First, sort the nets according to distance between end points, then find the shortest path for the net with shortest distance from starting point to terminal point by using a slightly modified Dijkstra’s algorithm.

Second, after connecting the points, store the points that the route goes through from terminal to start point in a stack, and mark the route on map. Then find the shortest path for the second shortest distance, and so on.

Finally, print the route in the route stack of each net according to the programming assignment description document.

Modified Dijkstra’s algorithm:

In conventional Dijkstra algorithm, it needs to maintain the distance from source to each vertex, but there are up to 10000 vertices in a graph for a 100\*100 map, which is so hard to maintain and costs a lot of space. That is why I decided to modify it by mixing some BFS concepts. First, we need a list for those vertices that were discovered but had not confirmed loss from source. Then is the path finding part. Start from source point, add the vertices beside in the unconfirm list like BFS and record the loss from source to each vertex. Then move to the vertex with minimum loss value in the unconfirm list, mark it as confirmed and remove it from list. Update the loss value like Dijkstra to other unconfirm vertex if possible and add new vertex to unconfirm list with recording estimate loss value. Keep do it with minimum-loss vertex in list until the loss value of end point is confirmed, and we will get the total loss of the net and the route if every loss updating step is properly recorded.