Class 22 Homework on Dijkstra’s Algorithm

1. **(10 points)** Simulate Dijkstra’s algorithm on the following graph. (Assume 2 is the start vertex.) Assume the vertices are numbered 1 to 5. The edges are:

   1: (2,2), (3,2), (4,1), (5,7)
   2: (1,3), (3,1), (4,5)
   3: (1,4), (4,8)
   4: (5,2)
   5: (3,9)

Here (5, 7) means that vertex 1 has an edge to vertex 5 with a weight of 7. Assuming that v0 in Dijkstra’s algorithm is vertex 2, show the values of *known*, *distance* and *path* for each vertex.

**Solution:** Here are the relevant values at the end of the algorithm:

<table>
<thead>
<tr>
<th>Vertex</th>
<th>known</th>
<th>distance</th>
<th>path</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>true</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>true</td>
<td>0</td>
<td>—</td>
</tr>
<tr>
<td>3</td>
<td>true</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>true</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>true</td>
<td>6</td>
<td>4</td>
</tr>
</tbody>
</table>

Class 22 Homework Problem

1. Problem 9.10 on page 380.

**Solution:**

(a) Keep an additional member variable, `num_paths`, in the vertex class. Initialize this variable to 1 at each vertex. Whenever a new shortest path to a vertex is found, say from v to w, then assign

   ```cpp
   w->num_paths = v->num_paths;
   ```

In addition, add a condition to check if an equal distance path has been found. In this case, assign

   ```cpp
   w->num_paths += v->num_paths;
   ```

Overall, the inner for loop in Dijkstra’s algorithm becomes:
for each w adjacent to v {
  if( !w->known ) {
    wgt = weight of edge from v to w;
    if ( v->distance + wgt < w->distance )
    {
      // Update w
      decrease( w->dist to be v.dist + wgt ); /* 2 */
      w->path = v;
      w->num_paths = v->num_paths;
    }
    else if ( v->distance + wgt == w->distance )
    w->num_paths += v->num_paths;
  }
}

Note: you didn’t need to write out the code as above, it just makes my answer clearer.

(b) Add a member variable edges_on_path, initialized to infinity. Modify the inner for loop of Dijkstra’s algorithm to work as follows:

for each w adjacent to v {
  if( !w->known ) {
    wgt = weight of edge from v to w;
    if ( v->distance + wgt < w->distance ||
      ( v->distance + wgt == w->distance
        && v->edges_on_path+1 < w->edges_on_path ) )
    {
      // Update w
      decrease( w->dist to be v.dist + wgt ); /* 2 */
      w->path = v;
      w->edges_on_path = v->edges_on_path + 1;
    }
  }
}