|  |  |
| --- | --- |
| Console Output showing both adjacency matrix along with minimum spanning tree output | Adjacency Matrix Graph Representation |
| Minimum Spanning Tree |

Vertex 0 was chosen as the “root.” In this case Vertex 0 would refer to 1 in the picture.

A vector was used to keep track of which nodes as visited. For now only Vertex 1 has been visited since we started there. Another vector keeps tracks of distances/weights. The root node has a weight of zero to itself. All other vertices in the distance vector have been initialized to INT\_MAX indicating that the distance is infinity, AKA there are no edges.

Next, Prim’s algorithm checks all neighboring vertices of vertex 1 for their respective distances and considers if they’ve been visited or not. In this case it would be vertices 2 & 3. These neighbors have not been visited yet and their weights are taken into consideration. They are then marked as visited and their indexed. Now it checks the neighbors of 2 & 3, which are 4 & 5.

The weights of 4 & 5 have been taken into consideration and 5 is calculated to be the more efficient path than 2 -> 3 -> 4 because 3 + 4 > 5. It marks 4 & 5 as visited and then checks it’s neighbor, which is the last vertex, 6. The edges are then indexed and printed out into the screen along with their weights. The minimum spanning tree is created.