CISC 320 Introduction to Algorithms
Fall 2003

Lecture 12
Shortest Paths

Growing a shortest-path tree
- Start at source vertex \( s \) and "branches out" by selecting edges that lead to new vertices
- For each vertex \( z \) in the fringe, there is at least one tree vertex \( v \) such that \( vz \) is an edge of \( G \).
  - why?
    (otherwise how can \( z \) be in the fringe)
- Choose \( v \) such that \( d(s,v) + W(vz) \) is minimized

Theorem 8.6 Let \( G = (V,E,W) \). \( V' \) is a subset of \( V \), and \( V' \) contains the source \( s \). Let \( d(s,y) \) be the shortest distance in \( G \) from \( s \) to \( y \), for each \( y \) in \( V' \). If edge \( yz \) is chosen to minimize \( d(s,y) + W(yz) \) over all edges with \( y \) in \( V' \) and \( z \) in \( V - V' \), then the path consisting of a shortest path from \( s \) to \( y \) followed by the edge \( yz \) is a shortest path from \( s \) to \( z \) in \( V' \).

Proof: For any other path \( P' \) from \( s \) to \( z \), we have

\[
W(P') = d(s,y) + W(yz) \leq d(s,v) + W(vz)
\]

That is why vertex \( z \) is chosen by the algorithm to expand \( V' \).

Dijkstra’s Shortest-Path Algorithm

Given a weighted graph \( G = (V,E,W) \) and a source vertex \( s \), find a shortest path from \( s \) to each vertex \( v \).

Dijkstra(G,w,s)
1. Initialize all vertices as unseen
2. Start the tree with the specified source vertex \( s \); reclassify it as tree
3. Define \( d(s,s) = 0 \)
4. Reclassify all vertices adjacent to \( s \) as fringe
5. While there are fringe vertices
   1. Select a tree vertex \( t \) and a fringe vertex \( v \)
   2. Such that \( d(s,t) + W(tv) \) is minimum
   3. Reclassify \( v \) as tree; add edge \( tv \) to the tree
   4. Define \( d(s,v) = d(s,t) + W(tv) \)
   5. Reclassify all unseen vertices adjacent to \( v \) as fringe

Dijkstra’s Shortest-Path Algorithm
The tree so far

\[ d(A, B) + W(B, C) = 6 \]
\[ d(A, G) + W(A, F) = 9 \]
\[ d(A, A) + W(A, G) = 5 \]

Dijkstra Algorithm

Graph \( G \), weights \( w \), source \( s \)

1. for each \( v \in V[G] \)
   - \( d(v) \leftarrow \infty \)
   - \( P(v) \leftarrow \text{nil} \)
   - \( S \leftarrow \text{empty} \)
2. while \( Q \) is not empty
   - \( u \leftarrow \text{Extract-Min}(Q) \)
   - for each \( v \in \text{Adj}(u) \)
     - if \( d(v) > d(u) + W(u, v) \)
       - then \( d(v) \leftarrow d(u) + W(u, v) \)
       - \( P(v) \leftarrow u \)

Time analysis

- Initialization of priority queue (as a binary heap): \( O(V) \)
- Extract-min is called \( |V| \) times
  - Each Extract-Min takes \( O(\lg V) \) time
  - For each adjacent vertex, update its distance (with Decrease-Key operation: \( O(\lg v) \))

Total running time: \( O((V+E)\lg V) \)

Dijkstra Algorithm

Running time with different \( Q \) implementations

1. Array
   - Extract-Min: \( O(V) \)
   - Decrease-Key: \( O(1) \)
   - Total: \( O(V^2) \)
2. Binary heap
   - Extract-Min: \( O(\lg V) \)
   - Decrease-Key: \( O(\lg V) \)
   - Total: \( O(V\lg V) \)
   - Fibonacci heap
     - Extract-Min: \( O(\lg V) \)
     - Decrease-Key (amortized): \( O(1) \)
     - Total: \( O(V\lg V + E) \)

Floyd algorithm

\( W \) is the weight matrix of graph \( G = (V, E, W) \).

\[ w_{ij} = W(v, v_j) \]
\[ = 0 \] if \( v_i \notin E \) and \( i \neq j \)
\[ = \min(0, W(v_i)) \] if \( v_i \notin E \)

\( D \) with entry \( d_{ij} \) = the shortest-path distance from \( v_i \) to \( v_j \).

Lemma

If a shortest path from \( v_i \) to \( v_j \) goes through an intermediate vertex \( v_k \), then the segments of that path from \( v_i \) to \( v_k \) and from \( v_k \) to \( v_j \) are themselves shortest paths.
Path $p$ is the shortest path from $i$ to $j$. Path $p_{ij}$, the portion of path $p$ from $i$ to $k$, is the shortest path from $i$ to $k$. Suppose $p'_i$ is the shortest path from $i$ to $k$, then $p_{ij}$ and $p'_{ij}$ would make a path with shorter distance than path $p$.

Let $k$ be the highest-indexed intermediate vertex in path $p$. The highest-indexed intermediate vertex in path $p_i$ must have an index lower than $k$, that is, all the intermediate vertices of $p_i$ are from subset $[v_1, ..., v_{k-1}]$. The same holds for path $p_j$.

Let $d^{(0)}_{ij}$ be the weight of a shortest path from $i$ to $j$ with all intermediate vertices in the set $[v_1, ..., v_j]$. We shall have

$$d^{(k)}_{ij} = \min(d^{(k-1)}_{ij}, d^{(k-1)}_{ik} + d^{(k-1)}_{kj}) \quad \text{for} \quad k \geq 1$$

$$d^{(k)}_{ij} = m_{ij} \quad \text{for} \quad k = 0$$

**Floyd-APSP($W$, $D$)**

1. $D \leftarrow W$
2. for $k \leftarrow 1$ to $n$
3. for $i \leftarrow 1$ to $n$
4. for $j \leftarrow 1$ to $n$
5. $D[i][j] \leftarrow \min(D[i][j], D[i][k] + D[k][j])$
6. return $D$

**Time analysis:**
- $O(n^3)$

**Transitive closure of a binary relation**

- Binary relation on a set $S$ is a subset of $S \times S$. If $(s, s) \in A$, we say $s$ is $A$-related to $s$, and use notation $sA_s$.
- Transitive closure of $A$ is a binary relation, denoted as $R$, such that, $sRa$ if and only if there is a path from $s$ to $s$ in graph $G=(S, A)$.
- Transitive closure is also called reachability relation. $R$ matrix is an $n \times n$ matrix

$$r_{ij} = 1 \quad \text{if there is a path from } s_i \text{ to } s_j$$

$$= 0 \quad \text{otherwise}$$

**Warshall Algorithm**

- Define $t^{(0)}_{ij} = 1$ if there exists a path in graph $G$ from vertex $i$ to vertex $j$ with all intermediate vertices in the set $\{1, 2, ..., k\}$, and 0 otherwise.

$$t^{(k)}_{ij} = 1 \quad \text{if } i = j \text{ or } (i, j) \in E$$

$$= 0 \quad \text{if } i \neq j \text{ and } (i, j) \notin E$$

$$t^{(k)}_{ij} = \bigvee (t^{(k-1)}_{ik} \land t^{(k-1)}_{kj})$$

- Running time: $O(n^3)$
Transitive-Closure(G)
1. for i ← 1 to n
2. for j ← 1 to n
3. if i=j or (i,j) ∈ E[G]
4. then t(0)_{ij} ← 1
5. else t(0)_{ij} ← 0
6. for k ← 1 to n
7. for i ← 1 to n
8. for j ← 1 to n
9. t(k)_{ij} ← t(k-1)_{ij} ∨ (t(k-1)_{ik} ∧ t(k-1)_{kj})
10. return T(n)