April 13 – Dijkstra’s Algorithm Analysis
G Carl Evans
If you’re interested in working on technical interview prep in a low-stress environment, join WCS x WECE on Mondays from 3-4:30pm in Siebel 1109 for our LeetCode Socials (starting next Monday, 4/18).

Our LeetCode Socials are open to anyone interested in learning or reviewing data structures in an informal setting! For more information on our LeetCode Socials, please check out: https://go.illinoiswcs.org/tech-prep

If you’re interested in participating, please register here: https://go.illinoiswcs.org/tech-prep-registration
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    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(v, m) < d[v]:
18                d[v] = cost(v, m)
19                p[v] = m
Dijkstra's Algorithm (SSSP)

```java
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] = m
```
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
14       repeat n times:
15          Vertex u = Q.removeMin()
16          T.add(u)
17          foreach (Vertex v : neighbors of u not in T):
18             if \( \text{cost}(u, v) + d[u] < d[v] \):
19                 d[v] = \( \text{cost}(u, v) + d[u] \)
20                 p[v] = u
21
Dijkstra’s Algorithm (SSSP)

Dijkstra gives us the shortest path from our path (single source) to **every** connected vertex!

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>p:</td>
<td>--</td>
<td>A</td>
<td>B</td>
<td>B</td>
<td>F</td>
<td>A</td>
<td>F</td>
<td>C</td>
</tr>
<tr>
<td>d:</td>
<td>0</td>
<td>10</td>
<td>17</td>
<td>15</td>
<td>12</td>
<td>7</td>
<td>11</td>
<td>21</td>
</tr>
</tbody>
</table>
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?
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)

**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 → F → E → (C → H → G → E)*

Length: 12 Length: -5 (repeatable)
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] = u
22
23    return T
```
Suppose I have a new heap:

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

What’s the updated running time?

<table>
<thead>
<tr>
<th></th>
<th>Binary Heap</th>
<th>Fibonacci Heap</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remove Min</td>
<td>O(lg(n))</td>
<td>O(lg(n))</td>
</tr>
<tr>
<td>Decrease Key</td>
<td>O(lg(n))</td>
<td>O(1)*</td>
</tr>
</tbody>
</table>
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?