Dijkstra's Algorithm (Single Source Shortest Path)

Dijkstra's Algorithm Overview:
- The overall logic is the same as Prim's Algorithm
- We will modify the code in only two places – both involving the update to the distance metric.
- The result is a directed acyclic graph or DAG

Pseudocode for Dijkstra's SSSP Algorithm

```plaintext
DijkstraSSSP(G, s):
Input: G, Graph;
        s, vertex in G, starting vertex of algorithm
Output: T, DAG w/ shortest paths (and distances) to s

foreach (Vertex v : G):
    d[v] = +inf
    p[v] = NULL
    d[s] = 0
PriorityQueue Q // min distance, defined by d[v]
Q.buildHeap(G.vertices())
Graph T // "labeled set"
repeat n times:
    Vertex m = Q.removeMin()
    T.add(m)
    foreach (Vertex v : neighbors of m not in T):
        if d[u] + cost(u, v) < d[v]:
            d[v] = d[u] + cost(u, v)
            p[v] = m
return T
```

Dijkstra: One heavy-weight edge vs. faster light-weight edges?

Dijkstra: One medium-weight edge vs. many light-weight edges?

Dijkstra: Undirected graphs?
Dijkstra: What if we have a negative-weight cycle?

Dijkstra: What if we have a minimum-weight edge, without having a negative-weight cycle?

...what assumption does Dijkstra’s algorithm make?

Dijkstra: What is the running time?

Landmark Path Problem: My favorite graph problem!

<table>
<thead>
<tr>
<th>CS 225 – Things To Be Doing:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Programing Exam C ongoing!</td>
</tr>
<tr>
<td>2. MP7 – due Tuesday!</td>
</tr>
<tr>
<td>3. lab_flow due Sunday</td>
</tr>
<tr>
<td>4. Daily POTDs are ongoing for +1 point /problem</td>
</tr>
</tbody>
</table>