VLSI CAD Project

Shortest Path using Floyd Warshall Algorithm and Prim's Minimum Spanning Tree

Pranav Poduval-150110007 Aditya Mittal(150070060)

Introduction

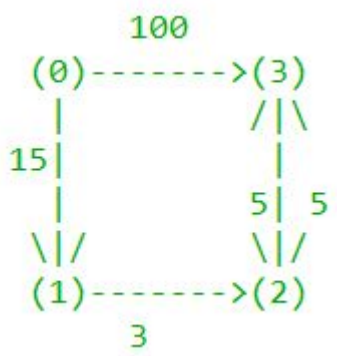
Due to the natural interpretation of a circuit as a graph model, a lot of research has been done on recently applying graph-theoretic algorithms to VLSI layout problems, so naturally when trying to optimal routing shortest path algorithms used in graph theory become directly applicable here. The representation of a graph as a spanning tree is also very important as it can be also be used in the routing of VLSI circuits.

Floyd Warshall Algorithm

The goal of Floyd Warhall Algorithm is to find the shortest path between every node in the graph, to do so the graph we pick a kth vertex and assume it is the intermediate between the i and jth vertex and update if only the path i to k and then from k to j is shorter than the currently stored i to j vertex. So for every pair (i,j) we need to explore {0,1,2,...,k-1} vertex.
Example - Consider the graph

```
graph = [[0,15,np.inf,100],
         [np.inf,0,3,np.inf],
         [np.inf, np.inf, 0,  5],
         [np.inf, np.inf,5 , 0] ]
```

Which is essentially nothing but -



The updation of the distance matrix with eack k is as follows-

```
The graph in 0th step
[[  0.  15.  inf 100.]
 [ inf   0.   3.  inf]
 [ inf  inf   0.   5.]
 [ inf  inf   5.   0.]]
The graph in 1th step
[[  0.  15.  18. 100.]
 [ inf   0.   3.  inf]
 [ inf  inf   0.   5.]
 [ inf  inf   5.   0.]]
The graph in 2th step
[[ 0. 15. 18. 23.]
 [inf  0.  3.  8.]
 [inf inf  0.  5.]
 [inf inf  5.  0.]]
The graph in 3th step
[[ 0. 15. 18. 23.]
 [inf  0.  3.  8.]
 [inf inf  0.  5.]
 [inf inf  5.  0.]]
```

The final graph (i,j) element corresponding to the shortest distance between the ith and jth vertex in a directed graph.

Time Complexity- O(|V|^3)

Prim's Minimum Spanning Tree Algorithm

A spanning tree of that graph is a subgraph that is a tree and connects all the vertices together. Minimum spanning tree(**minimum weight of edges**) is of interest to us when trying to optimize the cost of circuits

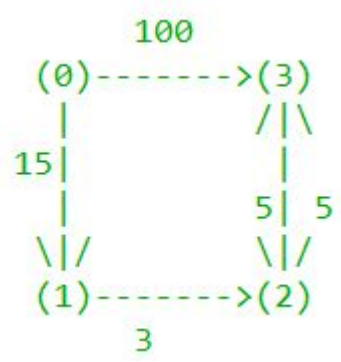
We use a greedy approach by dividing vertices into two sets one that is included in the MST and the other its complement.

At every step, it considers all the edges that connect the two sets, and picks the minimum weight edge from these edges. After picking the edge, it moves the other endpoint of the edge to the set containing MST.

So the algorithm goes as follows-
1) Create a set mstSet that keeps track of vertices already included in your Spanning Tree.
2) Assign a key value to the vertex, zero for the first and inf for the rest.

3) Pick a vertex u which is not there in mstSet and has minimum key value and include it into the set.
4)Update key value of all adjacent vertices of u. To update the key values, iterate through all adjacent vertices. For every adjacent vertex v, if weight of edge u-v is less than the previous key value of v, update the key value as weight of u-v
5)Terminate if and when we have V vertices in our set

So for the same example



The output of the Prims algorithm is -

```
Edge    Weight
0 - 1    15.0
1 - 2    3.0
2 - 3    5.0
```

Note we are considering the graph as **undirected** here.

Code Link- https://github.com/Pran97/Project