Shortest paths: Dijkstra

Topics for this week:
- weighted graphs
- weighted shortest path algorithm: Dijkstra
- priority queue data structure

Announcements/discussion:
- project part 3 due tomorrow
- read the QuickSort notes - we’ll come back to them
Problem: Distance between locations

We have a map with specified length for every road. What is the shortest distance from city A to city D?

For example:

A - F - C - D

the shortest distance out of all possible paths from A to D
Problem: Distance between locations

Let’s look at previous approaches – do any of them work?

- DFS (depth-first search) – does NOT work, we could e.g. get the path A-B-C-D
- BFS (breadth-first search) – does NOT work, we get the path A-E-D (BFS minimizes #edges on the path, not the overall distance)
- backtracking / enumeration
  - try all paths – works but slow!
- a greedy approach:
  - always follow the shortest outgoing road
    A-B-F-E-D – not the shortest path
Graph definitions

We need to expand our graph definition: now we have distances for every direct connection between locations.

Let’s review the original terminology:

- nodes/vertices: $A, B, C, D, E, F$ (one vertex)
- edges: $(E, F)$ etc.

Here we also have:

Weights - on the edges

Note: directed/undirected
Dijkstra's algorithm

- a different greedy approach

- idea: keep temporary distances from the initial vertex to every other vertex

Diagram:

1. Start with 0 distance to the initial vertex, and ∞ to everybody else.
2. Update the non-finalized smallest current distance to the neighbors.
3. Finalize the 3rd smallest current distance.
4. Finalize the 4th smallest current distance.
5. Finalize the 5th smallest current distance.

Path:

Follow the updates backwards:

D was updated by C
C was updated by F
F was updated by C
A was updated by F

Finalized distance:

Finalized 1st

Finalized 2nd

Finalized 3rd

Finalized 4th
Dijkstra's algorithm

Pseudo code: 
- assume initV is the initial vertex

for every vertex:
  set its distance to ∞ and set its parent to None
  set the initV's distance to 0
let NonFinal be the set of all vertices
while NonFinal still contains vertices:
  let v be the vertex in NonFinal with the smallest distance
  remove v from NonFinal (it is now finalized)
  for every neighbor u of v:
    if u's distance is > v's distance + weight of the edge (v,u):
      update u's distance to v's dist + weight(v,u)
      set u's parent to v
return the distances to all vertices + parents

Running time:
- Reconstructing the path:
  let current = final V
  path = []
  while current's parent is not None:
    append current to the path
    current = current's parent

<table>
<thead>
<tr>
<th>First 2 lines</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>0</td>
<td>∞</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>B</td>
<td>7</td>
<td>0</td>
<td>8</td>
<td>0</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>C</td>
<td>22</td>
<td>8</td>
<td>8</td>
<td>0</td>
<td>14</td>
<td>9</td>
</tr>
<tr>
<td>D</td>
<td>15</td>
<td>8</td>
<td>8</td>
<td>14</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>E</td>
<td>19</td>
<td>8</td>
<td>11</td>
<td>11</td>
<td>9</td>
<td>9</td>
</tr>
</tbody>
</table>
Dijkstra's algorithm

Pseudo code: Assume `initV` is the initial vertex.

for every vertex:
    set its distance to \( \infty \) and its path to \([\]\).
set the `initV`'s distance to 0 and its path to \([initV]\).
let `NonFinal` be the set of all vertices.
while `NonFinal` still contains vertices:
    let \( v \) be the vertex in `NonFinal` with the smallest distance.
    remove \( v \) from `NonFinal` (it is now finalized).
    for every neighbor \( u \) of \( v \):
        if \( u \)'s distance is > \( v \)'s distance + weight of the edge \((v, u)\):
            update \( u \)'s distance to \( v \)'s distance + weight of \((v, u)\).
            set \( u \)'s path to \( v \)'s path, with \( u \) appended at the end.
return the distances to all vertices and the paths.

Green: Figuring out the paths Bill's way (not using parents but keeping the full path for every vertex)
Dijkstra's algorithm

Pseudo code:  

- assume initV is the initial vertex

for every vertex:
  set its distance to \( \infty \)
  set the initV's distance to 0

let NonFinal be the set of all vertices

while NonFinal still contains vertices:
  let \( v \) be the vertex in NonFinal with the smallest distance
  remove \( v \) from NonFinal (it is now finalized)
  for every neighbor \( u \) of \( v \):
    if \( u \)'s distance is > \( v \)'s distance + weight of the edge \((v,u)\):
      update \( u \)'s distance to \( v \)'s dist + weight \((v,u)\)

return the distances to all vertices

Running time:

\[ O(n^2) \]

depending on the data structures, we'll discuss this more in the lab

\[ O(n) + O(n) + \]

\[ n \text{ times } \text{extractMin from NonFinal} \]

\[ 2 \text{ times } \text{update value} \]

we are updating once for every edge:
\[ m \text{ times } \text{update value} \]

we get \( O(n^2) \) overall
Priority queue - supports operations:
- insert
- extractMin
- updateCost

Representing a weighted graph: