CS 225
Data Structures

April 7 – Dijkstra’s Algorithm
G Carl Evans
Shortest Path
Dijkstra’s Algorithm (SSSP)

```
PrimMST(G, s):
6   foreach (Vertex v : G):
7       d[v] = +inf
8       p[v] = NULL
9       d[s] = 0
10
11   PriorityQueue Q // min distance, defined by d[v]
12   Q.buildHeap(G.vertices())
13
14   repeat n times:
15       Vertex u = Q.removeMin()
16       foreach (Vertex v : neighbors of u not in T):
17           if cost(u, v) < d[v]:
18               d[v] = cost(u, v)
19               p[v] = u
```
Dijkstra’s Algorithm (SSSP)

DijkstraSSSP(G, s):
6    foreach (Vertex v : G):
7        d[v] = +inf
8        p[v] = NULL
9        d[s] = 0
10
11    PriorityQueue Q // min distance, defined by d[v]
12    Q.buildHeap(G.vertices())
13    Graph T        // "labeled set"
14
15    repeat n times:
16        Vertex u = Q.removeMin()
17        T.add(u)
18        foreach (Vertex v : neighbors of u not in T):
19            if _______________ < d[v]:
20                d[v] = _______________
21                p[v] = u
Dijkstra's Algorithm (SSSP)

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())
    repeat n times:
        Vertex u = Q.removeMin()
        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
Dijkstra’s Algorithm (SSSP)

Dijkstra gives us the shortest path from our path (single source) to **every** connected vertex!
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?
Dijkstra’s Algorithm (SSSP)

Q: How does Dijkstra handle undirected graphs?
Dijkstra’s Algorithm (SSSP)

What is Dijkstra’s running time?

```java
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())
  repeat n times:
    Vertex u = Q.removeMin()
    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
  return T
```

What is Dijkstra’s running time?
Dijkstra’s Algorithm (SSSP)

Q: How does Dijkstra handle negative weight edges?
Modified 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 until Q.empty() times:
    Vertex u = Q.removeMin()
    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
        Q.push(v)

  return T
```

Dijkstra’s Algorithm (SSSP)

Q: How does Dijkstra handle negative weight cycles?
Dijkstra’s Algorithm (SSSP)

Q: How does Dijkstra handle negative weight cycles?
Dijkstra’s Algorithm (SSSP)

Q: How does Dijkstra handle negative weight cycles?

Shortest Path (A → E): $A \rightarrow F \rightarrow E \rightarrow (C \rightarrow H \rightarrow G \rightarrow E)^*$

Length: 12  Length: -5 (repeatable)