Design and Analysis of Algorithms VI

Shortest Path

Guoqiang Li
School of Software

Shanghai Jiao Tong University

The Problem

Let $G = (V, E)$ be a directed graph. Assume that each edge $(i, j) \in E$ has an associated weight c_{ij}.

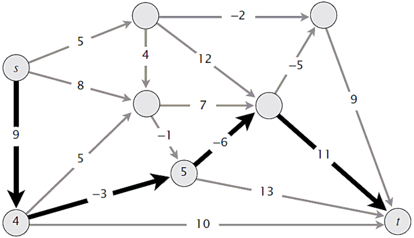
Dijkstra's Algorithm is given for finding shortest paths in graphs with positive edge costs.

Here we consider the more complex problem in which we seek shortest paths when costs may be negative.

Bellman-Ford-Moore Algorithm

Shortest paths with negative weights

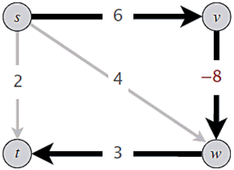
Shortest-path problem. Given a digraph $G = (V, E)$, with arbitrary edge lengths ℓ_{vw}, find shortest path from source node s to destination node t.



Failed Attempts

SHANGHAI JIAO TONG UNIVERSITY

Dijkstra. May not produce shortest paths when edge lengths are negative.

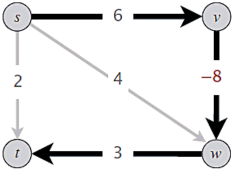


Dijkstra selects the vertices in the orderss, t, w, v
But shortest path from s to t is $s \to v \to w \to t$.

Failed Attempts

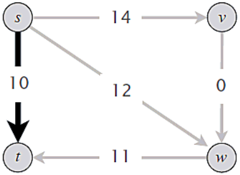
SHANGHAI JIAO TONG UNIVERSITY

Dijkstra. May not produce shortest paths when edge lengths are negative.



Dijkstra selects the vertices in the orders, t, w, v
But shortest path from s to t is $s \to v \to w \to t$.

Reweighting. Adding a constant to every edge length does not necessarily make Dijkstra's algorithm produce shortest paths.



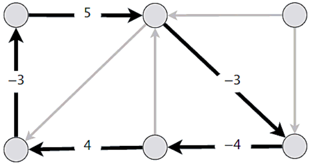
Adding 8 to each edge weight changes the shortest path from $s \to v \to w \to t$ to $s \to t$.

Negative Cycles

SHANGHAI JIAO TONG UNIVERSITY

Definition

A negative cycle is a directed cycle for which the sum of its edge lengths is negative.



Shortest Paths and Negative Cycles

Lemma 1

If some $v \rightsquigarrow t$ path contains a negative cycle, then there does not exist a shortest $v \rightsquigarrow t$ path.

Shortest Paths and Negative Cycles

Lemma 1

If some $v \rightsquigarrow t$ path contains a negative cycle, then there does not exist a shortest $v \rightsquigarrow t$ path.

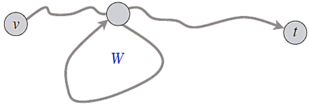
Proof.

Lemma 1

If some $v \rightsquigarrow t$ path contains a negative cycle, then there does not exist a shortest $v \rightsquigarrow t$ path.

Proof.

If there exists such a cycle W, then can build a $v \rightsquigarrow t$ path of arbitrarily negative length by detouring around W as many times as desired.



Shortest Paths and Negative Cycles

Lemma 2

If G has no negative cycles, then there exists a shortest $v \rightsquigarrow t$ path that is simple (and has $\leq n-1$ edges).

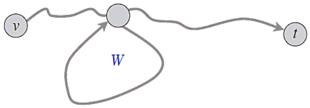
SHANGHAI JIAO TONG UNIVERSITY

Lemma 2

If G has no negative cycles, then there exists a shortest $v \rightsquigarrow t$ path that is simple (and has $\leq n-1$ edges).

Proof.

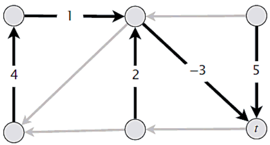
- Among all shortest $v \rightsquigarrow t$ paths, consider one that uses the fewest edges.
- If that path P contains a directed cycle W, can remove the portion of P corresponding to W without increasing its length.

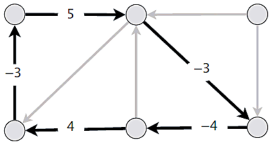


Shortest-Paths and Negative-Cycle Problems

Single-destination shortest-paths problem. Given a digraph $G = (V, E)$ with edge lengths ℓ_{vw} (but no negative cycles) and a distinguished node t, find a shortest $v \rightsquigarrow t$ path for every node v.

Negative-cycle problem. Given a digraph $G = (V, E)$ with edge lengths ℓ_{vw}, find a negative cycle (if one exists).





Dynamic Programming

Definition. $OPT(i, v)$: length of shortest $v \rightsquigarrow t$ path that uses $\leq i$ edges.
Goal $OPT(n-1, v)$ for each v.

Dynamic Programming

Definition. $OPT(i, v)$: length of shortest $v \rightsquigarrow t$ path that uses $\leq i$ edges.
Goal $OPT(n-1, v)$ for each v.

Case 1. Shortest $v \rightsquigarrow t$ path uses $\leq i-1$ edges.

- $OPT(i, v) = OPT(i-1, v)$

Dynamic Programming

Definition. $OPT(i, v)$: length of shortest $v \rightsquigarrow t$ path that uses $\leq i$ edges.
Goal $OPT(n-1, v)$ for each v.

Case 1. Shortest $v \rightsquigarrow t$ path uses $\leq i-1$ edges.

- $OPT(i, v) = OPT(i-1, v)$

Case 2. Shortest $v \rightsquigarrow t$ path uses exactly i edges.

- if (v, w) is first edge in shortest such $v \rightsquigarrow t$ path, incur a cost of ℓ_{vw}.
- Then, select best $w \rightsquigarrow t$ path using $\leq i-1$ edges.

Dynamic Programming

SHANGHAI JIAO TONG UNIVERSITY

Definition. $OPT(i, v)$: length of shortest $v \rightsquigarrow t$ path that uses $\leq i$ edges.
Goal $OPT(n-1, v)$ for each v.

Case 1. Shortest $v \rightsquigarrow t$ path uses $\leq i-1$ edges.

- $OPT(i, v) = OPT(i-1, v)$

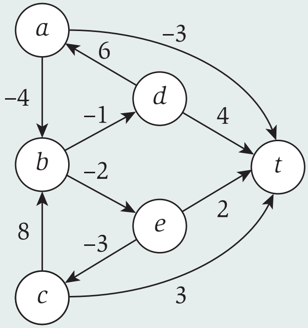
Case 2. Shortest $v \rightsquigarrow t$ path uses exactly i edges.

- if (v, w) is first edge in shortest such $v \rightsquigarrow t$ path, incur a cost of ℓ_{vw}.
- Then, select best $w \rightsquigarrow t$ path using $\leq i-1$ edges.

Bellman equation.

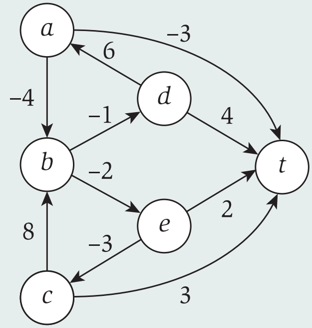
$$OPT(i, v) = \begin{cases} 0 & \text{if } i = 0 \text{ and } v = t \\ \infty & \text{if } i = 0 \text{ and } v \neq t \\ \min \left\{ OPT(i-1, v), \min\limits_{(v,w) \in E} \{ OPT(i-1, w) + \ell_{vw} \} \right\} & \text{if } i > 0 \end{cases}$$

An Example



An Example

SHANGHAI JIAO TONG UNIVERSITY



	0	1	2	3	4	5
t	0	0	0	0	0	0
a	∞	−3	−3	−4	−6	−6
b	∞	∞	0	−2	−2	−2
c	∞	3	3	3	3	3
d	∞	4	3	3	2	0
e	∞	2	0	0	0	0

Implementation

SHANGHAI JIAO TONG UNIVERSITY

```
ShortestPaths(V, E, ℓ, t)

for each node (v ∈ V) do
    M[0, v] ← ∞;
end
M[0, t] ← 0;
for i = 1 to n − 1 do
    for each node v ∈ V do
        M[i, v] ← M[i − 1, v];
        for each edge (v, w) ∈ E do
            M[i, v] ← min {M[i, v], M[i − 1, w] + ℓ_vw};
        end
    end
end
```

Implementation

Theorem

Given a digraph $G = (V, E)$ with no negative cycles, the DP algorithm computes the length of a shortest $v \rightsquigarrow t$ path for every node v in $\Theta(|V||E|)$ time and $\Theta(|V|^2)$ space.

Implementation

Theorem

Given a digraph $G = (V, E)$ with no negative cycles, the DP algorithm computes the length of a shortest $v \rightsquigarrow t$ path for every node v in $\Theta(|V||E|)$ time and $\Theta(|V|^2)$ space.

Proof.

Implementation

SHANGHAI JIAO TONG UNIVERSITY

Theorem

Given a digraph $G = (V, E)$ with no negative cycles, the DP algorithm computes the length of a shortest $v \rightsquigarrow t$ path for every node v in $\Theta(|V||E|)$ time and $\Theta(|V|^2)$ space.

Proof.

- Table requires $\Theta(|V|^2)$ space.
- Each iteration i takes $\Theta(|E|)$ time since we examine each edge once.

Implementation

Theorem

Given a digraph $G = (V, E)$ with no negative cycles, the DP algorithm computes the length of a shortest $v \rightsquigarrow t$ path for every node v in $\Theta(|V||E|)$ time and $\Theta(|V|^2)$ space.

Proof.

- Table requires $\Theta(|V|^2)$ space.
- Each iteration i takes $\Theta(|E|)$ time since we examine each edge once.

Finding the shortest paths.

Implementation

Theorem

Given a digraph $G = (V, E)$ with no negative cycles, the DP algorithm computes the length of a shortest $v \rightsquigarrow t$ path for every node v in $\Theta(|V||E|)$ time and $\Theta(|V|^2)$ space.

Proof.

- Table requires $\Theta(|V|^2)$ space.
- Each iteration i takes $\Theta(|E|)$ time since we examine each edge once.

Finding the shortest paths.

- Approach 1: Maintain $successor[i, v]$ that points to next node on a shortest $v \rightsquigarrow t$ path using $\leq i$ edges.
- Approach 2: Compute optimal lengths $M[i, v]$ and consider only edges with $M[i, v] = M[i-1, w] + \ell_{vw}$. Any directed path in this subgraph is a shortest path.

Practical Improvements

Space optimization. Maintain two 1D arrays (instead of 2D array).

- $d[v]$: length of a shortest $v \rightsquigarrow t$ path that we have found so far.
- $successor[v]$: next node on a $v \rightsquigarrow t$ path.

Practical Improvements

Space optimization. Maintain two 1D arrays (instead of 2D array).

- $d[v]$: length of a shortest $v \rightsquigarrow t$ path that we have found so far.
- $successor[v]$: next node on a $v \rightsquigarrow t$ path.

Performance optimization. If $d[w]$ was not updated in iteration $i - 1$, then no reason to consider edges entering w in iteration i.

Efficient Implementation

```
BELLMAN-FORD-MOORE(V, E, c, t)

for each node v ∈ V do
    d[v] ← ∞;
    successor[v] ← null;
end
d[t] ← 0;
for i = 1 to n − 1 do
    for each node w ∈ V do
        if d[w] was updated in previous pass then
            for each edge (v, w) ∈ E do
                if (d[v] > d[w] + ℓ_vw) then
                    d[v] ← d[w] + ℓ_ww;
                    successor[v] ← w;
                end
            end
        end
    end
    if no d[·] value changed in pass i then BREAK;
end
```

Bellman-Ford-Moore: Analysis

Shanghai Jiao Tong University

Theorem

Assuming no negative cycles, Bellman-Ford-Moore computes the lengths of the shortest $v \rightsquigarrow t$ paths in $O(|V||E|)$ time and $\Theta(|V|)$ extra space.

Bellman-Ford-Moore: Analysis

Theorem

Assuming no negative cycles, Bellman-Ford-Moore computes the lengths of the shortest $v \rightsquigarrow t$ paths in $O(|V||E|)$ time and $\Theta(|V|)$ extra space.

Remark

Bellman-Ford-Moore is typically faster in practice.

- Edge (v, w) considered in pass $i + 1$ only if $d[w]$ updated in pass i.
- If shortest path has k edges, then algorithm finds it after $\leq k$ passes.

Bellman-Ford-Moore: Analysis

Claim

Throughout Bellman-Ford-Moore, following the $successor[v]$ *pointers gives a directed path from* v *to* t *of length* $d[v]$.

Bellman-Ford-Moore: Analysis

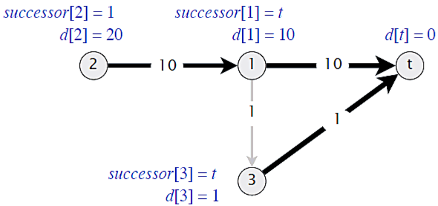
Claim

Throughout Bellman-Ford-Moore, following the $successor[v]$ pointers gives a directed path from v to t of length $d[v]$.

Counterexample. Claim is false!

- Length of successor $v \rightsquigarrow t$ path may be strictly shorter than $d[v]$.

consider nodes in order: t, 1, 2, 3



Bellman-Ford-Moore: Analysis

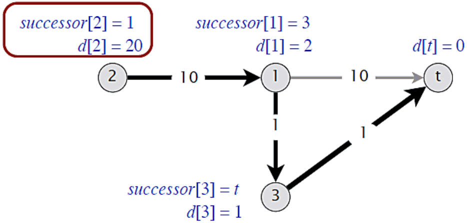
Claim

Throughout Bellman-Ford-Moore, following the $successor[v]$ pointers gives a directed path from v to t of length $d[v]$.

Counterexample. Claim is false!

- Length of successor $v \rightsquigarrow t$ path may be strictly shorter than $d[v]$.

consider nodes in order: t, 1, 2, 3



Bellman-Ford-Moore: Analysis

SHANGHAI JIAO TONG UNIVERSITY

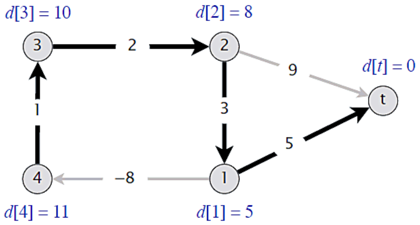
Claim

Throughout Bellman-Ford-Moore, following the $successor[v]$ pointers gives a directed path from v to t of length $d[v]$.

Counterexample. Claim is false!

- Length of successor $v \rightsquigarrow t$ path may be strictly shorter than $d[v]$.
- If negative cycle, successor graph may have directed cycles.

consider nodes in order: $t, 1, 2, 3$, 4



Bellman-Ford-Moore: Analysis

SHANGHAI JIAO TONG UNIVERSITY

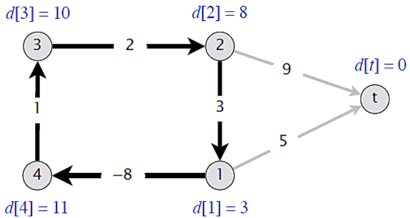
Claim

Throughout Bellman-Ford-Moore, following the $successor[v]$ pointers gives a directed path from v to t of length $d[v]$.

Counterexample. Claim is false!

- Length of successor $v \rightsquigarrow t$ path may be strictly shorter than $d[v]$.
- If negative cycle, successor graph may have directed cycles.

consider nodes in order: $t, 1, 2, 3, 4$



Finding the Shortest Paths

SHANGHAI JIAO TONG UNIVERSITY

Lemma

Any directed cycle W in the successor graph is a negative cycle.

Proof.

Finding the Shortest Paths

Lemma

Any directed cycle W in the successor graph is a negative cycle.

Proof.

- If $successor[v] = w$, we must have $d[v] \geq d[w] + \ell_{vw}$.
  (LHS and RHS are equal when $successor[v]$ is set; $d[w]$ can only decrease; $d[v]$ decreases only when $successor[v]$ is reset)
- Let $v_1 \to v_2 \to \ldots \to v_k \to v_1$ be the sequence of nodes in a directed cycle W.
- Assume that (v_k, v_1) is the last edge in W added to the successor graph.
- Just prior to that:

$$\begin{array}{lll} d[v_1] & \geq d[v_2] & +\ell(v_1, v_2) \\ d[v_2] & \geq d[v_3] & +\ell(v_2, v_3) \\ \vdots & \vdots & \vdots \\ d[v_{k-1}] & \geq d[v_k] & +\ell(v_{k-1}, v_k) \\ d[v_k] & > d[v_1] & +\ell(v_{k-1}, v_1) \end{array}$$

- Adding inequalities yields $\ell(v_1, v_2) + \ell(v_2, v_3) + \ldots + \ell(v_{k-1}, v_k) + \ell(v_k, v_1) < 0$

Finding the Shortest Paths

Theorem

Assuming no negative cycles, Bellman-Ford-Moore finds shortest $v \rightsquigarrow t$ paths for every node v in $O(mn)$ time and $\Theta(n)$ extra space.

Finding the Shortest Paths

Theorem

Assuming no negative cycles, Bellman-Ford-Moore finds shortest $v \rightsquigarrow t$ paths for every node v in $O(mn)$ time and $\Theta(n)$ extra space.

Proof.

Finding the Shortest Paths

Theorem

Assuming no negative cycles, Bellman-Ford-Moore finds shortest $v \rightsquigarrow t$ paths for every node v in $O(mn)$ time and $\Theta(n)$ extra space.

Proof.

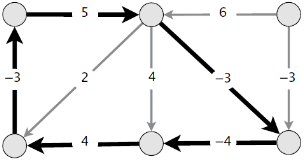
- The successor graph cannot have a directed cycle.
- Thus, following the successor pointers from v yields a directed path to t.
- Let $v = v_1 \to v_2 \to \ldots \to v_k = t$ be the nodes along this path P.
- Upon termination, if $successor[v] = w$, we must have $d[v] = d[w] + \ell_{vw}$. (LHS and RHS are equal when $successor[v]$ is set; $d[\cdot]$ did not change)
- Thus,

$$
\begin{array}{lll}
d[v_1] & = d[v_2] & + \ell(v_1, v_2) \\
d[v_2] & = d[v_3] & + \ell(v_2, v_3) \\
\vdots & \vdots & \vdots \\
d[v_{k-1}] & = d[v_k] & + \ell(v_{k-1}, v_k)
\end{array}
$$

- Adding equations yields $d[v] = d[t] + \ell(v_1, v_2) + \ell(v_2, v_3) + \ldots + \ell(v_{k-1}, v_k)$

Detecting Negative Cycles

Negative cycle detection problem. Given a digraph $G = (V, E)$, with edge lengths ℓ_{vw}, find a negative cycle (if one exists).



Detecting Negative Cycles

Lemma

If $OPT(n, v) = OPT(n-1, v)$ *for every node* v*, then no negative cycles.*

Detecting Negative Cycles

SHANGHAI JIAO TONG UNIVERSITY

Lemma

If $OPT(n, v) = OPT(n-1, v)$ *for every node* v, *then no negative cycles.*

Proof.

The $OPT(n, v)$ values have converged $\Rightarrow$ shortest $v \rightsquigarrow t$ path exists.

Detecting Negative Cycles

SHANGHAI JIAO TONG UNIVERSITY

Lemma

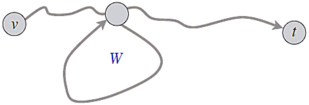
If $OPT(n, v) < OPT(n-1, v)$ for some node v, then (any) shortest $v \rightsquigarrow t$ path of length $\leq n$ contains a cycle W. Moreover W is a negative cycle.

Detecting Negative Cycles

SHANGHAI JIAO TONG UNIVERSITY

Lemma

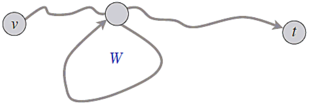
If $OPT(n, v) < OPT(n-1, v)$ *for some node* v, *then (any) shortest* $v \rightsquigarrow t$ *path of length* $\leq n$ *contains a cycle* W. *Moreover* W *is a negative cycle.*



Detecting Negative Cycles

Lemma

If $OPT(n, v) < OPT(n-1, v)$ *for some node* v, *then (any) shortest* $v \rightsquigarrow t$ *path of length* $\leq n$ *contains a cycle* W. *Moreover* W *is a negative cycle.*



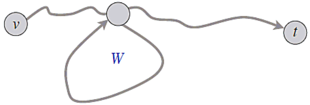
Proof. [by contradiction]

Detecting Negative Cycles

SHANGHAI JIAO TONG UNIVERSITY

Lemma

If $OPT(n,v) < OPT(n-1,v)$ for some node v, then (any) shortest $v \rightsquigarrow t$ path of length $\leq n$ contains a cycle W. Moreover W is a negative cycle.



Proof. [by contradiction]

- Since $OPT(n,v) < OPT(n-1,v)$, we know that shortest $v \rightsquigarrow t$ path P has exactly n edges.
- By pigeonhole principle, the path P must contain a repeated node x.
- Let W be any cycle in P.
- Deleting W yields a $v \rightsquigarrow t$ path with $< n$ edges $\Rightarrow$ W is a negative cycle.

Finding a Negative Cycle

Finding a Negative Cycle

DIY!

Single-Source Shortest Paths with Negative Weights

year	worst case	discovered by
1955	$O\left(n^4\right)$	Shimbel
1956	$O\left(mn^2W\right)$	Ford
1958	$O(mn)$	Bellman, Moore
1983	$O\left(n^{3/4}m\log W\right)$	Gabow
1989	$O\left(mn^{1/2}\log(nW)\right)$	Gabow-Tarjan
1993	$O\left(mn^{1/2}\log W\right)$	Goldberg
2005	$O\left(n^{2.38}W\right)$	Sankowsi, Yuster-Zwick
2016	$\tilde{O}\left(n^{10\backslash 7}\log W\right)$	Cohen-Madry-Sankowski-Vladu
20xx	???	

series single-source shortest paths with weights between $-W$ and W

Shortest Reliable Paths

Shortest Reliable Paths

In a network, even if edge lengths faithfully reflect transmission delays, there may be other considerations involved in choosing a path.

Shortest Reliable Paths

In a network, even if edge lengths faithfully reflect transmission delays, there may be other considerations involved in choosing a path.

For instance, each extra edge in the path might be an extra "hop" fraught with uncertainties and dangers of packet loss.

Shortest Reliable Paths

In a network, even if edge lengths faithfully reflect transmission delays, there may be other considerations involved in choosing a path.

For instance, each extra edge in the path might be an extra "hop" fraught with uncertainties and dangers of packet loss.

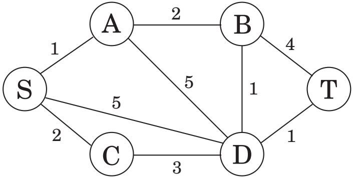
We would like to avoid paths with too many edges.

Shortest Reliable Paths

In a network, even if edge lengths faithfully reflect transmission delays, there may be other considerations involved in choosing a path.

For instance, each extra edge in the path might be an extra "hop" fraught with uncertainties and dangers of packet loss.

We would like to avoid paths with too many edges.



Shortest Reliable Paths

Suppose then that we are given a graph G with lengths on the edges, along with two nodes s and t and an integer k,

Shortest Reliable Paths

Suppose then that we are given a graph G with lengths on the edges, along with two nodes s and t and an integer k, and we want the shortest path from s to t that uses at most k edges.

Shortest Reliable Paths

Suppose then that we are given a graph G with lengths on the edges, along with two nodes s and t and an integer k, and we want the shortest path from s to t that uses at most k edges.

Dynamic programming will work!

Dynamic Programming

Dynamic Programming

For each vertex v and each integer $i \leq k$, let

$dist(v, i) =$ the length of the shortest path from s to v that uses i edges

Dynamic Programming

Shanghai Jiao Tong University

For each vertex v and each integer $i \leq k$, let
$dist(v, i) =$ the length of the shortest path from s to v that uses i edges

The starting values $dist(v, 0)$ are ∞ for all vertices except s, for which it is 0.

Dynamic Programming

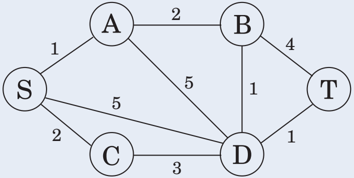
For each vertex v and each integer $i \leq k$, let
$dist(v, i) =$ the length of the shortest path from s to v that uses i edges

The starting values $dist(v, 0)$ are ∞ for all vertices except s, for which it is 0.

$$dist(v, i) = \min_{(u,v) \in E} \{dist(u, i-1) + l(u, v)\}$$

Shortest Reliable Paths

Find out the shortest reliable path from S to T, when $k = 3$.



All-Pairs Shortest Path

All-Pairs Shortest Path

What if we want to find the shortest path not just between s and t but between all pairs of vertices?

All-Pairs Shortest Path

What if we want to find the shortest path not just between s and t but between all pairs of vertices?

One approach would be to execute Bellman-Ford-Moore algorithm $|V|$ times, once for each starting node.

All-Pairs Shortest Path

What if we want to find the shortest path not just between s and t but between all pairs of vertices?

One approach would be to execute Bellman-Ford-Moore algorithm $|V|$ times, once for each starting node.

The total running time would then be $O(|V|^2|E|)$.

We'll now see a better alternative, the $O(|V|^3)$, named Floyd-Warshall algorithm.

The Subproblems

Number the vertices in V as $\{1, 2, \ldots, n\}$,

The Subproblems

SHANGHAI JIAO TONG UNIVERSITY

Number the vertices in V as $\{1, 2, \ldots, n\}$, and let

$dist(i, j, k)$ = the length of the shortest path from i to j in which only nodes $\{1, 2, \ldots, k\}$ can be used as intermediates.

The Subproblems

Shanghai Jiao Tong University

Number the vertices in V as $\{1, 2, \ldots, n\}$, and let

$dist(i, j, k)$ = the length of the shortest path from i to j in which only nodes $\{1, 2, \ldots, k\}$ can be used as intermediates.

Initially, $dist(i, j, 0)$ is the length of the direct edge between i and j, if it exists, and is ∞ otherwise.

The Subproblems

SHANGHAI JIAO TONG UNIVERSITY

Number the vertices in V as $\{1, 2, \ldots, n\}$, and let

$dist(i, j, k)$ = the length of the shortest path from i to j in which only nodes $\{1, 2, \ldots, k\}$ can be used as intermediates.

Initially, $dist(i, j, 0)$ is the length of the direct edge between i and j, if it exists, and is ∞ otherwise.

For $k \geq 1$

$$dist(i, j, k) = \min\{dist(i, j, k-1), dist(i, k, k-1) + dist(k, j, k-1)\}$$

The Program


```
for i = 1 to n do
    for j = 1 to n do
        dist(i, j, 0) = ∞;
    end
end
for all (i, j) ∈ E do
    dist(i, j, 0) = l(i, j);
end
for k = 1 to n do
    for i = 1 to n do
        for j = 1 to n do
            dist(i, j, k) = min{dist(i, j, k − 1), dist(i, k, k − 1) + dist(k, j, k − 1)};
        end
    end
end
```

Referred Materials

Referred Materials

- Content of this lecture comes from Section 6.8 and 6.10 in [KT05], and Section 6.6 in [DPV07].