CS 225
Data Structures

April 26 – Dijkstra’s Algorithm Analysis
Wade Fagen-Ulmschneider, Craig Zilles
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      PriorityQueue Q // min distance, defined by d[v]
11      Q.buildHeap(G.vertices())
12      Graph T         // "labeled set"
13      repeat n times:
14          Vertex u = Q.removeMin()
15          T.add(u)
16          foreach (Vertex v : neighbors of u not in T):
17              if cost(u, v) + d[u] < d[v]:
18                  d[v] = cost(u, v) + d[u]
19                  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 a single heavy-weight path vs. many light-weight paths?
Dijkstra’s Algorithm (SSSP)

Q: How does Dijkstra handle undirected graphs?
Final Exam Review

• Practice Final Theory Exams Available Now
  • Just like Theory Exam C, infinite exams with random questions
  • Over 500 questions so far – lots of practice, lots of variations, lots of ways to ensure you know DS!

• TA Review Session IN LECTURE on Wednesday
  • Final lecture is Monday, April 29
Dijkstra’s Algorithm (SSSP)

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

**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):
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 cost(u, v) + d[u] < d[v]:
20                d[v] = cost(u, v) + d[u]
21                p[v] = m
22
23    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!