CS 225
Data Structures

December 7 – Dijkstra’s Algorithm Analysis
Wade Fagen-Ulmschneider
Dijkstra's Algorithm (SSSP)

```plaintext
DijkstraSSSP(G, 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 u = Q.removeMin()
    T.add(u)
    foreach (Vertex v : neighbors of u not in T):
      if cost(u, v) + d[u] < d[v]:
        d[v] = cost(u, v) + d[u]
        p[v] = u
```

Diagram of a graph with nodes A, B, C, D, E, F, G, H and edges labeled with distances. The algorithm is visualized with the addition of labeled sets and the priority queue operations.
Dijkstra’s Algorithm (SSSP)

Dijkstra gives us the shortest path from our path (single source) to **every** connected vertex!
Q: How does Dijkstra handle a single heavy-weight path vs. many light-weight paths?
Dijkstra’s Algorithm (SSSP)

Q: How does Dijkstra handle a single heavy-weight path vs. many light-weight paths?
Dijkstra’s Algorithm (SSSP)

Q: How does Dijkstra handle a single heavy-weight path vs. many light-weight paths?
Q: How does Dijkstra handle undirected graphs?
Dijkstra’s Algorithm (SSSP)

**Q:** How does Dijkstra handle negative weight cycles?
Q: How does Dijkstra handle negative weight edges, without a negative weight cycle?
Dijkstra’s Algorithm (SSSP)

What is Dijkstra’s running time?

```
DijkstraSSSP(G, 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 u = Q.removeMin()
      T.add(u)
      foreach (Vertex v : neighbors of u not in T):
         if \text{cost}(u, v) + d[u] < d[v]:
            d[v] = \text{cost}(u, v) + d[u]
            p[v] = m

   return T
```
Landmark Path Problem

Suppose you want to travel from A to G.

Q1: What is the shortest path from A to G?
Landmark Path Problem

Suppose you want to travel from A to G.

Q2: What is the fastest algorithm to use to find the shortest path?
Landmark Path Problem

In your journey between A and G, you also want to visit the landmark L.

Q3: What is the shortest path from A to G that visits L?
Landmark Path Problem

In your journey between A and G, you also want to visit the landmark L.

Q4: What is the fastest algorithm to find this path?
Q5: What are the specific call(s) to this algorithm?
Thank you for an amazing semester!