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

Shortest Path Problems

Problems

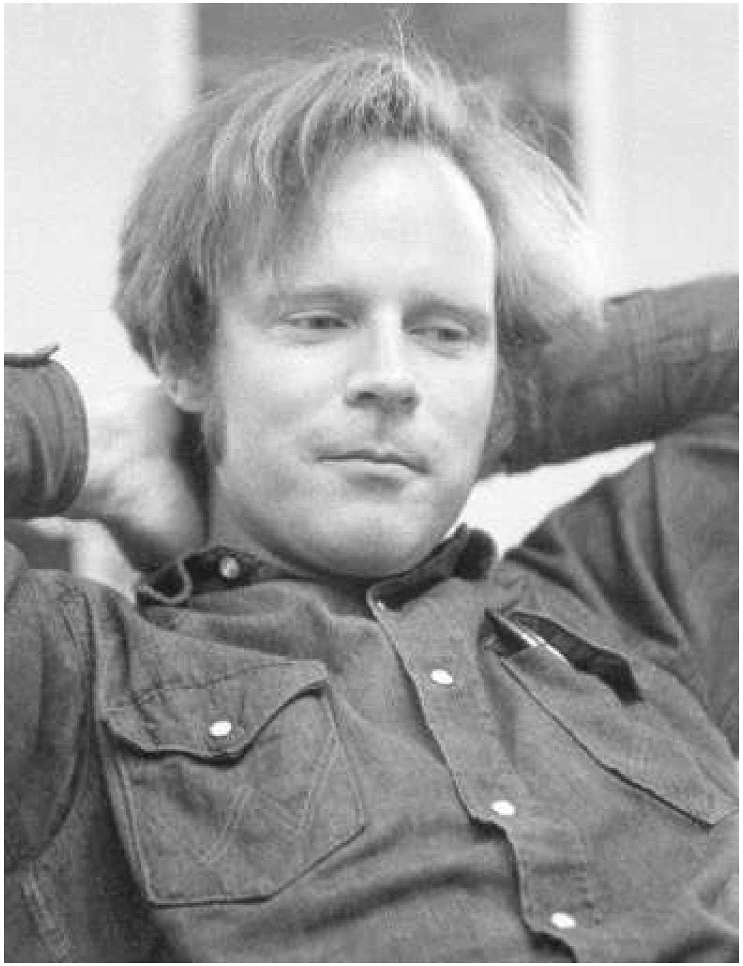
single source/destination pair
single source, all destinations
single destination, all sources
all-pairs

non-uniform weights?
negative edges?
negative cycles?

Floyd-Warshall

Robert W Floyd

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 = shortest path from i to j 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. $D^0 \leftarrow$ 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

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?

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?

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?

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 pointer inherited from d_{kj}^{k-1} as necessary

Break

- asst 10
- `https://doi.org/10.1117/1.AP.6.5.056011`

Random Problems

Your startup is booming and there is a lot to do. For each task, you have a list of the tasks that must be completed before it can begin. Each task takes one hour. You can assume an infinite supply of workers, each of whom is qualified to perform any of the tasks. Give an algorithm to find the minimum time required to accomplish all of the tasks.

Give an algorithm for finding, from among all the shortest paths from s to t in a graph, one that has the fewest edges.

Network Flow

The Problem

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

logistics
network design
tasking

flow constraints: edge capacity, conservation at vertices

$$0 \leq f(u, v) \leq c(u, v)$$

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

details: removing 'anti-parallel' edges, multiple sources or sinks

Ford-Fulkerson: The Idea

Iteratively augment flow until no augmenting path exists.

Find augmentation via 'residual network' G_f with costs

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

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. $\quad c_f(p) \leftarrow$ min capacity of edges along p
4. $\quad$ for each edge (u, v) in p
5. $\quad\quad$ if $(u, v) \in E$
6. $\quad\quad\quad (u, v).f \leftarrow (u, v).f + c_f(p)$
7. $\quad\quad$ else
8. $\quad\quad\quad (v, u).f \leftarrow (v, u).f - c_f(p)$

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!