**Weighted Graph**

*Solution for Weight Graph:*

class WeightedGraph {

constructor() {

this.adjacentList = {};

}

addVertex(vertex) {

if (!this.adjacentList[vertex]) {

this.adjacentList[vertex] = [];

}

}

addEdge(v1, v2, weight) {

this.adjacentList[v1].push({node: v2, weight: weight});

this.adjacentList[v2].push({node: v1, weight: weight});

}

}

**Dijkstra’s Algorithm**

*The Approach:*

* Everytime we look to visit a new node, we pick the node with the smallest know distance to visit first
* Once we’ve moved to the node we’re going to visit, we look at each of its neighbors
* For each neighboring node, we calculate the distance by summing the total edges that lead to the node we’re checking from the starting node
* If the new total distance to a node is less than the previous total, we store the new shorter distance for that node

*Pseudocode for Dijkstra’s Algorithm:*

* This function should accept a starting and ending vertex
* Create an object (we’ll call it distance) and set each key to be every vertex in the adjacent list with a value of infinity, except for the starting vertex which should have a value of 0
* After setting a value in the distances object, add each vertex with a priority of infinity to the priority queue, except the starting vertex, which should have a priority of 0 because that’s where we begin
* Create another object called previous and set each key to be every vertex in the adjacent list with a value of null
* Start looping as long as there is anything in the priority queue
  + Dequeue a vertex from the priority queue
  + If that vertex is the same as the ending vertex – we are done
  + Otherwise loop through each value in the adjacent list at that vertex
    - Calculate the distance to that vertex from the starting vertex
    - If the distance is less than what is currently stored in our distances object
      * Update the distance object with new lower distance
      * Update the previous object to contain that vertex
      * Enqueue the vertex with the total distance from the start node