Intro to Contemporary Math

Dijkstra's Algorithm

Nicholas Nguyen
nicholas.nguyen@uky.edu

Department of Mathematics
UK

Agenda

- Dijkstra's Algorithm
  - Procedure
  - Evaluation

Distance Between Vertices

- Definition: A **distance** between two vertices is the total weight of a path between them
- Goal: Want shortest path (path with smallest total weight) which gives shortest distance between a starting vertex and destination vertex

Dijkstra's Algorithm Overview, Steps 1-2

Instead of testing entire paths, we build one.

Step 1: Mark the destination vertex as **current**. Record its **distance** as **zero**: its has distance zero from the destination vertex.

-
-

Dijkstra's Algorithm Overview, Steps 1-2

Instead of testing entire paths, we build one.

Step 1: Mark the destination vertex as **current**. Record its **distance** as **zero**: its has distance zero from the destination vertex.

Step 2: For each unvisited vertex **connected** to the current one, add these numbers:

- the **weight of the edge** from this vertex to current one
- the **distance** from the current vertex to the destination

Dijkstra's Algorithm Overview, Steps 1-2

Instead of testing entire paths, we build one.

Step 1: Mark the destination vertex as **current**. Record its **distance** as **zero**: its has distance zero from the destination vertex.

Step 2: For each unvisited vertex **connected** to the current one, add these numbers:

- the **weight of the edge** from this vertex to current one
- the **distance** from the current vertex to the destination

Step 2a: **Record the sum (distance) and the current vertex** next to this vertex's label **UNLESS** the vertex is **already recorded with a smaller distance.**

Dijkstra's Algorithm Steps 2-4

Step 2: For each unvisited vertex **connected** to the current one, add these numbers:

- the **weight of the edge** from this vertex to current one
- the **distance** from the current vertex to the destination

Step 2a: **Record the sum (distance) and the current vertex** next to this vertex's label **UNLESS** the vertex is **already recorded with a smaller distance.**

Step 3: Mark the **current** vertex as **visited**.

Step 4: Mark the **unvisited vertex** with the **smallest distance** as **current**, and repeat from step 2.

Dijkstra's Algorithm Steps 2a-4+End

Step 2a: **Record the sum (distance) and the current vertex** next to this vertex's label **UNLESS** the vertex is **already recorded with a smaller distance.**
Step 3: Mark the **current** vertex as **visited**.
Step 4: Mark the **unvisited vertex** with the **smallest distance** as **current**, and repeat from step 2.
Stop when all vertices (except the start vertex) are visited.

Dijkstra's Algorithm Evaluation

Optimal: Guaranteed to find shortest path (path of least weight)
Algorithm must have safeguards to avoid missing shortest path

- Step 4 prevents algorithm from missing shorter paths from an unvisited vertex to the destination.
- Unvisited vertices with higher number records may get rerecorded in Step 2a. Going to them first will miss shorter paths from them!

Dijkstra's Algorithm Evaluation

Efficient: Computers can use it to find a shortest path quickly

Algorithm must have ways to avoid testing bad paths

- Step 3 prevents the algorithm from going in circles.
- A vertex is visited when we know the shortest distance from it to the destination, so the algorithm knows not to try other paths from it.

Dijkstra's vs Brute Force

- Both are optimal - in theory they find the shortest path
- Dijkstra's is efficient, but
- Brute Force is not efficient because it tests every path, including bad ones (that go in circles, make U-turns, etc.)

Dijkstra's vs Brute Force

For comparison, on a graph with 25 vertices,

- Dijkstra's takes 625 calculations
- Brute Force takes 10^{25} calculations - that's 1 with 25 zeroes after it.

Next Time

- Dijkstra's Algorithm: Interpreting Output

Bibliography

- Lippman, David. Math in Society. 2nd ed. 16 November 2013. <http://www.opentextbookstore.com/mathinsociety/current2.php?chapter=GraphTheory.pdf>. Web.