CSE 326: Data Structures
Shortest Path Problems

Ben Lerner
Spring 2008

Announcements (5/28/08)

- HW7 due Friday.
- Laura will not be holding office hours Friday 5/30 or Tuesday 6/3 but will be available by email.
- Reading for this lecture: Chapter 9.

The Shortest Path Problem

Given a graph $G$, and vertices $s$ and $t$ in $G$, find the shortest path from $s$ to $t$.

Two cases: weighted and unweighted.
For a path $p = v_0, v_1, v_2 \ldots v_k$
- unweighted length of path $p = k$ (a.k.a. length)

- weighted length of path $p = \sum_{i=0}^{k-1} c_{i,i+1}$ (a.k.a. cost)

Path length equals path cost when ?

Single Source Shortest Paths (SSSP)

Given a graph $G$ and vertex $s$, find the shortest paths from $s$ to all vertices in $G$.

- Is this harder or easier than the previous problem?
Variations of SSSP

- Weighted vs. unweighted
- Directed vs undirected
- Cyclic vs. acyclic
- Positive weights only vs. negative weights allowed
- Shortest path vs. longest path
- ...

SSSP: Unweighted Version

Ideas?

```cpp
void Graph::unweighted (Vertex s){
    Queue q(NUM_VERTICES);
    Vertex v, w;
    q.enqueue(s);
    s.dist = 0;

    while (!q.isEmpty()){
        v = q.dequeue();
        for each w adjacent to v
            if (w.dist == INFINITY){
                w.dist = v.dist + 1;
                w.path = v;
                q.enqueue(w);
            }
    }
}
```

total running time: $O(|E| + |V|)$
Dijkstra's Algorithm: Idea

Adapt BFS to handle weighted graphs

Two kinds of vertices:
- Finished or known vertices
  - Shortest distance has been computed
- Unknown vertices
  - Have tentative distance

At each step:
1) Pick closest unknown vertex
2) Add it to known vertices
3) Update distances

Weighted SSSP:
All edges are not created equal

Can we calculate shortest distance to all nodes from Allen Center?
Dijkstra’s Algorithm: Pseudocode

Initialize the cost of each node to $\infty$
Initialize the cost of the source to 0

While there are unknown nodes left in the graph
    Select an unknown node $a$ with the lowest cost
    Mark $a$ as known
    For each node $b$ adjacent to $a$
        if(cost($a$) + cost($a$, $b$) < cost($b$))
            cost($b$) = cost($a$) + cost($a$, $b$)
            previous($b$) = $a$

Important Features

- Once a vertex is made known, the cost of the shortest path to that node is known
- While a vertex is still not known, another shorter path to it might still be found
- The shortest path itself can be found by following the backward pointers stored at each node
Dijkstra’s Algorithm in action

<table>
<thead>
<tr>
<th>Vertex</th>
<th>Visited?</th>
<th>Cost</th>
<th>Found by</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Y</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>&lt;=2</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>Y</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>D</td>
<td>&lt;=4</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>&lt;=12</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>&lt;=4</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>??</td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>??</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Dijkstra’s Algorithm in action

<table>
<thead>
<tr>
<th>Vertex</th>
<th>Visited?</th>
<th>Cost</th>
<th>Found by</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Y</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>Y</td>
<td>2</td>
<td>A</td>
</tr>
<tr>
<td>C</td>
<td>Y</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>D</td>
<td>&lt;=4</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>&lt;=12</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>&lt;=4</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>??</td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>&lt;=7</td>
<td>F</td>
<td></td>
</tr>
</tbody>
</table>

Dijkstra’s Algorithm in action

<table>
<thead>
<tr>
<th>Vertex</th>
<th>Visited?</th>
<th>Cost</th>
<th>Found by</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Y</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>Y</td>
<td>2</td>
<td>A</td>
</tr>
<tr>
<td>C</td>
<td>Y</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>D</td>
<td>&lt;=4</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>&lt;=12</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>&lt;=4</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>??</td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>&lt;=7</td>
<td>F</td>
<td></td>
</tr>
</tbody>
</table>

Dijkstra’s Algorithm in action

<table>
<thead>
<tr>
<th>Vertex</th>
<th>Visited?</th>
<th>Cost</th>
<th>Found by</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Y</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>Y</td>
<td>2</td>
<td>A</td>
</tr>
<tr>
<td>C</td>
<td>Y</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>D</td>
<td>&lt;=4</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>&lt;=12</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>&lt;=4</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>??</td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>&lt;=7</td>
<td>F</td>
<td></td>
</tr>
</tbody>
</table>
Dijkstra's Alg: Implementation

Initialize the cost of each node to $\infty$
Initialize the cost of the source to 0
While there are unknown nodes left in the graph
   Select the unknown node $a$ with the lowest cost
   Mark $a$ as known
   For each node $b$ adjacent to $a$
      $\text{cost}(b) = \min(\text{cost}(b), \text{cost}(a) + \text{cost}(a, b))$
      previous($b$) = $a$ (if we updated $b$'s cost)
What data structures should we use?

- priority queue
- dictionary

Running time: $O((V + E) \log V)$

Dijkstra's Algorithm: Summary

- Classic algorithm for solving SSSP in weighted graphs without negative weights
- A greedy algorithm (irrevocably makes decisions without considering future consequences)
- Intuition for correctness:
  - shortest path from source vertex to itself is 0
  - cost of going to adjacent nodes is at most edge weights
  - cheapest of these must be shortest path to that node
  - update paths for new node and continue picking cheapest path

Correctness: The Cloud Proof

Next shortest path from inside the known cloud

Better path to $V$? No!

How does Dijkstra's decide which vertex to add to the Known set next?
- If path to $V$ is shortest, path to $W$ must be at least as long
  (or else we would have picked $W$ as the next vertex)
- So the path through $W$ to $V$ cannot be any shorter!
Correctness: Inside the Cloud

Prove by induction on # of nodes in the cloud:
Initial cloud is just the source with shortest path 0
Assume: Everything inside the cloud has the correct shortest path
Inductive step: Only when we prove the shortest path to some node $v$ (which is not in the cloud) is correct, we add it to the cloud

When does Dijkstra’s algorithm not work?

The Trouble with Negative Weight Cycles

What’s the shortest path from A to E?

Problem?

Dijkstra’s vs BFS

At each step:
1) Pick closest unknown vertex
2) Add it to finished vertices
3) Update distances

$Dijkstra's$ $Algorithm$

$fringe = cost < \infty$

Some Similarities:
- known nodes
- new nodes "fringe"

$Breadth-first$ $Search$

$fringe = queue$