Shortest Paths Problem

Problem: find shortest paths in a directed graph with edge lengths (e.g., Google maps)

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 $O(m)$?

Shortest Path Example

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

set $A = V$  \hspace{1cm} ▷ Priority queue
set $d'(v) = \infty$ for all nodes  \hspace{1cm} ▷ Tentative arrival time
set $d'(s) = 0$  \hspace{1cm} ▷ Source

while $A$ not empty  \hspace{1cm} ▷ Nodes left to explore
  extract node $v \in A$ with smallest $d'(v)$ value  \hspace{1cm} ▷ Wave arrives at $v$
  for all edges $(v, w) \in A$ do
    if $d(v) + \ell(v, w) < d'(w)$ then  \hspace{1cm} ▷ Shorter path to $w$?
      $d'(w) = d(v) + \ell(v, w)$
      $\text{prev}(w) = v$
    end if
  end for
end while

Running Time?

Use heap-based priority queue for $A$

set $A = V$
set $d'(v) = \infty$ for all nodes
set $d'(s) = 0$

while $A$ not empty do
  extract node $v \in A$ with smallest $d'(v)$ value  \hspace{1cm} ▷ Extract-min
  for all edges $(v, w) \in A$ do
    if $d(v) + \ell(v, w) < d'(w)$ then
      $d'(w) = d(v) + \ell(v, w)$
      $\text{prev}(w) = v$
    end if
  end for
end while

- $n$ extract-min operations. $O(n \log n)$
- $m$ update-key operations. $O(m \log n)$
- Total: $O((m + n) \log n)$

Tracking the Shortest Path

Keep track of node that last updated arrival time $d'(v)$

Call it $\text{prev}(v) = \text{predecessor}$ in shortest path

set $A = V$
set $\text{prev}(v) = \text{null}$ for all nodes
set $d'(v) = \infty$ for all nodes
set $d'(s) = 0$

while $A$ not empty do
  extract node $v \in A$ with smallest $d'(v)$ value
  for all edges $(v, w) \in A$ do
    if $d(v) + \ell(v, w) < d'(w)$ then
      $d'(w) = d(v) + \ell(v, w)$
      $\text{prev}(w) = v$
    end if
  end for
end while

Proof of Correctness

- **Note (optimal substructure):** If $u$ is a node on a shortest path $s \leadsto u \leadsto v$, then $s \leadsto u$ is also a shortest path.

- **Idea:** nodes are explored in increasing order of distance from $s$.

- At each step, we have:
  - $A$: set of nodes still to explore
  - $S = V \setminus A$: explored nodes, $d'(v)$ assigned

- **Claim (invariant):**
  1. For $v \in S$, $d'(v)$ is length of shortest $s \leadsto v$ path
  2. For $v \in A$, $d'(v)$ is the length of the shortest $s \leadsto v$ path with all nodes in $S$ except $v$.

- **Proof:** By induction on $|S|$

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 $S$. ✓
The alternate $s-v$ path $P$ through $x$ and $y$ is already too long by the time it has left the set $S$.

**Figure 4.6** The shortest path $P_s$ and an alternate $s-v$ path $P$ through the node $y$. (Aho, Hopcroft & Tarjan)

- Consider a path with some prior node not in $S$.
  - Let $y \in A$ be the first such node on a path $s \rightarrow y \rightarrow v$.
  - But then $\ell(s \rightarrow y \rightarrow v) \geq \ell(s \rightarrow y) \geq d'(y) \geq d'(v)$, so we cannot have a shorter path.
- Critical point: segment $y \rightarrow v$ cannot have negative length.

**Induction Proof (2)**

**Proof: Maintaining the Invariant**

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

```plaintext
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 $s \rightarrow w$ for all edges $(v,w)$.

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

**Clicker Question 3**

Dijkstra’s algorithm works for nonnegative edges. (We’ll discuss the Bellman-Ford algorithm, which can handle negative edges.)

In general, there exists a shortest $s \rightarrow v$ path if

A. There are no negative-length edges on any path $s \rightarrow v$

B. There is no negative-length cycle on any path $s \rightarrow v$

C. Any path $s \rightarrow v$ that has a negative-length cycle also has a positive-length cycle

D. Any path $s \rightarrow v$ that has a negative-length cycle also has a positive-length cycle, longer in absolute value

**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 $s$.

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

Mikkel Thorup

AT&T Labs Research, Parkham Park, New Jersey

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

Since 1930, all theoretical developments in SSP 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 have 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.

**Table:**

<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</td>
<td>$O(d \log_2 n)$</td>
<td>$O(d \log_2 n)$</td>
<td>$O(d \log_2 n)$</td>
<td>$O(m \log_\log n)$</td>
</tr>
<tr>
<td>Fibonacci heap</td>
<td>(Joel 1975)</td>
<td>$O(\log n)$</td>
<td>$O(1)$</td>
<td>$O(m + n \log n)$</td>
</tr>
<tr>
<td>integer priority queue</td>
<td>(Thorup 2004)</td>
<td>$O(\log n)$</td>
<td>$O(1)$</td>
<td>$O(m + n \log n)$</td>
</tr>
</tbody>
</table>

$\dagger$ Amortized