VLSI Physical Design with Timing Analysis

Dr. Bishnu Prasad Das

Department of Electronics and Communication Engineering

Indian Institute of Technology, Roorkee

Week - 08

Lecture 41

Finding Shortest Paths with Dijkstra's Algorithm

Welcome to the course on VLSI Physical Design with Timing Analysis. In this lecture, we will discuss how to find a shortest path using Dijkstra's shortest path algorithm. So, the content of this lecture includes Dijkstra's algorithm and how we can apply that Dijkstra's algorithm for finding the shortest path in case of a global routing. Then we will show some examples how the Dijkstra's algorithm can be applicable for a routing problem. So, what is the Dijkstra's algorithm? So, we discussed Dijkstra's shortest path algorithm in case in the first week of the lecture taking an graph. So, in that case we are considering the edges with weighted edges, but how it is different from that algorithm and this algorithm is that in that case we are considering one edge has one weight, but in this case our edge has two weight, one is horizontal capacity, one is vertical capacity. We will discuss this in an example later. So, what it does? It does it finds the shortest path from a specified source node. So, in that graph we have a source node and some termination nodes are there. So, it finds the shortest path from a specified source node to all the nodes in the graph with non-negative edge weight.

So, here we assume that all the edge weights in a graph is non-negative and we are finding the shortest path from a source node to all the nodes in the graph. Then it finds the shortest path between two specific nodes in that graph from a source node to a target node to any target node. So, this algorithm is popularly known as a maze routing. This algorithm is also known as maze routing. So, whenever we are discussing this algorithm, we have some input to that algorithm.

So, in this case the input to the algorithms are basically the graph. So, the graph V, E with non-negative edge weights W. Then we have a source node. The second parameter is we have a source node S. Then the third parameter we have a target or ending node T. So, these three are input to our algorithm. So, in the algorithm it has three different types of node. One is first one is on visited node. Then the second one is the node which is considered and the third one is the node which is already known. So, initial stage all the

nodes are on visited. After the algorithm continues, each of the nodes are considered and it finds its shortest path. Once it finds its shortest path, then the node falls into the category of known nodes. So, we have three different types of groups. What is this group? We have vertices. We have vertices in this graph. So, initially what happens is that all the vertices are basically collected in a group 1. So, all the vertices initially when we are starting the algorithm, all the vertices are in group 1. That means that none of the vertices or the nodes are visited at the starting of the algorithm. Then what the group 2 will contain? The group 2 will contain the nodes or the vertices that has been visited but for which the shortest path crossed from the starting node has not been found out. So, the node is visited but its shortest path from the starting node or the source node has not been found out. That falls into the category of group 2. Now we have the group 3. What the group 3 contains? It contains the nodes which has been visited and for which the shortest path crossed from the source node or the start node has been found. So, in the group 1, the nodes have not been visited. In the group 2, it has been visited but the shortest path has not been found. In group 3, we have visited that node and we also found the shortest path for that node. That falls into the category of group 3. Now we will go inside the algorithm. So, whatever I told, each algorithm has some input and some output. So, what are the inputs to the algorithm? Basically the weighted graph. So, the weighted graph is input to the algorithm. So, each edge has a weight w. We have a source node s. We have a target node t. We have edges of each of the edges in the graph.

We have a weighted graph and each of the edges of the graph has weight and we have a source node, we have a target node. That is given as input to the algorithm. What it finds the output of the algorithm? It finds the shortest path from the source node s to the target node t. Now we will go step by step. The first step, we have initialized all the nodes to group 1.

The group 1 means the nodes are not visited. All the vertices in that weighted graph is collected in a group 1. All the nodes are not visited. Then the group 2 and group 3 at initialization stage is phi. So, these two steps are the initialization steps. This step is the initialization step. Then we are doing for each node we are doing the traversing or visiting. We are traversing each of the nodes. For each node, this node belongs to group 1 which is not visited. We are giving a basically initialization. Its parent node is not unknown. Means that value is initially not unknown for each node. And the cost of each of the node is infinity, is very high, maximum. So, this for loop starts here and ends here. It has only 4 and 5. So, this will end here. So, it will assign the cost of each of the node as infinity. Now we have the start node cost. The start node cost is 0. Because it is a starting node, the distance will be obviously 0. The start node and the target node is same. So, initially the start node, this first one is the start and this is your target. So, both are same. So, your cost or the distance or the weight is 0. Let us take the current node as the S node.

Then the current node is the start node. Then the start node, once it is visited, then we will move that one from group 1 to the group 3. It will not go to the group 2, it will go directly to the group 3 because it is a start node. Now we will repeat that one. I will repeat that the current node till I reach the target node. This while loop will run till I reach the target node. So, until I have not reached the target node, it will continue it. If I have not reached the target node, this loop will continue. So, what I am doing for each loop? So, I am finding all the neighbours or adjacent nodes of the current node. I am finding all the neighbouring nodes of the current node and for each node belongs to group 3, I am finding the shortest path. If I found the shortest path, I am putting that node to the group 3. So, now I am finding the trial cost for each of the node which is the sum of the cost plus the weight of that edge. Now what we are doing? I am taking each node from the group 1. Then I am finding the distance, here I am finding the distance from the source to that node which is my trial cost and my parent node becomes a current node. Mean objective of this portion is to find the shortest path from the source node to each of the nodes in the graph, each of the other nodes in the graph and finally once I get that node, move that node to group 3. And finally I will reach the target node. From the target node I can find the distance from the source node. In this manner this algorithm will continue.

So, basically I have a graph G V, E with non-negative edge weights W and I have a source node or the starting node S, I have a target or ending node T. So, now we will go to an example. Before going to that example I am just again highlighting this group 1 and group 2 and group 3 again because this is very essential to understand that example. In case of group 1 is nodes which is not been visited, it happens at the initial time then we will take one node at a time to do the processing and the group 2 will contain the nodes which have been visited but we do not know the shortest path from the starting node for that node. Then the group 3 will contain the nodes which is visited and we also know the shortest path caused from the starting node to that node. The group 3 contain the nodes having the shortest path, group 2 contains the node which has not found the shortest path and group 1 contain the nodes which has not been visited.

Now we will take an example here. So, basically here what we are doing? We are finding the shortest path using Dijkstra's algorithm. I will tell you again the conventional Dijkstra's algorithm used in graph theory uses an edge which has a single weight, single weight but in our VLSI routing problem we have an edge which is having two capacity, two weights. One is horizontal capacity, one is vertical capacity. So, if you can see here, the edge has two variables. One is horizontal capacity, the first one is the horizontal and the second one is the vertical capacity.

So, we have to consider both the edge weight in case of Dijkstra's algorithm. And second thing is that whenever we are doing the conventional Dijkstra's algorithm our edges are not basically perpendicular to x axis or y axis. It is basically non-minetian. But here our interconnect will be either horizontal or vertical. Our interconnect will be either horizontal or vertical. So, these two are the difference from a conventional Dijkstra's algorithm which is used in graph theory. So, it is an extension of that algorithm. So, here we have a source node. Now our aim is to find the shortest path from the source node to the target node such that my cost, the cost in the horizontal direction and the vertical direction combined cost is minimum. Now all the nodes are in group 1. So, I am not showing the group 1 here. I am interested on group 2 and group 3 because all the nodes at starting of the algorithm is in group 1. So, we are starting the algorithm from the source node. Now the source node will directly go to the group 3 as I discussed in the algorithm because its cost is 0. Distance from the source node to the source node is 0.

So, it will go to the shortest path, it already found its shortest path, it will go to the group 3. Now the adjacent, basically the neighbors of the source node 1 is 4 and 2. So, now we find the cost of node 2 and 4. It is 8, 6. Now the node 4 is 1, 4. Now which one is the minimum among them? So, 8, 6 is basically 14, 1, 4 is the summation of w1 plus summation of w2 should be minimum. This is our objective. So, in this case if I do the sum it comes out to be 14, it comes out to be 5, then the 5 is the minimum so I will take that to the group 3 and which is the shortest path from source node to the node 4. Now I will consider the node 4 because node 4 is the shortest path. Now I will consider the node 4 which is the shortest path. Then we will see that which is the neighbors of node 4 is 7 and 5. So, the 7, the 5 will go to this one and 7 will go to this one. So, now if you can see here the weight of node 5, this node 5 is basically 10, 11. How it is 10, 11? If you can see here, so this one plus 9, 9 plus 1 is 10.

Then you have 4 plus 7, 4 plus 7 is 11. Similarly, this 9, 12 is basically found out for the node 7, this distance is found out basically this 9, the first one let us say I will write in a different color, let us say this 9 will be found out from basically 8 and 1. So, 1 plus 8 equals to 9, then this 4 plus 8, 4 plus 8 basically 12. So, now I have a group of nodes in group 2, I need to find out which is the shortest. So, basically node 2, this one is the shortest among them because this will be having 14, this will be having 21 and this will be having basically also 21. So, I will take 8, 6. Now if I go to the node 2 because the node 2, I have taken the node 2 among them. So, now I will consider what are the neighbors of node 2. The neighbors of node 2 are the 5 and 3. So, if I can see here the 5 node will be given here and the 3 nodes is given here. Now I can see how this number 9 and 10 comes into here. This 9 and 10 comes because if you can see the node 3, node 3 will have node 2, node 2 value is there this one. Node 2 value is given 8, 6. From till this point my shortest path is already there and it is 8, 6. So, I will just I do not go on to again the source node, I just want to add the corresponding is between 2 to 3.

So, here basically you have 8, this 8 plus this 8 plus 1. So, we have 8 plus 1, it will be 9. Similarly I have basically 6 coming from till node 2 plus so this 4 and this 6. The 6 plus this is 6 plus 4 it is 10. So, hence this is 9, 10.

Now if you can see here, then the W of 5 is 10, 12. How it is happening? It is the this W part is happening because 8 here, 8 plus 2 is 10, 8 plus 2 this 2 is 10 and 6 plus 6 is basically 12. So, this is 10 plus 12. Then if you can see here we have removed this W 5, 10, 12. Why we removed? Because if you can see here we have 2 node 5 is there.

Here if you can see this is node 5, 1. This is node 5, 2. So, this net weight is basically 21 and this net weight is basically 22. So, which is having the higher cost we remove it. So, we remove this one. So, we remove this from the group 2 because we have a better solution in the group 2. So, what we do? We have all these nets are there we find which is having the minimum weight cost. So, here if you can see 9, 10 we will find the net cost for each one of them 9, 12 is basically 21 again and 9 plus 10 is basically 19. So, we will take the 19 to this node. So, this is my new node 3 will be added to group 3. Node 3 will be added to the group 3. Now we will do the same thing for the neighbors of node 3 because now node 3 is there in the group 3. I will look into the all the neighbors of node 3. So, we will get node 6 only. It has only one neighbor. I look into this neighbor. I can find its weight basically how can I find its weight? So, I have this till this point my weight is 9, 10. So, the cost of till this point will be 9 plus 9. Then I have this 9 comes from here, this 9 comes from here. So, this combinedly it will be 18. Now I have this 10 here, I have this 8 here, 10 plus 8 it will be 18. So, hence my W6 becomes 18, 18. So, now I had gone till node 6 then we will see which is the shortest basically path in group 2 that we will take.

So, here if you can see that the node 5, I will find the cost of the weight. So, here it is 21, here it is also 21, 12 plus 9 is 21, now here it is 36. So, here we have node 5 and node 7, any one of them we can take because both are having the same cost. So, what we took? We took the node 5 and it is not there in the group 3. Now what we have to do? We have to find the neighbors of node 5. Neighbors of node 5 is node 8 and node 6. So, how can I find the cost of node 8 and node 6? Basically till 5 actually, till 5 my cost is basically 10, 10, 11. Now if I go to the node 6, I will add this 2 here, it will be 12 and 11 plus 8 becomes 19. So, if you can see here node 6 is 12, 19. Similarly for node 8, I have 10 plus 2, 10 plus 2 is 12 and 11 plus 8 basically it is 19.

So, if you can see here for node 8 which is the target node, the cost is 12, 19. Now if I look into this complete group 2 which is having the minimum cost, we will add them all. So, here it is 21. So, the cost of node 6 here it is 31 and here it is 36. So, we have to use which is having the less cost. So, we will use this one and we will remove this from the group 2. So, now we have basically 3 nodes in the group 2, 1, this is 2 and this is 3 nodes

in the group 2. We have to choose basically one of them which is having the minimum cost and here if you can see this one will be 19 plus 12, again it is 31. So, we will choose the node 7. So, node 7 will come from here to here. Node 7 will come from here to here, group 2 to group 3. Now what is happening is that from the group 7, we will do the neighbor of the group 7. The neighbor of the group 7 is basically group 8. The group 8 basically if I can see here 9, 12 is still 9, 12 is still group node 7 and the group basically the target node will have in this point 9 plus 3, this 9 plus 3 it is basically 12, 12 plus 2 is basically 14.

So, we have two paths. In this path if I come from 5 to 8 it is 12 plus 19 and if I go from in this path, this is my cost, 12, 14. So, 12, 14 is least actually. So, that path is the shortest path. So, here I have to write w8 basically here it is 12, 14, cost is basically 26, 12, 14 here it is there. So, now this 12, 14 will come to here which is the shortest. Now if you can see here we need to go basically the path which is giving me the least cost. If you can see here this 8 is 12, 14 the previous to that one is basically w which is 9, 12 and previous to that one is basically 1, 4 then final one is basically the 1. So, this is the shortest path from the source node to the target node 8. So, we need to go from your target node to the source node which path we have come to find the shortest cost. So, in this method we can find the shortest cost while length or interconnect using the Dijkstra's algorithm.

So, in this method we can find the shortest path from a source node to the destination node using the horizontal and vertical capacity of the graph. So, this is the shortest path 1, 4, 7 and 8 is the shortest path from the source node to the termination node and its cost is 12, 14.

Thank you for your attention.