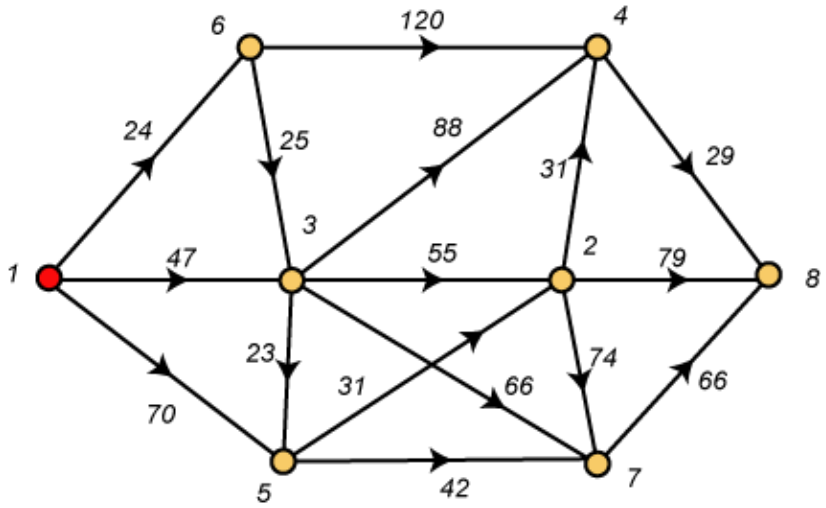
November 14, 2017

19 – Shortest Paths Algorithms

William T. Trotter
trotter@math.gatech.edu

Finding Shortest Paths



Problem Given a connected digraph G with non-negative weights on the edges and a root vertex r, find for each vertex x, a directed path $P(x)$ from r to x so that the sum of the weights on the edges in $P(x)$ is as small as possible.

Dijkstra's Algorithm

1. At each step, and each vertex x, keep track of a "distance" $d(x)$ and a directed path $P(x)$ from root to vertex x of length $d(x)$.
2. Scan first from the root and take initial paths $P(r,x) = (r, x)$ with $d(x) = w(rx)$ when rx is an edge, and $d(x) = \infty$ when rx is not an edge.
3. Vertices are either "permanent" or "temporary". At first, the root r is the only permanent vertex. For each permanent vertex x, the current value of $d(x)$ is the length of a shortest path from r to x.

Dijkstra's Algorithm - The Inductive Step

1. Of all temporary vertices, choose one, say x, whose "distance" to the root is minimum. Mark it permanent.
2. For each temporary vertex y distinct from x, set

   $$d(y) = \min\{d(y), d(x)+w(xy)\}$$

3. If this assignment lowers the value of $d(y)$, set $P(y)$ to be the path obtained by appending y to the end of $P(x)$

Initialization

Remark Initially, the paths are just the edges (if they exist) joining each vertex to the root. If there is no edge, take the length of the edge to be infinite. The sequence of permanent vertices is the trivial sequence (1).

1. d(1) = 0; P(1) = (1)
2. d(2) = ∞; P(2) = (1,2)
3. d(3) = 47; P(3) = (1,3)
4. d(4) = ∞; P(4) = (1,4)
5. d(5) = 70; P(5) = (1,5)
6. d(6) = 24; P(6) = (1,6)
7. d(7) = ∞; P(7) = (1,7)
8. d(8) = ∞; P(8) = (1,8)

Step 1

Among the temporary vertices, choose one closest to the root. This is vertex 6. Make vertex 6 permanent. Scan edges from 6 and see edge (6,3) of weight 25 and edge (6,4) of weight 120. This results in improvements for vertex 4 but not for 3.

1. d(1) = 0; P(1) = (1)
2. d(2) = ∞; P(2) = (1,2)
3. d(3) = 47; P(3) = (1,3)
4. d(4) = ∞; P(4) = (1,4) New d(4) = 144; P(4) = (1,6,4)
5. d(5) = 70; P(5) = (1,5)
6. d(6) = 24; P(6) = (1,6)
7. d(7) = ∞; P(7) = (1,7)
8. d(8) = ∞; P(8) = (1,8)

Step 2

Among the temporary vertices, choose one closest to the root. This is vertex 3. Make vertex 3 permanent. Scan edges from 3 and see edge (3,2) of weight 55, edge (3,4) of weight 88, edge (3,5) of weight 23 and edge (3,7) of weight 66. These edges result in improvements for vertices 2, 4 and 7 but not 5 (where there is a tie).

1. d(1) = 0; P(1) = (1)
2. d(2) = ∞; P(2) = (1,2) New d(2) = 102; P(2) = (1,3,2)
3. d(3) = 47; P(3) = (1,3)
4. d(4) = 144; P(4) = (1,6,4) New d(4) = 135; P(4) = (1,3,4)
5. d(5) = 70; P(5) = (1,5)
6. d(6) = 24; P(6) = (1,6)
7. d(7) = ∞; P(7) = (1,7) New d(7) = 113; P(7) = (1,3,7)
8. d(8) = ∞; P(8) = (1,8)

Step 3

Among the temporary vertices, choose one closest to the root. This is vertex 5. Make vertex 5 permanent. Scan edges from 5 and see edge (5,2) of weight 32 and edge (5,7) of weight 42. These edges result in improvements for vertices 2 and 7.

1. d(1) = 0; P(1) = (1)
2. d(2) = 102; P(2) = (1,3,2) New d(2) = 101; P(2) = (1,5,2)
3. d(3) = 47; P(3) = (1,3)
4. d(4) = 135; P(4) = (1,3,4)
5. d(5) = 70; P(5) = (1,5)
6. d(6) = 24; P(6) = (1,6)
7. d(7) = 113; P(7) = (1,3,7) New d(7) = 112; P(7) = (1,5,7)
8. d(8) = ∞; P(8) = (1,8)

Step 4

Among the temporary vertices, choose one closest to the root. This is vertex 2. Make vertex 2 permanent. Scan edges from 2 and see edge (2,4) of weight 31, edge (2,7) of weight 74 and edge (2,8) of weight 79. These edges result in improvements for vertices 4 and 8, but not vertex 7.

1. d(1) = 0; P(1) = (1)
2. d(2) = 101; P(2) = (1,5,2)
3. d(3) = 47; P(3) = (1,3)
4. d(4) = 135; P(4) = (1,3,4) New d(4) = 132; P(4) = (1,5,2,4)
5. d(5) = 70; P(5) = (1,5)
6. d(6) = 24; P(6) = (1,6)
7. d(7) = 112; P(7) = (1,5,7)
8. d(8) = ∞; P(8) = (1,8) New d(8) = 180; P(8) = (1,5,2,8)

Step 5

Among the temporary vertices, choose one closest to the root. This is vertex 7. Make vertex 7 permanent. Scan edges from 7 and see edge (7,8) of weight 66. This edge results in an improvement for vertex 8.

1. d(1) = 0; P(1) = (1)
2. d(2) = 101; P(2) = (1,5,2)
3. d(3) = 47; P(3) = (1,3)
4. d(4) = 132; P(4) = (1,5,2,4)
5. d(5) = 70; P(5) = (1,5)
6. d(6) = 24; P(6) = (1,6)
7. d(7) = 112; P(7) = (1,5,7)
8. d(8) = 180; P(8) = (1,5,2,8) New d(8) = 178; P(8) = (1,5,7,8)

Step 6

Among the temporary vertices, choose one closest to the root. This is vertex 4. Make vertex 4 permanent. Scan edges from 4 and see edge (4,8) of weight 29. This edge results in an improvement for vertex 8.

1. d(1) = 0; P(1) = (1)
2. d(2) = 101; P(2) = (1,5,2)
3. d(3) = 47; P(3) = (1,3)
4. d(4) = 132; P(4) = (1,5,2,4)
5. d(5) = 70; P(5) = (1,5)
6. d(6) = 24; P(6) = (1,6)
7. d(7) = 112; P(7) = (1,5,7)
8. d(8) = 178; P(8) = (1,5,7,8) New d(8) = 161; P(8) = (1,5,2,4,8)

Step 7

The last temporary vertex becomes permanent. There are no edges to scan. DONE!!

1. d(1) = 0; P(1) = (1)
2. d(2) = 101; P(2) = (1,5,2)
3. d(3) = 47; P(3) = (1,3)
4. d(4) = 132; P(4) = (1,5,2,4)
5. d(5) = 70; P(5) = (1,5)
6. d(6) = 24; P(6) = (1,6)
7. d(7) = 112; P(7) = (1,5,7)
8. d(8) = 161; P(8) = (1,5,2,4,8)

The Correctness of the Algorithm (1)

Proof A very important first observation is that Dijkstra's Algorithm determines a permutation

$$\sigma = (x_1, x_2, x_3, x_4, \ldots, x_n)$$

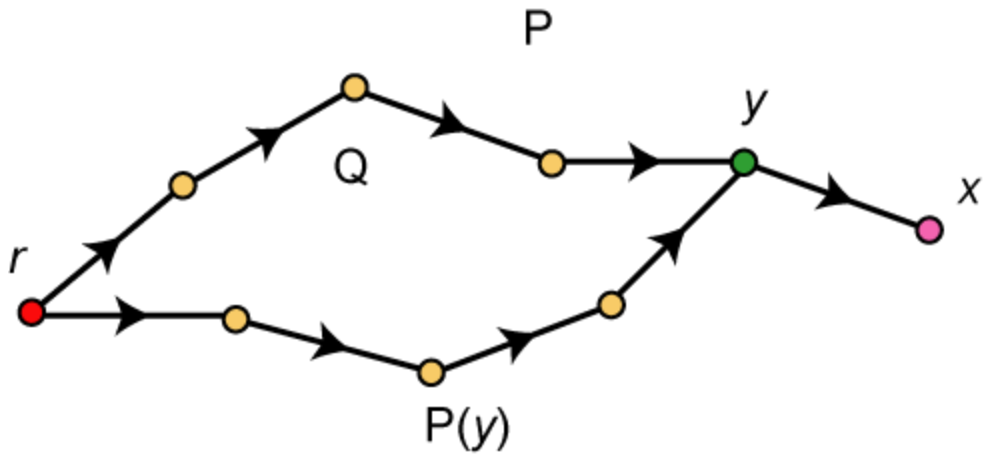
of the vertex set of the digraph according to the order in which the vertices are marked permanent. Of course, x_1 is the root r. For each vertex x_i, the algorithm has determined a path $P(x_i)$ from r to x_i having length $d(x_i)$. At this stage, it is not clear that $d(x_i)$ is really the shortest distance from r to x_i. However, we do know that these values are increasing, i.e.,

$$d(x_1) \leq d(x_2) \leq d(x_3) \leq d(x_4) \leq \ldots \leq d(x_n)$$

The Correctness of the Algorithm (2)

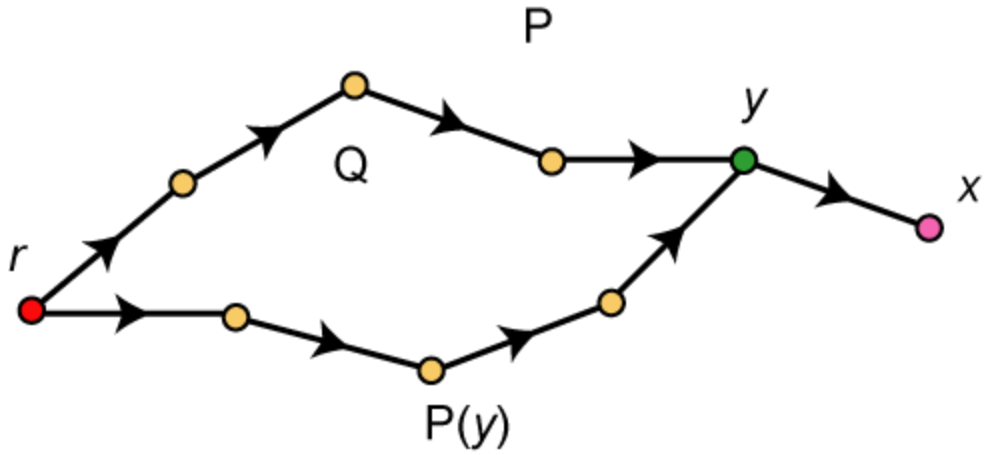
1. We show that for each vertex x, the length d(x) of the path P(x) is the shortest distance from r to x. The argument proceeds by induction on the minimum number k of edges in a shortest path from r to x. Note that the claim holds for $k = 1$, since we scan the edge (r,x) at Step 1.
2. Now assume that for some positive integer k, Dijkstra's Algorithm find a shortest path from r to x whenever the minimum number of edges in such a path is at most k. Then let x be a vertex for which the minimum number of edges in a shortest path from r to x is $k+1$. Let P be such a path and let y b the point immediately before x on P.

The Correctness of the Algorithm (3)



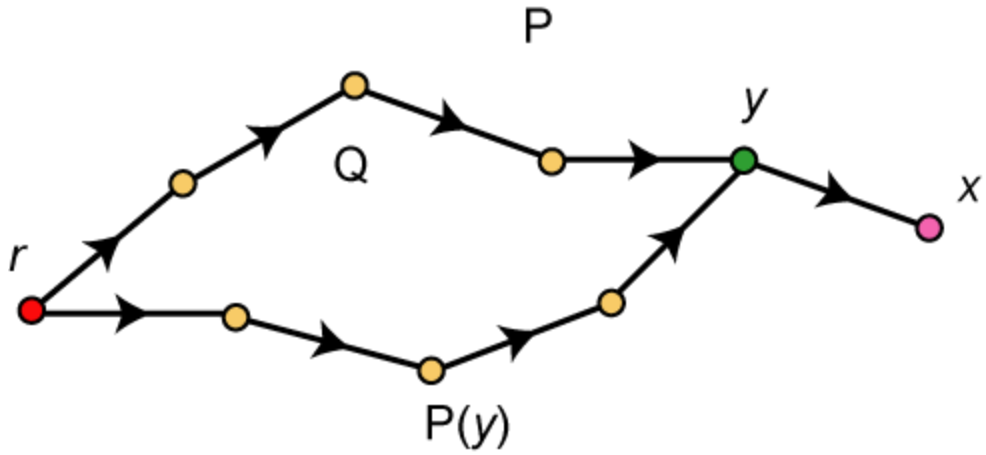
Let Q be the initial segment of P beginning at *r* and ending at *y*. Then Q is a shortest path from *r* to *y*, so the minimum number of edges in a shortest path from *r* to *y* is at most *k*. Therefore Dijkstra's algorithm finds a shortest path P(y) from *r* to *y*. Note that P(y) need not be the same as Q. However, Q and P(y) both length d(y).

The Correctness of the Algorithm (4)



The length of path P is $d(y) + w(x,y) \geq d(y)$ since all weights are non-negative. If x is marked permanent before y, then the algorithm has already found a path $P(x)$ from r to x of length $d(x) \leq d(y)$. This implies that $d(x) = d(y)$ and $w(x,y) = 0$. So the algorithm has found a shortest length path from r to x, albeit one where the last edge is "free", i.e., has weight zero.

The Correctness of the Algorithm (5)



Now suppose that y is marked permanent before x. When we scan from y, we will see the edge (x,y) having weight w(x,y). Therefore, $d(x) \leq d(y) + w(x,y)$, i.e., the algorithm will find a shortest path from r to x. This observation completes the proof.

Data Structure and Computational Issues

1. Dijkstra's Algorithm has modest space requirements since we only maintain information about the candidate optimal path.
2. If the graph has n vertices, each iteration marks a new vertex as permanent, so there are only n iterations. Also, each scan involves $O(n)$ calculations. So the running time is $O(n^2)$. In fact, the running time is essentially the same as the time it takes to read the data.