Shortest Paths

- Shortest Paths
- Properties of Shortest Paths
- Dijkstra's Algorithm
- Shortest Paths on DAGs

Philip Bille

Shortest Paths

- Shortest paths. Given a directed, weighted graph \( G \) and vertex \( s \), find shortest path from \( s \) to all vertices in \( G \).

- Shortest path tree. Represent shortest paths in a tree from \( s \).
Applications

- Routing, scheduling, pipelining, ...

Shortest Paths

- Shortest Paths
- Properties of Shortest Paths
- Dijkstra's Algorithm
- Shortest Paths on DAGs

Properties of Shortest Paths

- Assume for simplicity:
  - All vertices are reachable from s.
  - \( \Rightarrow \) a (shortest) path to each vertex always exists.

![Diagram showing properties of shortest paths]

- Subpath property. Any subpath of a shortest path is a shortest path.
- Proof.
  - Consider shortest path from \( s \) to \( t \) consisting of \( p_1, p_2 \) and \( p_3 \).
  - Assume \( q_2 \) is shorter than \( p_2 \).
  - \( \Rightarrow \) Then \( p_1, q_2 \) and \( p_3 \) is shorter than \( p \).
Shortest Paths

- Shortest Paths
- Properties of Shortest Paths
- Dijkstra’s Algorithm
- Shortest Paths on DAGs

Dijkstra’s Algorithm

- Goal. Given a directed, weighted graph with non-negative weights and a vertex s, compute shortest paths from s to all vertices.

- Dijkstra’s algorithm.
  - Maintains distance estimate \( v.d \) for each node \( v \) = length of shortest known path from s to \( v \).
  - Updates distance estimates by relaxing edges.

Dijkstra’s Algorithm

\[
\begin{align*}
\text{RELAX}(u,v) & \quad \text{if } (v.d > u.d + w(u,v)) \\
& \quad v.d = u.d + w(u,v)
\end{align*}
\]
Dijkstra’s Algorithm

- **Lemma.** Dijkstra’s algorithms computes shortest paths.
- **Proof.**
  - Consider some step after growing tree $T$ and assume distances in $T$ are correct.
  - Consider closest vertex $u$ of $s$ not in $T$.
  - Shortest path from $s$ to $u$ ends with an edge $(v,u)$.
  - $v$ is closer than $u$ to $s$ $\Rightarrow$ $v$ is in $T$. ($u$ was closest not in $T$)
  - $\Rightarrow$ shortest path to $u$ is in $T$ except last edge $(u,v)$.
  - Dijkstra adds $(u,v)$ to $T$ $\Rightarrow$ $T$ is shortest path tree after $n-1$ steps.

Dijkstra’s Algorithm

- **Implementation.** How do we implement Dijkstra’s algorithm?
- **Challenge.** Find vertex with smallest distance estimate.
Dijkstra's Algorithm

Implementation. Maintain vertices outside $T$ in priority queue.
- Key of vertex $v = v.d$.
- In each step:
  - Find vertex $u$ with smallest distance estimate = EXTRACT-MIN
  - Relax edges that $u$ point to with DECREASE-KEY.

Priority queues and Dijkstra's algorithm. Complexity of Dijkstra’s algorithm depend on priority queue.
- $n$ INSERT
- $n$ EXTRACT-MIN
- $< m$ DECREASE-KEY

<table>
<thead>
<tr>
<th>Priority queue</th>
<th>INSERT</th>
<th>EXTRACT-MIN</th>
<th>DECREASE-KEY</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>array</td>
<td>$O(1)$</td>
<td>$O(n)$</td>
<td>$O(1)$</td>
<td>$O(n)$</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>Fibonacci heap</td>
<td>$O(1)\dagger$</td>
<td>$O(\log n)\dagger$</td>
<td>$O(1)\dagger$</td>
<td>$O(m + n \log n)$</td>
</tr>
</tbody>
</table>

$\dagger = \text{amortized}$

Greed. Dijkstra’s algorithm is a greedy algorithm.

Dijkstra's Algorithm

\[
\text{DIJKstra}(G, s) \\
\text{for all vertices } v \in V \\
v.d = \infty \\
v.\pi = \text{null} \\
\text{INSERT}(P, v) \\
\text{DECREASE-KEY}(P, s, 0) \\
\text{while (} P \neq \emptyset \text{)} \\
\hspace{1em} u = \text{EXTRACT-MIN}(P) \\
\hspace{2em} \text{for all } v \text{ that } u \text{ point to} \\
\hspace{3em} \text{RELAX}(u, v) \\
\]

Dijkstra algorithm.

- Time.
  - $n$ EXTRACT-MIN
  - $n$ INSERT
  - $< m$ DECREASE-KEY
  - Total time with min-heap. $O(n \log n + n \log n + m \log n) = O(m \log n)$

Edsger W. Dijkstra

- Edsger Wybe Dijkstra (1930-2002)
- Contributions. Foundations for programming, distributed computation, program verifications, etc.
- Quotes. “Object-oriented programming is an exceptionally bad idea which could only have originated in California.”
  - “The use of COBOL cripples the mind; its teaching should, therefore, be regarded as a criminal offence.”
  - “APL is a mistake, carried through to perfection. It is the language of the future for the programming techniques of the past: it creates a new generation of coding bums.”
Shortest Paths

- Shortest Paths
- Properties of Shortest Paths
- Dijkstra's Algorithm
- Shortest Paths on DAGs

Challenge. Is it computationally easier to find shortest paths on DAGs?

DAG shortest path algorithm.
- Process vertices in topological order.
- For each vertex v, relax all edges from v.
- Also works for negative edge weights.

Shortest Paths on DAGs

Implementation.
- Sort vertices in topological order.
- Relax outgoing edges from each vertex.
- Total time. $O(m + n)$.

Lemma. Algorithm computes shortest paths in DAGs.

Proof.
- Consider some step after growing tree $T$ and assume distances in $T$ are correct.
- Consider next vertex $u$ of $s$ not in $T$.
- Any path to $u$ consists vertices in $T$ + edge $e$ to $u$.
- Edge $e$ is relaxed $\Rightarrow$ distance to $u$ is shortest.
Shortest Paths Variants

- **Vertices**
  - Single source.
  - Single source, single target.
  - All-pairs.

- **Edge weights**
  - Non-negative.
  - Arbitrary.
  - Euclidian distances.

- **Cycles**
  - No cycles
  - No negative cycles.

---

Shortest Paths

- Shortest Paths
- Properties of Shortest Paths
- Dijkstra's Algorithm
- Shortest Paths on DAGs