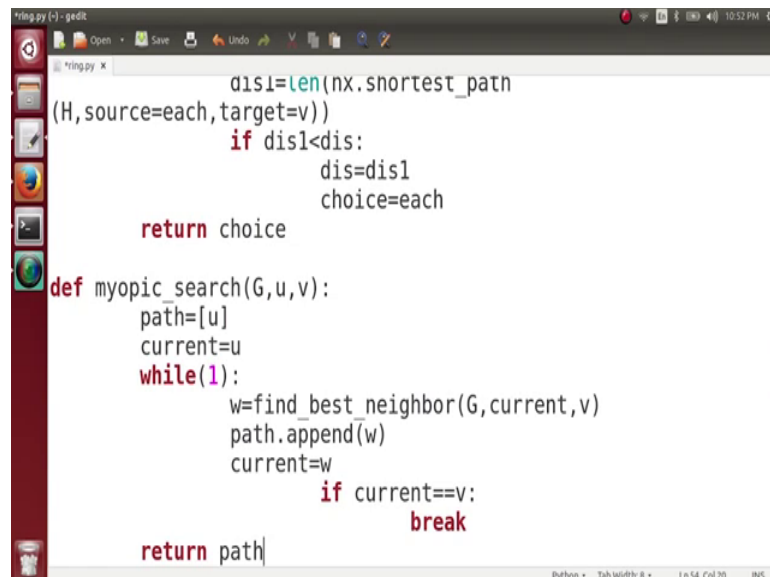
(Refer Slide Time: 04:01)

Define find best neighbor, define find best neighbor G current node and the target. So, I get a variable over here distance equals to G dot number of nodes, you will soon understand why I have taken why I am taking this variable over here. So, what do I do next is I look at each of the neighbors of this node current here. So, for each in G dot neighbors c so, for each in G dot neighbors c what I am going to do is, I am going to calculate again a distance, distance 1 which is the distance of this neighbor from the target. So, for finding a distance of the neighbor from the target we can use the function nx dot shortest path which gives us the shortest path between 2 nodes in a graph; nx dot shortest path and the graph H.

So, you understand why we are taking H here because, we have to look at the distance across the cycle not across the complete graph, we have to ignore the long range ties. So, H and what is my source. So, here this is the source for my shortest path. So, the shortest path function requires three parameters the graph, the source that is the first node and the target which is the second node; approximate I am finding the distance. Please do not confuse it with a source and target over for our myopic search, the source and target for myopic search are different and these source and target are different. So, when we were doing the myopic search, we were at a node current and now this node current is

interested in finding the best neighbor and for all of its neighbors it is calculating their distance from the target.

So, here the source here is nothing, but the neighbor of my node current. So, the source is what? Source here is each which is the neighbor and target here is v. So, the target is actually the same the source changes, now what I am going to do is, if this distance is less than the previous value of distance which was quite high. So, we want this value to be quite high and if my distance 1 it is less than distance; what I am going to do is distance equals to distance 1.
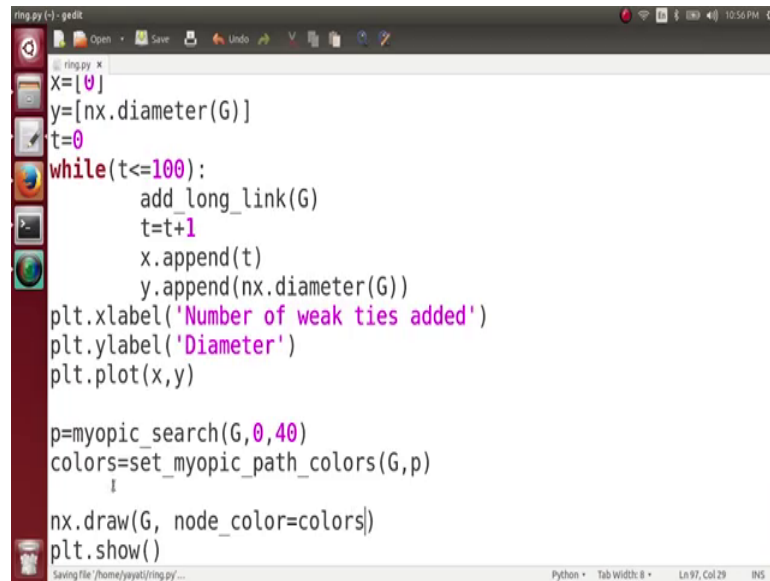
(Refer Slide Time: 06:41)

So, this is a simple method in coding which we used to find out the maximum value or the minimum value. And, I set the value of choice the best neighbor to be equal to each and at the end I return choice. So, I think that this is pretty clear and then I return back to my code for myopic search. Here I have find out the best neighbor for my node current and after finding the best neighbor I append this best neighbor to my list to my path.

And, then the value of current becomes equals to this neighbor which I have found and also by doing this if the value of current becomes equals to v which is the target; then what we have to do is simply come out of this while loop and then we can return the so, we can return path h v. So, we return path here ok. So, this is the code for our myopic search and we return path here ok. How do we execute it?
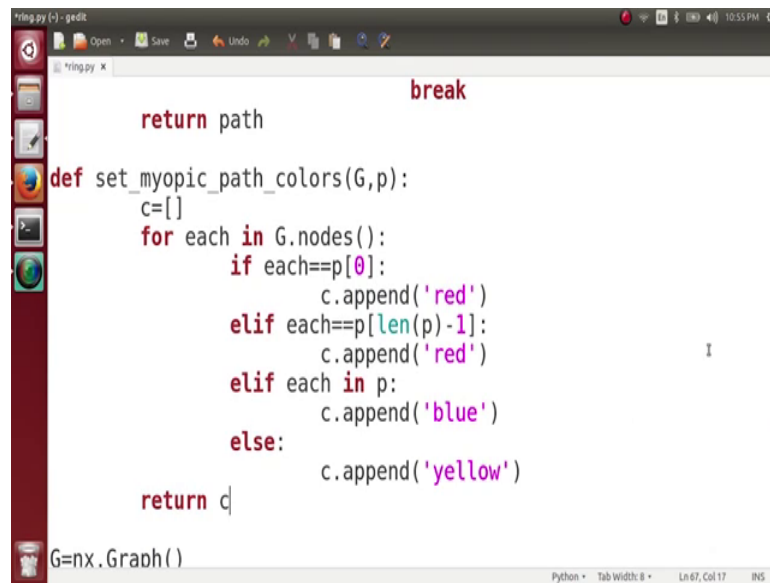
(Refer Slide Time: 07:53)

So, let us say p equals to the path which I wanted to find out and myopic search and let us say I have performed the myopic search, let us say from the node 0 to node 40. And, I wanted to see how this path looks like; I am going to play around with colors little bit here what I am going to do is, I am going to call a function here set myopic path colors why no.

So, I guess you know what I am going to do here. So, when we are going from this node from 0 to 40 I want to see how does this path look like. So, what I am going to do is I am going to define a function here, function for set myopic path colors. And, here I have to pass my graph G and the path p and it gives me an array called colors and let us define this function here.

(Refer Slide Time: 09:05)


```python
                              break
        return path

def set_myopic_path_colors(G,p):
        c=[]
        for each in G.nodes():
                if each==p[0]:
                        c.append('red')
                elif each==p[len(p)-1]:
                        c.append('red')
                elif each in p:
                        c.append('blue')
                else:
                        c.append('yellow')
        return c

G=nx.Graph()
```

Define set myopic path colors and what the function is going to do is for I have a array here c, which is for colors for each in G dot nodes. What I am going to do is, if each equals to let us say p 0 which is the starting node of our path; what I am going to do is for each in 0th node, if each is a starting node of our path and going to append a let us say red color here. And, if each equals to equals to the last element of my path then also I am going append red. What I am going to do is the source and the target here are going to be red.

The source and target here are going to be red and are going to be red we call it is actually not else if, it is elif and elif each in p ok. So, if it is the starting node of this path p, we append a red here. If it is the end path of this list p of this path p we append a red here, if it is neither starting node and but exists in p. So, what I am going to append there is let us say blue and what do I do for rest of the nodes I make them yellow.

So, all the nodes in my graph are now going to be displayed in yellow, the starting and the ending nodes are going to be red and the rest of the paths on my; rest of the nodes on my path are going to be blue. And, then I return c here you will soon understand why I am using colors here and then we got colors here nx dot draw G node underscore color equals to colors.
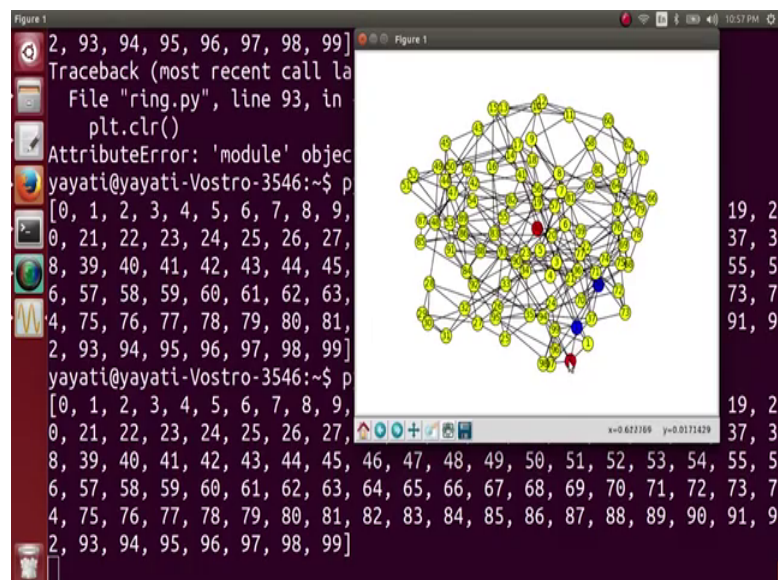
(Refer Slide Time: 11:35)

And, let us now see execute it and see line 52 shows an error. So, I have commented this portion over here and let us now execute it and see.

(Refer Slide Time: 11:47)

You can see that when we have to find a path from this node 0 here to 40. So, we can go from 0 to 2, 2 to 38 and 38 to 40. So, this is the path which my myopic search gives you and the network here looks quite random. So, we know why it looks random over here because, we have added a lot of long range links to this network. So, what I going to do is, I am going to change this value of t here to let us say 10.

(Refer Slide Time: 12:13)

So, let us add only 10 long range links over here and we know that what was the initial diameter of this network was nothing, but 25. From that 25 diameter after that I have added 10 long range links and then I find my path from the node 0 to 40 using my myopic search and let us see ok.
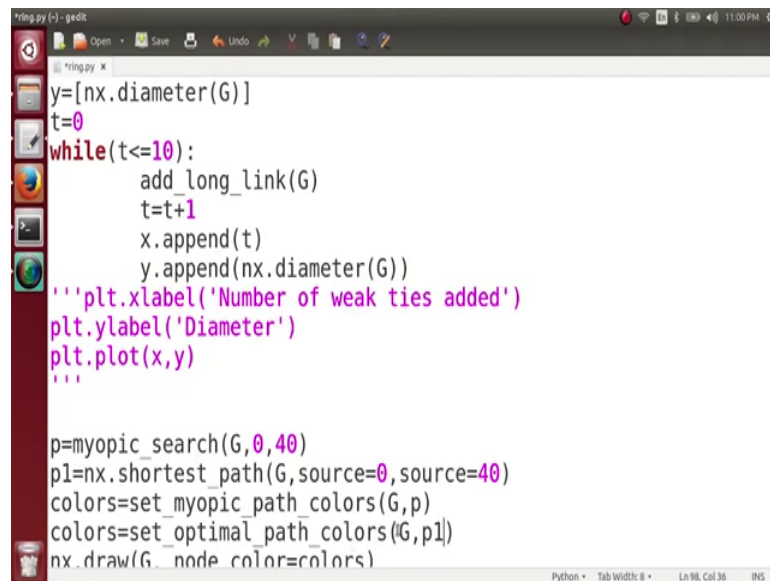
(Refer Slide Time: 12:39)
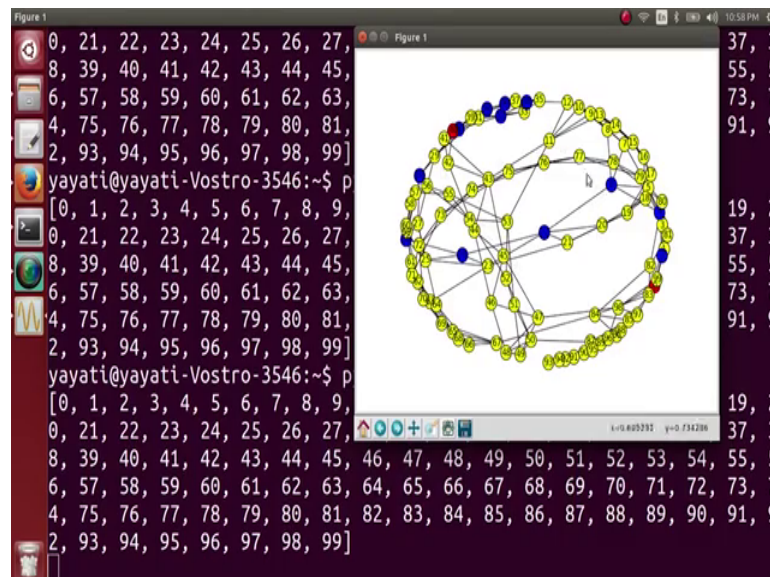
Now you see the path length has increased quite a lot and this is my ring network although it does not appear here quite like a ring. So, it start here from node 0 and there

are 1 2 3 4 5 6 7 8 9 10 11 12 nodes in between. So, within so, the path length here is 12. So, the myopic search here finds a path of length 12.

What next I am going to do is we have seen that the myopic search is not an optimal search. Let us now compare the myopic search to my optimal search and what we are going to do now, what we are going to do now is we will find the optimal distance let us say from this node 0 to 40 only.
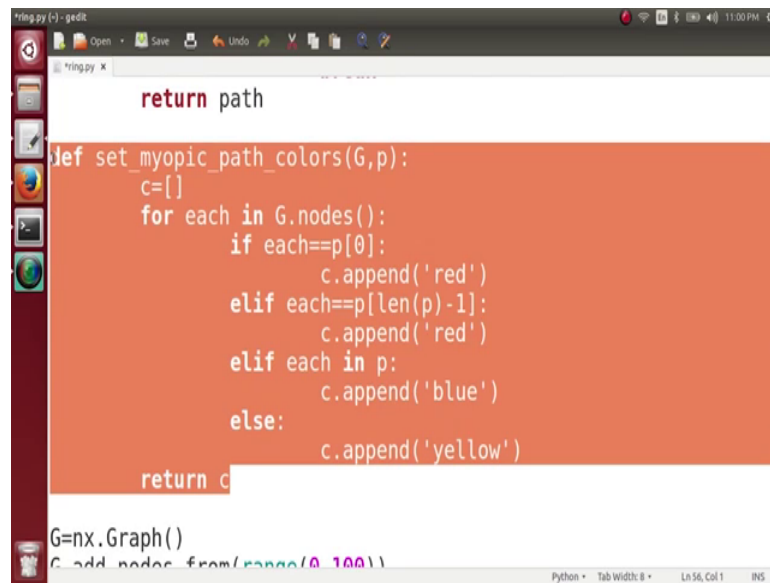
(Refer Slide Time: 13:27)


```python
y=[nx.diameter(G)]
t=0
while(t<=10):
        add_long_link(G)
        t=t+1
        x.append(t)
        y.append(nx.diameter(G))
'''plt.xlabel('Number of weak ties added')
plt.ylabel('Diameter')
plt.plot(x,y)
'''

p=myopic_search(G,0,40)
p1=nx.shortest_path(G,source=0,source=40)
colors=set_myopic_path_colors(G,p)
colors=set_optimal_path_colors(G,p1)
nx.draw(G, node_color=colors)
```

So, what I am going to do is I am going to find another path and this path is nothing, but the shortest path, the optimal shortest path on my graph G from node 0 to 40. So, you see here I am applying a shortest path on the graph G.

So, this graph G here is consisting of the long range link also and then I find p 1 and what actually we can do here is I want to see a different path also. So, here I am going to, what I am going to do is set optimal path color. So, I am also going to set optimal path colors here based on the path obtained from here from G to p 1 and then this is going to be quite easy.
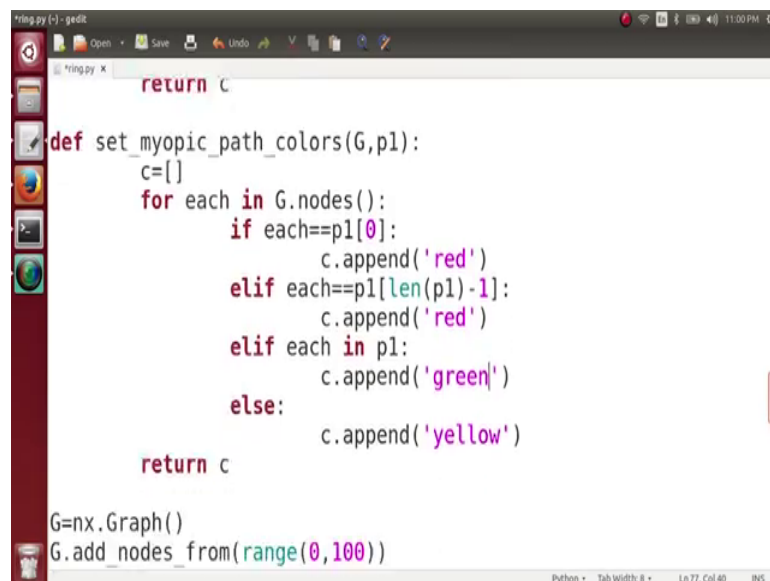
(Refer Slide Time: 14:21)

So, we are going to use the same code which is here.

(Refer Slide Time: 14:29)

And, let us say so, here it is G p 1 and then p 1, p 1, p 1, p 1 and what I am going to do is the starting node is red the ending the target node is also red. The middle nodes on this optimal paths let us color them green and the remaining nodes in the network are yellow. And then we can see and let us now execute this code and see the difference between both the paths.

(Refer Slide Time: 14:59)

Source I am very sorry source equals to 0 and this is target equals to 40, let us execute it and see set optimal path colors.

(Refer Slide Time: 15:19)

 This is set optimal path colors.
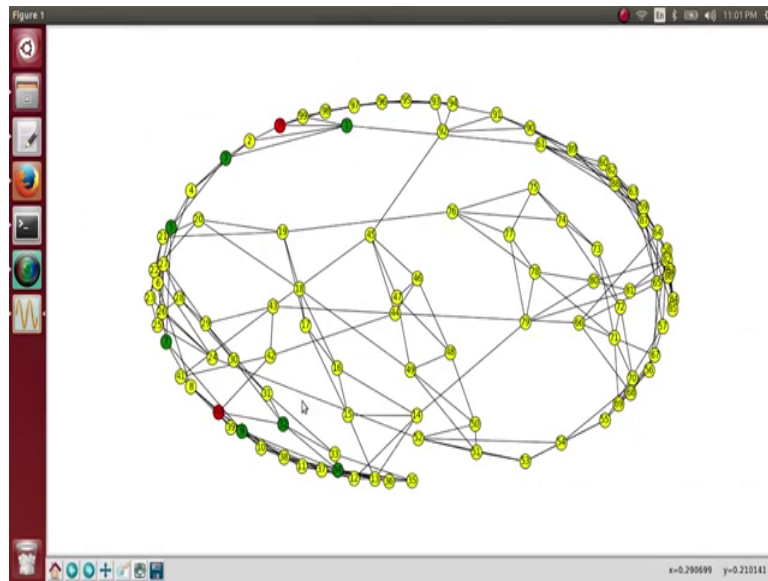
Now, you see this network over here ok. So, here is an error. So, what is the error? When I am doing the set optimal paths colors I have recolor the nodes, which I have found in the myopic path. So, we have to be a little bit careful so, what we will do is here when we have to find colors we will actually have to combine both these functions.
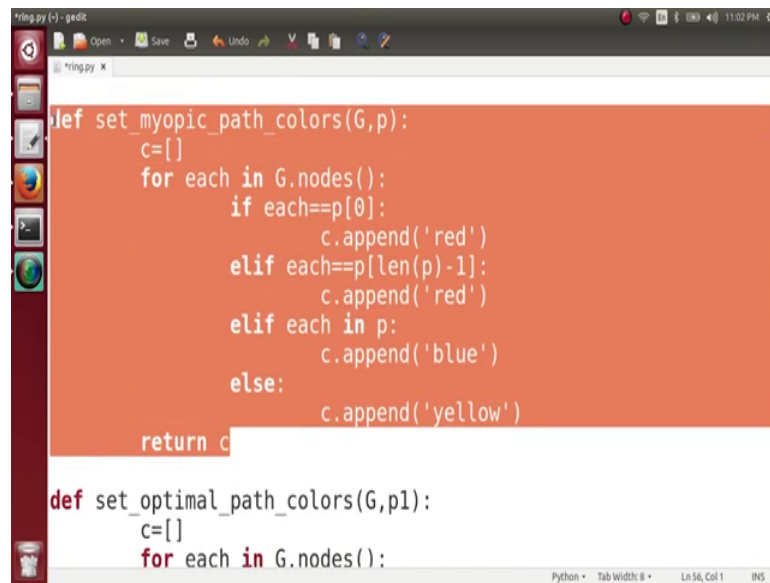
Let us say set path colors G p and p 1 ok, G p and p 1 and then we need only one function here.

(Refer Slide Time: 16:05)

Set path colors.

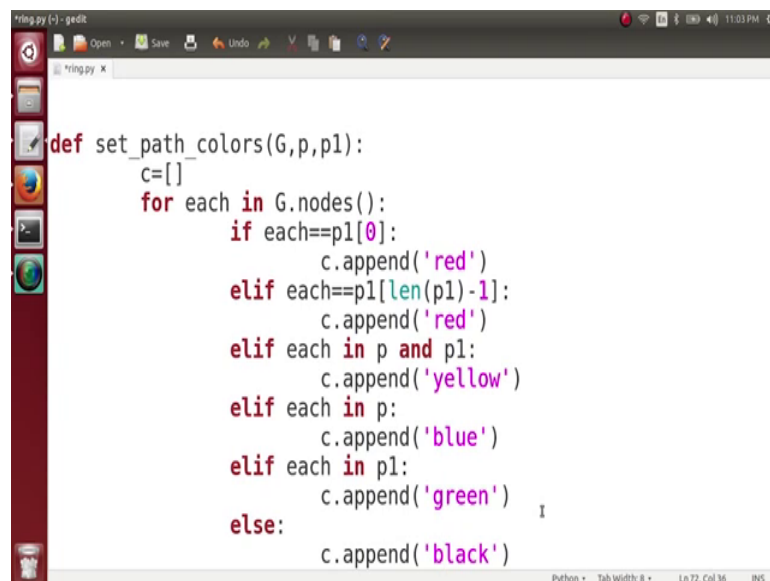(Refer Slide Time: 16:09)

G, p, p 1 and what I am going to do is; obviously, what is if each is p 1 0 it is p 0 also because, the source node is same red and then here it is red. And, it is each in p 1 when we are making it green and you see here one node can be both in p and p 1 right. So, if there is a node which is present in both p as well as p 1 and it is neither the source node not the target node we are coloring that ok.

Please be careful with the colors here, we are coloring that node to be yellow ok. So, yellow is a node which occurs in both these paths p and p 1 elif each in p, p is the path for myopic search what we are going to append here is; for myopic search we are going to use blue. And, elif each in p 1 it is then we are going to append green over here and else it is in none, we are going to append a black over here. So, please see all the nodes in this network are going to be black, the source node is red the target node is red.

The nodes which are in myopic in the path for myopic search are blue, nodes which are in path for optimal search are in green and the nodes which are in the paths for both the myopic search and the optimal search are in yellow. So, we are going to make it a little bit more rigorous and more understandable, what we are going to do is ok.

(Refer Slide Time: 17:55)

If each in p and p 1 and each is not equals to p 1 0 and each is not equals to p 1 length of p 1 minus 1, which clearly says that if each in p and p 1 it is in both the paths. But its neither equal to the source vertex not equal to the target vertex then we are going to append our yellow here, if each in p and each not in p 1 then we are going to make it blue. And, if each in p 1 and let us say each not in p then we are going to make it green. So, instead of an elif here we can use a if here, here also we can use a if and at last if each not in p 1 and each not in p we are going to make it black.
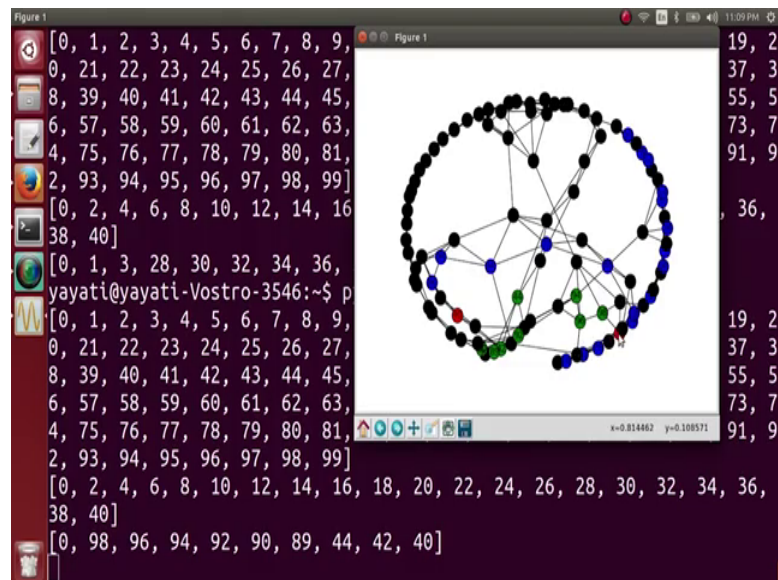
Now, let us execute it and see, from here you see the there is no node common between both of the paths. If you look at the myopic search its start from 0 and close to 40, it takes this path 0 to 2 to 4 ok; it is actually little bit difficult to make out where this path is going. So, before 30 there is 28 ok. So, this path goes something like this, the ending path is here ok. Let us see from 0 it goes to 2, 2 is somewhere here, from 2 it goes to 4 which is here and then to 6 8.

So, it goes all the way here and then it comes here, here and here and reaches 40 and why an optimal path is starts from 0 and goes like this. So, we can see that optimal path is much shorter than what are myopic search is finding out, which clearly shows that an myopic search is not the optimal search.