COMPSCI 311: Introduction to Algorithms
Lecture 8: Shortest Paths

Marius Minea
University of Massachusetts Amherst

slides credit: Dan Sheldon

Let’s Formalize the Problem

- Directed graph $G = (V, E)$ with nonnegative edge lengths $\ell(e) \geq 0$
- Define length of path $P$ consisting of edges $e_1, e_2, \ldots, e_k$ as $\ell(P) = \ell(e_1) + \ell(e_2) + \ldots + \ell(e_k)$
- Starting node $s$
- Let $d(v)$ be the length of shortest $s \rightarrow v$ path.
- Problem: Can we efficiently find $d(v)$ for all nodes $v \in V$?

Question: Why for all nodes at the same time?

Clicker Question 1
Consider a car’s GPS navigation system. What shortest paths does it compute at any given time?

A. Single sink: from every node to one node $t$.
B. Source-sink: from one node $s$ to another node $t$.
C. Single source: from one node $s$ to every other node.
D. All pairs: between all pairs of nodes.

Shortest Paths Problem
Suppose all edges have integer length. Can we use BFS to solve this problem?

Recall: nodes in layer $L_i$ are at distance $i$ from start.
Clicker Question 2

If edge lengths are integers, and $C$ is the maximum length, the running time is (choose the most restrictive):

A. $O(C + m + n)$
B. $O(n + C \cdot m)$
C. $O(m + C \cdot n)$
D. $O(C \cdot (m + n))$

Hint: Why is BFS $O(m + n)$ and not $O(m)$?

Towards an Algorithm

Notation:

- $d'(v)$ — earliest tentative arrival time so far for node $v$
- $d(v)$ — shortest distance (actual arrival time)

How to keep track of the wavefront?

- Find next arrival: node $v$ with smallest $d'(v)$
- Set shortest distance: $d(v) = d'(v)$
- Update $d'(v)$ for neighbors of $v$ if path through $v$ shorter

What data structure supports find smallest and update values? Priority queue.
Shortest Paths Problem

Dijkstra’s Algorithm

Running Time?

Proof of Correctness

Induction Proof

- **Base case**: Initially $S = \emptyset$. (1) is vacuously true. The only path satisfying (2) is the empty path, and $d'(s) = 0.$

- **Induction step**:
  - Assume the invariant is true for $|S| = k \geq 0.$
  - Let $v = \text{next node added to } S$, with $d'(v) = \min_{w \in A} d'(w)$. We claim (1) $d(v) = d'(v)$ is the shortest path.
  - By (2), $d'(v)$ is shortest path with all prior nodes in $S$. 

- **Proof**: By induction on $|S|$.
Induction Proof (2)

Consider a path with some prior node not in $S$.

Let $y \in A$ be the first such node on a path $s \leadsto y \leadsto v$.

But then $\ell(s \leadsto y \leadsto v) \geq \ell(s \leadsto y) \geq d'(y) \geq d'(v)$,

so we cannot have a shorter path.

Critical point: segment $y \leadsto v$ cannot have negative length.

Proof: Maintaining the Invariant

Show we maintain (2): for $v \in A$, $d'(v)$ is the length of the shortest $s \leadsto v$ path with all nodes in $S$ except $v$.

for all edges $(v, w)$ where $w \in A$
do
  if $d'(v) + \ell(v, w) < d'(w)$ then
    $d'(w) = d'(v) + \ell(v, w)$
  end if
end for

Adding $v$ to $S$, we get new such paths $s \leadsto w$ for all edges $(v, w)$.

Updating $d'(w) = \min(d'(w), d(v) + \ell(v, w))$ maintains invariant.

Dijkstra’s algorithm: which priority queue?

Performance. Depends on PQ: $n$ INSERT, $n$ DELETE-MIN, $\leq m$ DECREASE-KEY.

- Array implementation optimal for dense graphs.
- Binary heap much faster for sparse graphs.
- 4-way heap worth the trouble in performance-critical situations.

<table>
<thead>
<tr>
<th>priority queue</th>
<th>INSERT</th>
<th>DELETE-MIN</th>
<th>DECREASE-KEY</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>unordered array</td>
<td>$O(1)$</td>
<td>$O(n)$</td>
<td>$O(1)$</td>
<td>$O(n^2)$</td>
</tr>
<tr>
<td>binary heap</td>
<td>$O(\log n)$</td>
<td>$O(\log n)$</td>
<td>$O(\log n)$</td>
<td>$O(m \log n)$</td>
</tr>
<tr>
<td>d-way heap (johnson 1975)</td>
<td>$O(d \log n)$</td>
<td>$O(d \log n)$</td>
<td>$O(d \log n)$</td>
<td>$O(m \log n)$</td>
</tr>
<tr>
<td>Fibonacci heap</td>
<td>$O(1)$</td>
<td>$O(\log n)$</td>
<td>$O(1)$</td>
<td>$O(n^2)$</td>
</tr>
<tr>
<td>(Fredman-Tarjan 1984)</td>
<td>$O(1)$</td>
<td>$O(\log n)$</td>
<td>$O(1)$</td>
<td>$O(n + 4 \log n)$</td>
</tr>
<tr>
<td>integer priority queue</td>
<td>$O(1)$</td>
<td>$O(\log n)$</td>
<td>$O(1)$</td>
<td>$O(m + n \log n)$</td>
</tr>
</tbody>
</table>

\*amortized

Integers: Special Case

Thorup 1999: Single-source shortest paths in undirected graphs with positive integer edge lengths in $O(m)$ time.

Does not explore nodes by increasing distance from $s$.

Undirected Single-Source Shortest Paths with Positive Integer Weights in Linear Time

MIKKEL THORUP

Abstract. The single-source shortest paths problem (SSSP) is one of the classic problems in algorithmic graph theory. Given a positive-weighted graph $G$ with a source vertex, find the shortest \* path from $s$ to all other vertices in the graph.

Since 1939, all theoretical developments in SSSP for general directed and undirected graphs have been based on Dijkstra’s algorithm, visiting the vertices in order of increasing distance from $s$. Thus, any implementation of Dijkstra’s algorithm with the vertices according to their distance from $s$. However, we do not know how to sort in linear time.

Here, a deterministic linear-time and linear-space algorithm is presented for the undirected single source shortest path problem with positive integer weights. The algorithm avoids the sorting bottleneck by building a hierarchical bucketing structure, identifying vertex sets that may be visited in any order.