http://www.cs.unh.edu/~ruml/cs758

2 handouts: slides, asst 10
Shortest Path Problems
Problems

Single source/destination pair
Single source, all destinations
Single destination, all sources
All-pairs

Non-uniform weights?
Negative edges?
Negative cycles?
Floyd-Warshall
1936–2001; Turing Award ’78
BA at 17, prof at CMU at 27,
full prof at Stanford at 32. No
PhD.
invented ‘method of
invariants’, parsing, dithering,
most cited author in TAoCP
students included Tarjan,
Rivest
Can it be faster than $V \times$ single-source?

How to use optimal substructure?
The Idea

\[ d_{ij}^k = \text{shortest path from } i \text{ to } j \text{ using intermediate vertices in } 1..k \]

How to construct if we know all-pairs shortest paths using only intermediate vertices in $1..k - 1$?
The Algorithm

1. \( D^0 \leftarrow \) the \( n \times n \) weighted adjacency matrix
2. for \( k = 1 \) to \( n \)
3. for \( i = 1 \) to \( n \)
4. for \( j = 1 \) to \( n \)
5. \( d_{ij}^k \leftarrow \min(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1}) \)
6. return \( D^n \)

Correctness?
The Algorithm

1. \(D^0 \leftarrow \text{the } n \times n \text{ weighted adjacency matrix}\)
2. for \(k = 1\) to \(n\)
3. for \(i = 1\) to \(n\)
4. for \(j = 1\) to \(n\)
5. \(d_{ij}^k \leftarrow \min(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1})\)
6. return \(D^n\)

correctness? induction on
The Algorithm

1. \( D^0 \leftarrow \text{the } n \times n \text{ weighted adjacency matrix} \)
2. for \( k = 1 \) to \( n \)
3. \hspace{1em} for \( i = 1 \) to \( n \)
4. \hspace{2em} for \( j = 1 \) to \( n \)
5. \hspace{3em} \( d_{ij}^k \leftarrow \min(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1}) \)
6. return \( D^n \)

Correctness? induction on allowable intermediate vertices
Running time?
The Algorithm

1. \( D^0 \leftarrow \text{the } n \times n \text{ weighted adjacency matrix} \\
2. \text{for } k = 1 \text{ to } n \\
3. \quad \text{for } i = 1 \text{ to } n \\
4. \quad \quad \text{for } j = 1 \text{ to } n \\
5. \quad \quad \quad d_{ij}^k \leftarrow \min(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1}) \\
6. \text{return } D^n \\

correctness? induction on allowable intermediate vertices 
running time? \( O(V^3) \) 
negative weights?
The Algorithm

1. \( D^0 \leftarrow \text{the } n \times n \text{ weighted adjacency matrix} \)
2. for \( k = 1 \) to \( n \)
3. for \( i = 1 \) to \( n \)
4. for \( j = 1 \) to \( n \)
5. \( d_{ij}^k \leftarrow \min(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1}) \)
6. return \( D^n \)

Correctness? induction on allowable intermediate vertices
Running time? \( O(V^3) \)
Negative weights? no problem!
Solutions?
The Algorithm

1. \( D^0 \leftarrow \) the \( n \times n \) weighted adjacency matrix
2. for \( k = 1 \) to \( n \)
3. for \( i = 1 \) to \( n \)
4. for \( j = 1 \) to \( n \)
5. \( d_{ij}^k \leftarrow \min(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1}) \)
6. return \( D^n \)

Correctness? induction on allowable intermediate vertices
Running time? \( O(V^3) \)
Negative weights? no problem!
Solutions? predecessor pointers
Shortest Paths
Floyd-Warshall
- Bob Floyd
- All-Pairs
- The Idea
- Algorithm
- Break

Network Flow

- asst 9
- asst 10
The Problem

Given directed graph, source and sink, find flow of maximum value.

logistics
network design
tasking

flow constraints: edge capacity, conservation at vertices

\[ 0 \leq f(u, v) \leq c(u, v) \]

\[ \forall v \in V - \{s, t\}, \sum_{v \in V} f(v, u) = \sum_{v \in V} f(u, v) \]

details: removing ‘anti-parallel’ edges, multiple sources or sinks
Iteratively augment flow until no augmenting path exists.

Find augmentation via ‘residual network’ $G_f$ with costs

$$c_f(u, v) = \begin{cases} 
  c(u, v) - f(u, v) & \text{if } (u, v) \in E \\
  f(v, u) & \text{if } (v, u) \in E \\
  0 & \text{otherwise}
\end{cases}$$

residual network has reverse flow edges: not a legal ‘flow network’

to augment $(u, v)$, add $f(u, v)$ and subtract $f(v, u)$
Ford-Fulkerson: The Algorithm

1. for each edge, \((u, v).f \leftarrow 0\)
2. while there exists an \(s \rightsquigarrow t\) path \(p\) in the residual network
3. \(c_f(p) \leftarrow \text{min capacity of edges along } p\)
4. for each edge \((u, v)\) in \(p\)
5. if \((u, v) \in E\)
6. \((u, v).f \leftarrow (u, v).f + c_f(p)\)
7. else
8. \((u, v).f \leftarrow (v, u).f - c_f(p)\)
For example:

- What’s still confusing?
- What question didn’t you get to ask today?
- What would you like to hear more about?

Please write down your most pressing question about algorithms and put it in the box on your way out.

*Thanks!*