sac8371@rit.edu, jkr4080@rit.edu

Dijkstra's Algorithm

(single-source shortest-path)

Salvatore Castro
Jason Roberts

Overview

Tuesday, December 2, 2:20:08 PM

sac8371@rit.edu, jkr4080@rit.edu

Introduction

Tuesday, December 2, 2:20:08 PM

- Single-source shortest-path
- Applies to weighted-directed graph
- $G = (V, E)$
- Running time lower than Bellman-Ford
- Does not run on negative weights

sac8371@rit.edu, jkr4080@rit.edu

History

Tuesday, December 2, 2:20:08 PM

- Edsger Wybe Dijkstra
- May 11, 1930 – August 6, 2002
- *Go To Statement Considered Harmful* (1968)
- "Computer science is no more about computers than astronomy is about telescopes." -E. Dijkstra

sac8371@rit.edu, jkr4080@rit.edu

Shortest Path

Tuesday, December 2, 2:20:08 PM

- Dijkstra's Algorithm
  - single source problem if all edge weights are greater than or equal to zero. Without worsening the run time, this algorithm can in fact compute the shortest paths from a given start point *s* to *all other* nodes.
- Bellman-Ford
  - single source problem if edge weights may be negative.
- A* Shortest Paths
  - a heuristic for single source shortest paths.
- Floyd-Warshall
  - solves all pairs shortest paths
- Johnson's Algorithm
  - solves all pairs shortest paths, may be faster than Floyd-Warshall on sparse graphs.

sac8371@rit.edu, jkr4080@rit.edu

Bellman-Ford

Tuesday, December 2, 2:20:08 PM

- Quick Overview
-
```
BF(G,w,s) // G = Graph, w = weight, s=source
Determine Single Source(G,s);
set Distance(s) = 0;
Predecessor(s) = nil;
for each vertex v in G other than s,
        set Distance(v) = infinity, Predecessor(v) = nil;
for i <- 1 to |V(G)| - 1 do //|V(G)| Number of vertices in the graph
        for each edge (u,v) in G do
                if Distance(v) > Distance(u) + w(u,v) then
                        set Distance(v) = Distance(u) + w(u,v), Predecessor(v) = u;
for each edge (u,r) in G do
        if Distance(r) > Distance(u) + w(u,r);
                return false; //This means that the graph contains a cycle of negative
                                        //weight and the shortest paths are not well defined
return true; //Lengths of shortest paths are in Distance array
```

sac8371@rit.edu, jkr4080@rit.edu

Applications

Tuesday, December 2, 2:20:08 PM

- Routing, Routing, and Routing
- OSPF (Open shortest path first) is a well known real world implementation used in internet routing.

sac8371@rit.edu, jkr4080@rit.edu

7

Theoretical Basis

Tuesday, December 2, 2:20:08 PM

- Assume that the function w: V x V -> [0, ∞] describes the cost w(x,y) of moving from vertex x to vertex y (non-negative cost).
- We can define the cost to be infinite for pairs of vertices that are not connected by an edge.
- The cost of a path between two vertices is the sum of costs of the edges in that path. The cost of an edge can be thought of as (a generalisation of) the distance between those two vertices.
- For a given pair of vertices s,t in V, the algorithm finds the path from s to t with lowest cost (i.e. the shortest path).

sac8371@rit.edu, jkr4080@rit.edu

8

Theoretical Basis (cont'd)

Tuesday, December 2, 2:20:08 PM

- The algorithm works by constructing a subgraph S of such that the distance of any vertex v' (in S) from s is **known** to be a minimum within G.
- Initially S is simply the single vertex s, and the distance of s from itself is known to be zero.
- Edges are added to S at each stage by (a) identifying all the edges ei = (vi1,vi2) in G-S such that vi1 is in S and vi2 is in G, and then (b) choosing the edge ej = (vj1,vj2) in G-S which gives the minimum distance of its vertex vj2 (in G) from s from all edges ei.
- The algorithm terminates either when S becomes a spanning tree of G, or when all the vertices of interest are within S.

sac8371@rit.edu, jkr4080@rit.edu

9

Pseudocode (auxilliary functions)

Tuesday, December 2, 2:20:08 PM

- Initialize-Single-Source(G,s)

```
for each vertex v in V[G]
    do d[v] := infinite
       previous[v] := 0
d[s] := 0
```

- Relax(u,v,w)

```
if d[v] > d[u] + w(u,v)
    then d[v] := d[u] + w(u,v)
         previous[v] := u
```

sac8371@rit.edu, jkr4080@rit.edu

10

Pseudocode

Tuesday, December 2, 2:20:08 PM

- Dijkstra(G,w,s)

```
Initialize-Single-Source(G,s)
S := empty set
Q := set of all vertices
while Q is not an empty set
    do u := Extract-Min(Q)
           S := S union {u}
           for each vertex v which is a neighbor of u
                  do Relax(u,v,w)
```

sac8371@rit.edu, jkr4080@rit.edu

11

Algorithm Steps

Tuesday, December 2, 2:20:08 PM

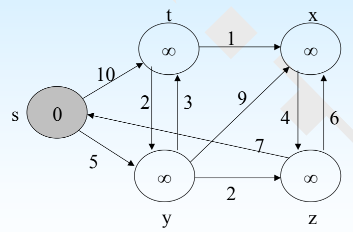
1. Make the source node a “Permanent” Node. The source node is the first working node.
2. Examine each non-permanent node adjacent to the working node. If it is not labeled, label it with the distance from the source and the name of the working node. If it is labeled, see if the cost computed using the working node is cheaper than the cost in the label; if so re-label the node as above.
3. Find the non-permanent node with the smallest label, and make it permanent. If this is the destination, then finished. Otherwise, this node is the next working node, continue from step 2.

sac8371@rit.edu, jkr4080@rit.edu

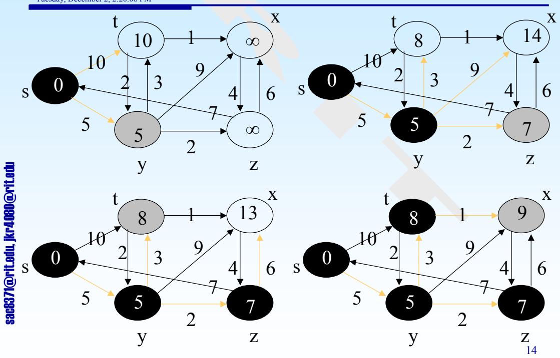
12

Example

For this example we will use ‘S’ as the start node.

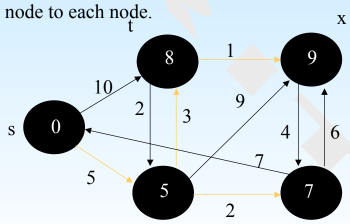


Example Continued



Final Result

This directed graph illustrates the shortest path from the source node to each node.



Here is a Java Applet Example

Complexity Analysis

- First Case: Min-Priority Queue has ordered number of vertices numbered 1 to |V|. Each INSERT and DECREASE-KEY operation is O(1). EXTRACT-Min takes O(V) time. Total: $O(V^2+E) \rightarrow O(V^2)$.
- Second Case: Min-Priority Queue is implemented as a Fibonacci Heap. The total running time for this will be O(V lgV + E). EXTRACT-Min will take O(lgV) and DECREASE-Key will take O(1).

Conclusion

- Dijksta’s algorithm has some similarity to both breadth-first search (BFS) and Prim’s Algorithm for computing minimum spanning trees.
- BFS: Similar in that the set S corresponds to the set of black vertices in the BFS just as the vertices in S have their final shortest-path weights, so do black vertices in a BFS for their correct vertices.
- Prim’s: Similar in that both algorithms use a min-priority queue to find the “lightest” vertex outside a given set, add this vertex into the set, and adjust the weights of the remaining vertices outside the set accordingly.

Questions?

?’s

References

Tuesday, December 2, 2:20:08 PM

- Introduction to Algorithms 2nd Edition T. H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein MIT, Copyright 2001 by McGraw-Hill pg. 385-393

  ISBN 0-07-013151-1
- Computer Networking James F. Kurose & Keith W. Ross, Copyright 2003 by Addison-Wesley pg. 350-353

  ISBN 0-07-013151-1
- http://carbon.cudenver.edu/~hgreenbe/sessions/dijkstra/DijkstraApplet.html

sac8371@rit.edu, jkr4080@rit.edu