3-8 Cool? We are APSP Algorithms.

Hengfeng Wei

hfwei@nju.edu.cn

November 19, 2018

Please Help Me Out Here.

Definition (Shortest Path)

$$G = (V, E, w) : \text{ weighted digraph}$$

$$\delta(u, v) = \begin{cases} \min \left\{ w(p) : u \overset{p}{\rightsquigarrow} v \right\} & \text{if } u \rightsquigarrow v \\ \infty & \text{o.w.} \end{cases}$$

Path *vs.* Simple path

Definition (Shortest Path)

$$G = (V, E, w): \text{ weighted digraph}$$

$$\delta(u, v) = \begin{cases} \min\left\{ w(p) : u \overset{p}{\rightsquigarrow} v \right\} & \text{if } u \rightsquigarrow v \\ \infty & \text{o.w.} \end{cases}$$

Path *vs.* Simple path

Shortest-path Problem *vs.* Longest-path Problem

Digraph *vs.* Undirected Graph

Single Source Digraph

Shortest-path Problem *vs.* Longest-path Problem

Single Source Digraph

Shortest-path Problem *vs.* Longest-path Problem

$$\textsf{SP} \text{ in } G \iff \textsf{LP} \text{ in } -G$$

Single Source Digraph

Shortest-path Problem *vs.* Longest-path Problem

$$\mathsf{SP} \text{ in } G \iff \mathsf{LP} \text{ in } -G$$

$O(VE)$ (Bellman-Ford) *vs.* NP-hard (I just told you.)

Single Source Digraph

Shortest-path Problem *vs.* Longest-path Problem

SP in G $\iff$ LP in $-G$

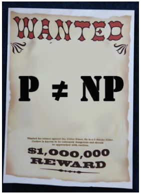
$O(VE)$ (Bellman-Ford) *vs.* NP-hard (I just told you.)

Single Source   Digraph

Shortest-path Problem *vs.* Longest-path Problem

$$\text{SP in } G \iff \text{LP in } -G$$

$O(VE)$ (Bellman-Ford) *vs.* NP-hard (I just told you.)

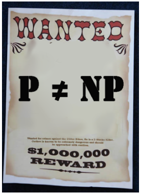


Single Source Digraph

Shortest-path Problem *vs.* Longest-path Problem

$$\text{SP in } G \iff \text{LP in } -G$$

$O(VE)$ (Bellman-Ford) *vs.* NP-hard (I just told you.)



Definition (Shortest Path)

$$G = (V, E, w): \text{ weighted digraph}$$

$$\delta(u,v) = \begin{cases} \min \{ w(p) : \underbrace{u \overset{p}{\leadsto} v}_{\text{Path}} \} & \text{if } u \leadsto v \\ \infty & \text{o.w.} \end{cases}$$

Definition (Shortest Path)

$$G = (V, E, w): \text{ weighted digraph}$$

$$\delta(u,v) = \begin{cases} \min \{ w(p) : \underbrace{u \overset{p}{\leadsto} v}_{\text{Path}} \} & \text{if } u \leadsto v \\ \infty & \text{o.w.} \end{cases}$$

Q : How does Bellman-Ford Handle with Negative-weight Cycles?

Definition (Shortest Path)

$$G = (V, E, w): \text{ weighted digraph}$$

$$\delta(u,v) = \begin{cases} \min\left\{w(p) : \underbrace{u \overset{p}{\rightsquigarrow} v}_{\text{Path}}\right\} & \text{if } u \rightsquigarrow v \\ \infty & \text{o.w.} \end{cases}$$

Q : How does Bellman-Ford Handle with Negative-weight Cycles?

A : Report it and Treat Shortest Path as "Undefined".

Definition (Shortest Path)

$$G = (V, E, w): \ \text{weighted digraph}$$

$$\delta(u, v) = \begin{cases} \min\left\{w(p) : \underbrace{u \overset{p}{\leadsto} v}_{\text{Path}}\right\} & \text{if } u \leadsto v \\ \infty & \text{o.w.} \end{cases}$$

Q : How does Bellman-Ford Handle with Negative-weight Cycles?

A : Report it and Treat Shortest Path as "Undefined".

$$O(VE)$$

Definition (Shortest Simple Path)

$$G = (V, E, w): \text{ weighted digraph}$$

$$\delta(u, v) = \begin{cases} \min \{ w(p) : \underbrace{u \overset{p}{\rightsquigarrow} v}_{\text{Simple Path}} \} & \text{if } u \rightsquigarrow v \\ \infty & \text{o.w.} \end{cases}$$

Definition (Shortest Simple Path)

$$G = (V, E, w) : \text{ weighted digraph}$$

$$\delta(u, v) = \begin{cases} \min \{ w(p) : \underbrace{u \overset{p}{\leadsto} v}_{\text{Simple Path}} \} & \text{if } u \leadsto v \\ \infty & \text{o.w.} \end{cases}$$

Q : How Should an Algorithm for Shortest Simple Path Problem Handle with Negative-weight Cycles?

Definition (Shortest Simple Path)

$$G = (V, E, w) : \text{ weighted digraph}$$

$$\delta(u, v) = \begin{cases} \min \left\{ w(p) : \underbrace{u \overset{p}{\leadsto} v}_{\text{Simple Path}} \right\} & \text{if } u \leadsto v \\ \infty & \text{o.w.} \end{cases}$$

Q : How Should an Algorithm for Shortest Simple Path Problem Handle with Negative-weight Cycles?

A : Still to Find Shortest Simple Path.

Definition (Shortest Simple Path)

$$G = (V, E, w) : \text{ weighted digraph}$$

$$\delta(u,v) = \begin{cases} \min\left\{w(p) : \underbrace{u \overset{p}{\leadsto} v}_{\text{Simple Path}}\right\} & \text{if } u \leadsto v \\ \infty & \text{o.w.} \end{cases}$$

Q : How Should an Algorithm for Shortest Simple Path Problem Handle with Negative-weight Cycles?

A : Still to Find Shortest Simple Path.

NP-hard

Shortest Path Problem

Shortest Simple Path Problem

Shortest Path Problem

Longest Path Problem

Shortest Simple Path Problem

Longest Simple Path Problem

Shortest Path Problem	$\Longleftrightarrow$	Longest Path Problem
Shortest Simple Path Problem	$\Longleftrightarrow$	Longest Simple Path Problem

Single Source Undirected Graph

Single Source Undirected Graph

Negative-weight edges allowed (Why?)

Single Source Undirected Graph

Negative-weight edges allowed (Why?)

Simple path (Why?)

Single Source Undirected Graph

Negative-weight edges allowed (Why?)

Simple path (Why?)

No negative-weight cycles (o.w., NP-hard)

Single Source Undirected Graph

Negative-weight edges allowed (Why?)

Simple path (Why?)

No negative-weight cycles (o.w., NP-hard)

Single-source $s \rightsquigarrow$ Single-target t

Shortest Path Algorithms

Luis Goddyn, Math 408

Given an edge weighted graph (G, d), $d : E(G) \to \mathbb{Q}$ and two vertices $s, t \in V(G)$, the *Shortest Path Problem* is to find an s, t-path P whose total weight is as small as possible. Here, G may be either directed or undirected. A path in a graph is a sequence $v_0 e_1, v_1, \ldots, v_k$ of vertices and edges such that no vertex or edge appears twice, and e_i joins v_{i-1} to v_i. If G is directed, then e_i should be oriented from v_{i-1} to v_i.

Minimum-weight Perfect Matching

Shortest Path Algorithms

Luis Goddyn, Math 408

Given an edge weighted graph (G, d), $d : E(G) \rightarrow \mathbb{Q}$ and two vertices $s, t \in V(G)$, the *Shortest Path Problem* is to find an s, t-path P whose total weight is as small as possible. Here, G may be either directed or undirected. A path in a graph is a sequence $v_0 e_1, v_1, \ldots, v_k$ of vertices and edges such that no vertex or edge appears twice, and e_i joins v_{i-1} to v_i. If G is directed, then e_i should be oriented from v_{i-1} to v_i.

Minimum-weight Perfect Matching

We leave it to the reader to $\cdots$

Shortest Path Algorithms

Luis Goddyn, Math 408

Given an edge weighted graph (G, d), $d : E(G) \to \mathbb{Q}$ and two vertices $s, t \in V(G)$, the *Shortest Path Problem* is to find an s, t-path P whose total weight is as small as possible. Here, G may be either directed or undirected. A path in a graph is a sequence $v_0e_1, v_1, \ldots, v_k$ of vertices and edges such that no vertex or edge appears twice, and e_i joins v_{i-1} to v_i. If G is directed, then e_i should be oriented from v_{i-1} to v_i.

Minimum-weight Perfect Matching

We leave it to the reader to $\cdots$

It is easy to check that $\cdots$

Shortest Path Algorithms

Luis Goddyn, Math 408

Given an edge weighted graph (G, d), $d : E(G) \to \mathbb{Q}$ and two vertices $s, t \in V(G)$, the *Shortest Path Problem* is to find an s, t-path P whose total weight is as small as possible. Here, G may be either directed or undirected. A path in a graph is a sequence $v_0 e_1, v_1, \ldots, v_k$ of vertices and edges such that no vertex or edge appears twice, and e_i joins v_{i-1} to v_i. If G is directed, then e_i should be oriented from v_{i-1} to v_i.

Minimum-weight Perfect Matching

We leave it to the reader to $\cdots$

It is easy to check that $\cdots$

Why? This will be a homework question.

Shortest Path Algorithms

Luis Goddyn, Math 408

Given an edge weighted graph (G, d), $d : E(G) \to \mathbb{Q}$ and two vertices $s, t \in V(G)$, the *Shortest Path Problem* is to find an s, t-path P whose total weight is as small as possible. Here, G may be either directed or undirected. A path in a graph is a sequence $v_0 e_1, v_1, \ldots, v_k$ of vertices and edges such that no vertex or edge appears twice, and e_i joins v_{i-1} to v_i. If G is directed, then e_i should be oriented from v_{i-1} to v_i.

Minimum-weight Perfect Matching

We leave it to the reader to $\cdots$

It is easy to check that $\cdots$

Why? This will be a homework question.

And Errors.

INTERESTED?

let's talk.

Robert W. Floyd (1936–1901)

For having a clear influence on methodologies for the creation of efficient and reliable software, and for helping to found the following important subfields of computer science:

the theory of parsing, the semantics of programming languages, automatic program verification, automatic program synthesis, and analysis of algorithms

— *Turing Award*, 1978

$$\boxed{d_{ij}^{(k)} :}$$

the weight of a shortest path from i to j

for which all *intermediate* vertices are in $\{1, 2, \cdots, k\}$

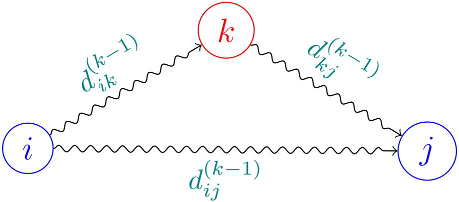
$$\boxed{d_{ij}^{(k)} :}$$

the weight of a shortest path from i to j

for which all *intermediate* vertices are in $\{1, 2, \cdots, k\}$

$$\boxed{D^{(n)} \triangleq \left(d_{ij}^{(n)}\right)}$$

$$k \in \mathrm{SP}_{ij}^{(k)}?$$



$$d_{ij}^{(k)} = \begin{cases} w_{ij} & k = 0 \\ \min\left\{d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right\} & k \geq 1 \end{cases}$$

procedure FLOYD-WARSHALL(W)

$D^{(0)} = W$

for $k \gets 1$ **to** n **do**

$D^{(k)} \triangleq \left(d_{ij}^{(k)}\right) \gets$ a new $n \times n$ matrix

for $i \gets 1$ **to** n **do**

for $j \gets 1$ **to** n **do**

$d_{ij}^{(k)} = \min \left\{ d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)} \right\}$

return $D^{(n)}$

procedure FLOYD-WARSHALL(W)

$\quad D^{(0)} = W$

$\quad$ **for** $k \leftarrow 1$ **to** n **do**

$\quad\quad D^{(k)} \triangleq \left(d_{ij}^{(k)}\right) \leftarrow$ a new $n \times n$ matrix

$\quad\quad$ **for** $i \leftarrow 1$ **to** n **do**

$\quad\quad\quad$ **for** $j \leftarrow 1$ **to** n **do**

$\quad\quad\quad\quad d_{ij}^{(k)} = \min\left\{d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right\}$

$\quad$ **return** $D^{(n)}$

$$\text{Space} : \Theta(n^3)$$

```
procedure Floyd-Warshall(W)
    D^(0) = W

    for k ← 1 to n do
        D^(k) ≜ (d_ij^(k)) ← a new n × n matrix

        for i ← 1 to n do
            for j ← 1 to n do
                d_ij^(k) = min { d_ij^(k-1), d_ik^(k-1) + d_kj^(k-1) }

    return D^(n)
```

$$\text{Space}: \Theta(n^3) \implies \Theta(n^2)$$

Floyd-Warshall Made Simple (Problem 25.2-4)

procedure Floyd-Warshall-Simplified(W)
    $D = W$
    for $k \leftarrow 1$ **to** n **do**
        for $i \leftarrow 1$ **to** n **do**
            for $j \leftarrow 1$ **to** n **do**
                $d_{ij} = \min \left\{ d_{ij}, d_{ik} + d_{kj} \right\}$
    return D

Floyd-Warshall Made Simple (Problem 25.2-4)

---

procedure Floyd-Warshall-Simplified(W)
    $D = W$
    for $k \leftarrow 1$ **to** n **do**
        for $i \leftarrow 1$ **to** n **do**
            for $j \leftarrow 1$ **to** n **do**
                $d_{ij} = \min \left\{ d_{ij}, d_{ik} + d_{kj} \right\}$
    return D

$$D^{(k-1)} \implies D^{(k)}$$

Floyd-Warshall Made Simple (Problem 25.2-4)

```
procedure Floyd-Warshall-Simplified(W)
    D = W
    for k ← 1 to n do
        for i ← 1 to n do
            for j ← 1 to n do
                d_ij = min{d_ij, d_ik + d_kj}
    return D
```

$$D^{(k-1)} \implies D^{(k)} \qquad D \implies D$$

FLOYD-WARSHALL Made Simple (Problem 25.2-4)

```
procedure FLOYD-WARSHALL-SIMPLIFIED(W)
    D = W
    for k ← 1 to n do
        for i ← 1 to n do
            for j ← 1 to n do
                d_ij = min{d_ij, d_ik + d_kj}
    return D
```

$$D^{(k-1)} \Longrightarrow D^{(k)} \qquad D \Longrightarrow D$$

"Decrease" does no harm.

Negative-weight Cycle Detection (Problem 25.2-6)

To detect negative-weight cycle (NC) using FLOYD-WARSHALL.

Negative-weight Cycle Detection (Problem 25.2-6)

To detect negative-weight cycle (NC) using FLOYD-WARSHALL.

$$\exists i : d_{ii}^{(n)} < 0$$

Proof.

Negative-weight Cycle Detection (Problem 25.2-6)

To detect negative-weight cycle (NC) using FLOYD-WARSHALL.

$$\exists i : d_{ii}^{(n)} < 0$$

Proof.

Negative-weight Cycle Detection (Problem 25.2-6)

To detect negative-weight cycle (NC) using FLOYD-WARSHALL.

$$\exists i : d_{ii}^{(n)} < 0$$

Proof.

Negative-weight Cycle Detection (Problem 25.2-6)

To detect negative-weight cycle (NC) using FLOYD-WARSHALL.

$$\exists i : d_{ii}^{(n)} < 0$$

Proof.

$$\exists i : d_{ii}^{(n)} < 0 \iff \exists\ \mathsf{NC} \subseteq G$$

□

$$\exists i : d_{ii}^{(n)} < 0 \iff \exists\, \mathsf{NC} \subseteq G$$

Proof.

$$\exists i : d_{ii}^{(n)} < 0 \iff \exists \, \mathsf{NC} \subseteq G$$

Proof.

“ $\implies$ ”

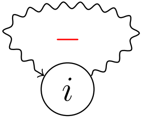
Proof. This is trivial.

$$\exists i : d_{ii}^{(n)} < 0 \iff \exists\ \mathsf{NC} \subseteq G$$

Proof.

" $\implies$ "

" $\impliedby$ "

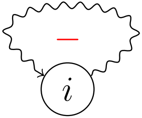


A Cycle

$$\exists i : d_{ii}^{(n)} < 0 \iff \exists\ \mathsf{NC} \subseteq G$$

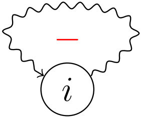
Proof.

“ $\Longrightarrow$ ”



A Cycle

“ $\Longleftarrow$ ”

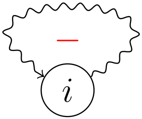


A Simple Cycle

$$\exists i : d_{ii}^{(n)} < 0 \iff \exists\ \mathsf{NC} \subseteq G$$

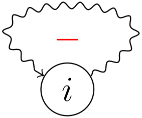
Proof.

"$\implies$"



A Cycle

"$\impliedby$"

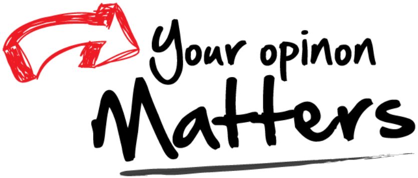


A Simple Cycle

$$d_{ii}^{(n)} < 0$$

□


THANK YOU



Office 302

Mailbox: H016

hfwei@nju.edu.cn