CSE 326: Data Structures
Shortest Path Problems

Steve Seitz
Winter 2009

Announcements (3/6/09)

- HW7 due now
- HW8 (last one!) out today
- 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*)

Single Source Shortest Paths (SSSP)

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

- How much harder is this than finding single shortest path from $s$ to $t$?

  - *no harder in worst case*
  - *slightly easier on average*
Variations of SSSP

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

Applications

- Network routing
- Driving directions
- Cheap flight tickets
- Critical paths in project management (see textbook)
- ...

SSSP: Unweighted Version

```
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.prev = v;
                q.enqueue(w);
            }
    }
}
```

Breadth-first search (BFS)

each edge examined at most once - if adjacency lists are used

each vertex enqueued at most once

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

Adapt BFS to handle weighted graphs

Two kinds of vertices:
- **Known**
  - shortest distance is already known
- **Unknown**
  - Have tentative distance

### Weighted SSSP: All edges are not created equal

Can we calculate shortest distance to all vertices from Allen Center?

- **Ben & Jerry’s**: 40
- **Delfino’s**: 25
- **U Village**: 375
- **HUB**: 350
- **Vending Machine in EE1**: 10
- **Coke Closet**: 365
- **Café Allegro**: 15
- **Neelam’s**: 70
- **285**
- **75**
- **350**
- **35**
- **15,356**
- **Parent’s Home**
Dijkstra’s Algorithm: Pseudocode

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

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

Important Features

- Once a vertex is known, the cost of the shortest path to that vertex is known
- While a vertex is still unknown, another shorter path to it might still be found
- The shortest path can found by following the previous pointers stored at each vertex

Dijkstra’s Alg: Implementation

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

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

What data structures should we use?

Running time?

\[ \text{naive: } O(\nu^2 + \nu \ell), \text{ heap: } O(\nu \log \nu + \ell \log \nu) \]
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)
- Why does it work?

Correctness: The Cloud Proof

Next shortest path from inside the known cloud

Better path to V? No!

The Known Cloud

Source

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: by argument on previous slide, we can safely add min-cost vertex to cloud

Negative Weights?

Next shortest path from inside the known cloud

Better path to V? -1000

The Known Cloud

Source

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!

When does Dijkstra’s algorithm not work?
Dijkstra for BFS

- You can use Dijkstra’s algorithm for BFS

- Is this a good idea? **No**
  
  *(more expensive)*