CMPSC 465
Data Structures and Algorithms
Spring 2022

Instructor: Chunhao Wang

Dynamic Programming

Dynamic Programming

All-pair shortest path (Textbook Section 6.6)

All-pair shortest path

Consider $G = (V, E)$ weighted, directed graph without negative cycles

All-pair shortest path

Consider $G = (V, E)$ weighted, directed graph without negative cycles

How to compute the $\mathrm{shortest\_path}(u, v)$?

All-pair shortest path

Consider $G = (V, E)$ weighted, directed graph without negative cycles

How to compute the $\mathrm{shortest\_path}(u, v)$?

Recall Bellman-Ford: $\mathrm{shortest\_path}(u, v)$ for fixed u, all v takes $O(|V| \cdot |E|)$ time

All-pair shortest path

Consider $G = (V, E)$ weighted, directed graph without negative cycles

How to compute the $\mathrm{shortest\_path}(u, v)$?

Recall Bellman-Ford: $\mathrm{shortest\_path}(u, v)$ for fixed u, all v takes $O(|V| \cdot |E|)$ time

If for all u, v, APSP takes $O(|V|^2|E|)$ time

All-pair shortest path

Consider $G = (V, E)$ weighted, directed graph without negative cycles

How to compute the $\text{shortest\_path}(u, v)$?

Recall Bellman-Ford: $\text{shortest\_path}(u, v)$ for fixed u, all v takes $O(|V| \cdot |E|)$ time

If for all u, v, APSP takes $O(|V|^2|E|)$ time
When $|E| = O(|V|^2)$, its running time becomes $O(|V|^4)$

All-pair shortest path

Consider $G = (V, E)$ weighted, directed graph without negative cycles

How to compute the $\text{shortest\_path}(u, v)$?

Recall Bellman-Ford: $\text{shortest\_path}(u, v)$ for fixed u, all v takes $O(|V| \cdot |E|)$ time

If for all u, v, APSP takes $O(|V|^2|E|)$ time
When $|E| = O(|V|^2)$, its running time becomes $O(|V|^4)$

Rethink this problem using DP.

Subproblem

WLOG, index the vertices as $V = \{1, 2, \ldots, n\}$

Subproblem

WLOG, index the vertices as $V = \{1, 2, \ldots, n\}$

Subproblem: find the shortest path $u \to v$ using intermediate vertices from $\{1, \ldots, k\} \subseteq V$.

Subproblem

WLOG, index the vertices as $V = \{1, 2, \ldots, n\}$

Subproblem: find the shortest path $u \to v$ using intermediate vertices from $\{1, \ldots, k\} \subseteq V$. Denote it by $\mathrm{sp}(u, v, k)$

Subproblem

WLOG, index the vertices as $V = \{1, 2, \ldots, n\}$

Subproblem: find the shortest path $u \to v$ using intermediate vertices from $\{1, \ldots, k\} \subseteq V$. Denote it by $\mathrm{sp}(u, v, k)$

Optimal solution: the entries $\mathrm{sp}(u, v, n)$ for all u, v

Subproblem

WLOG, index the vertices as $V = \{1, 2, \ldots, n\}$

Subproblem: find the shortest path $u \to v$ using intermediate vertices from $\{1, \ldots, k\} \subseteq V$. Denote it by $\mathrm{sp}(u, v, k)$

Optimal solution: the entries $\mathrm{sp}(u, v, n)$ for all u, v

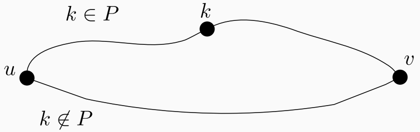
To find out the recurrence relation, we need to relate $\mathrm{sp}(u, v, k)$ to smaller subproblems $\mathrm{sp}(u, v, k-1)$

Recurrence

Suppose $\mathrm{sp}(\mathrm{u}, \mathrm{v}, \mathrm{k}) = P$

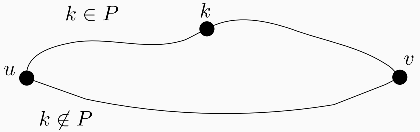
Recurrence

Suppose $\mathrm{sp}(\mathrm{u}, \mathrm{v}, \mathrm{k}) = P$



Recurrence

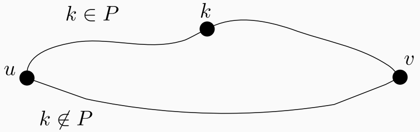
Suppose $\mathrm{sp}(\mathrm{u}, \mathrm{v}, \mathrm{k}) = P$



- if $k \notin P$, then $\mathrm{sp}(u, v, k) = \mathrm{sp}(u, v, k - 1)$

Recurrence

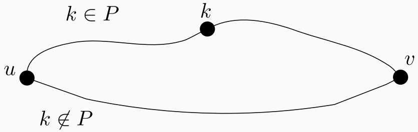
Suppose $\mathrm{sp}(\mathrm{u}, \mathrm{v}, \mathrm{k}) = P$



- if $k \notin P$, then $\mathrm{sp}(u, v, k) = \mathrm{sp}(u, v, k-1)$
- if $k \in P$, then consider

Recurrence

Suppose $\mathrm{sp(u, v, k)} = P$

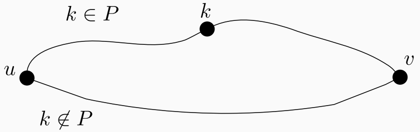


- if $k \notin P$, then $\mathrm{sp}(u, v, k) = \mathrm{sp}(u, v, k-1)$
- if $k \in P$, then consider

$$P : \quad u \xrightarrow{P_1} k \xrightarrow{P_2} v$$

Recurrence

Suppose $\mathrm{sp}(\mathrm{u}, \mathrm{v}, \mathrm{k}) = P$



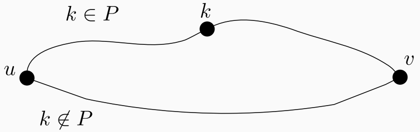
- if $k \notin P$, then $\mathrm{sp}(u, v, k) = \mathrm{sp}(u, v, k-1)$
- if $k \in P$, then consider

$$P : \quad u \xrightarrow{P_1} k \xrightarrow{P_2} v$$

P_1, P_2 are paths whose intermediate vertices are from $\{1, \ldots, k-1\}$.

Recurrence

Suppose $\mathrm{sp(u, v, k)} = P$



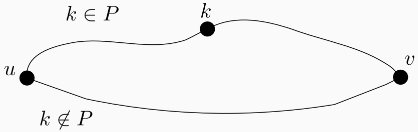
- if $k \notin P$, then $\mathrm{sp}(u, v, k) = \mathrm{sp}(u, v, k-1)$
- if $k \in P$, then consider

$$P: \quad u \xrightarrow{P_1} k \xrightarrow{P_2} v$$

  P_1, P_2 are paths whose intermediate vertices are from $\{1, \ldots, k-1\}$. Because there's no negative cycles, there's no repeated vertices in shortest path

Recurrence

Suppose $\mathrm{sp}(\mathrm{u}, \mathrm{v}, \mathrm{k}) = P$



- if $k \notin P$, then $\mathrm{sp}(u, v, k) = \mathrm{sp}(u, v, k-1)$
- if $k \in P$, then consider

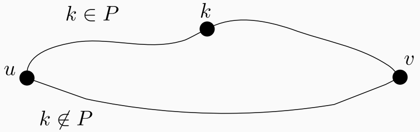
$$P: \quad u \xrightarrow{P_1} k \xrightarrow{P_2} v$$

P_1, P_2 are paths whose intermediate vertices are from $\{1, \ldots, k-1\}$. Because there's no negative cycles, there's no repeated vertices in shortest path

$$\text{Hence,} \quad P_1 = \mathrm{sp}(u, k, k-1), P_2 = \mathrm{sp}(k, v, k-1)$$

Recurrence

Suppose $\mathrm{sp(u, v, k)} = P$



- if $k \notin P$, then $\mathrm{sp}(u, v, k) = \mathrm{sp}(u, v, k-1)$
- if $k \in P$, then consider

$$P: \quad u \xrightarrow{P_1} k \xrightarrow{P_2} v$$

P_1, P_2 are paths whose intermediate vertices are from $\{1, \ldots, k-1\}$. Because there's no negative cycles, there's no repeated vertices in shortest path

$$\text{Hence,} \quad P_1 = \mathrm{sp}(u, k, k-1), P_2 = \mathrm{sp}(k, v, k-1)$$

Using k is better if

$$|\mathrm{sp}(i, k, k-1)| + |\mathrm{sp}(k, v, k-1)| \leq |\mathrm{sp}(i, v, k-1)|$$

Dynamic programming

Let $\mathrm{dist}(u, v, k) = |\mathrm{sp}(u, v, k)|$

Dynamic programming

Let $\mathrm{dist}(u, v, k) = |\mathrm{sp}(u, v, k)|$

- **Recurrence:**

$$\mathrm{dist}(u, v, k) = \min\{\mathrm{dist}(u, v, k-1), \mathrm{dist}(u, k, k-1) + \mathrm{dist}(k, v, k-1)\}$$

Dynamic programming

Let $\mathrm{dist}(u, v, k) = |\mathrm{sp}(u, v, k)|$

- **Recurrence:**

  $$\mathrm{dist}(u, v, k) = \min\{\mathrm{dist}(u, v, k-1), \mathrm{dist}(u, k, k-1) + \mathrm{dist}(k, v, k-1)\}$$

- **Optimal solution:** $\mathrm{dist}(\cdot, \cdot, n)$

Dynamic programming

Let $\mathrm{dist}(u, v, k) = |\mathrm{sp}(u, v, k)|$

- **Recurrence:**

$$\mathrm{dist}(u, v, k) = \min\{\mathrm{dist}(u, v, k-1), \mathrm{dist}(u, k, k-1) + \mathrm{dist}(k, v, k-1)\}$$

- **Optimal solution:** $\mathrm{dist}(\cdot, \cdot, n)$
- **Base case:**

$$\mathrm{dist}(u, v, 0) = \begin{cases} w_{u,v} & \text{if } (u, v) \in E \\ \infty & \text{otherwise} \end{cases}$$

Pseudocode

The Floyd-Warshall algorithm:

Pseudocode

The Floyd-Warshall algorithm:

def FLOYD_WARSHALL(G, w)**:**

Pseudocode

The Floyd-Warshall algorithm:

def $\textsc{Floyd\_Warshall}(G, w)$**:**

    for $u = 1 \ldots n$**:**

Pseudocode

The Floyd-Warshall algorithm:

def $\textsc{Floyd\_Warshall}(G, w)$**:**

    for $u = 1 \ldots n$**:**

        for $v = 1 \ldots n$**:**

Pseudocode

The Floyd-Warshall algorithm:

def $\textsc{Floyd\_Warshall}(G, w)$**:**

- **for** $u = 1 \ldots n$**:**
  - **for** $v = 1 \ldots n$**:**
    - $\mathrm{dist}(u, v, 0) = \begin{cases} w_{u,v} & \text{if } (u, v) \in E \\ \infty & \text{otherwise} \end{cases}$;

Pseudocode

The Floyd-Warshall algorithm:

def $\textsc{Floyd\_Warshall}(G, w)$**:**

    for $u = 1 \ldots n$**:**

        for $v = 1 \ldots n$**:**

            $\mathrm{dist}(u, v, 0) = \begin{cases} w_{u,v} & \text{if } (u, v) \in E \\ \infty & \text{otherwise} \end{cases};$

    for $k = 1 \ldots n$**:**

Pseudocode

The Floyd-Warshall algorithm:

def $\textsc{Floyd\_Warshall}(G, w)$:

    for $u = 1 \dots n$:
        for $v = 1 \dots n$:
            $\mathrm{dist}(u, v, 0) = \begin{cases} w_{u,v} & \text{if } (u, v) \in E \\ \infty & \text{otherwise} \end{cases};$
    for $k = 1 \dots n$:
        for $u = 1 \dots n$:

Pseudocode

The Floyd-Warshall algorithm:

def $\textsc{Floyd\_Warshall}(G, w)$**:**

    for $u = 1 \ldots n$**:**

        for $v = 1 \ldots n$**:**

$$\mathrm{dist}(u, v, 0) = \begin{cases} w_{u,v} & \text{if } (u, v) \in E \\ \infty & \text{otherwise} \end{cases};$$

    for $k = 1 \ldots n$**:**

        for $u = 1 \ldots n$**:**

            for $v = 1 \ldots n$**:**

Pseudocode

The Floyd-Warshall algorithm:

def $\textsc{Floyd\_Warshall}(G, w)$**:**

```
for u = 1 ... n:
    for v = 1 ... n:
        dist(u, v, 0) = { w_{u,v}   if (u, v) ∈ E
                        { ∞         otherwise        ;
for k = 1 ... n:
    for u = 1 ... n:
        for v = 1 ... n:
            dist(u, v, k) =
              min{dist(u, v, k − 1), dist(u, k, k − 1) + dist(k, v, k − 1)};
```

$$\mathrm{dist}(u, v, 0) = \begin{cases} w_{u,v} & \text{if } (u, v) \in E \\ \infty & \text{otherwise} \end{cases};$$

$$\mathrm{dist}(u, v, k) = \min\{\mathrm{dist}(u, v, k-1), \mathrm{dist}(u, k, k-1) + \mathrm{dist}(k, v, k-1)\};$$

Pseudocode

The Floyd-Warshall algorithm:

def $\textsc{Floyd\_Warshall}(G, w)$:

```
for u = 1 ... n:
    for v = 1 ... n:
        dist(u, v, 0) = { w_{u,v}   if (u, v) ∈ E
                        { ∞         otherwise        ;
for k = 1 ... n:
    for u = 1 ... n:
        for v = 1 ... n:
            dist(u, v, k) =
              min{dist(u, v, k − 1), dist(u, k, k − 1) + dist(k, v, k − 1)};
return dist(·, ·, n);
```

$$\mathrm{dist}(u,v,0) = \begin{cases} w_{u,v} & \text{if } (u,v) \in E \\ \infty & \text{otherwise} \end{cases};$$

$$\mathrm{dist}(u,v,k) = \min\{\mathrm{dist}(u,v,k-1), \mathrm{dist}(u,k,k-1) + \mathrm{dist}(k,v,k-1)\};$$

Pseudocode

The Floyd-Warshall algorithm:

def $\textsc{Floyd\_Warshall}(G, w)$**:**

```
for u = 1...n:
    for v = 1...n:
        dist(u, v, 0) = { w_{u,v}  if (u, v) ∈ E
                        { ∞        otherwise ;
for k = 1...n:
    for u = 1...n:
        for v = 1...n:
            dist(u, v, k) =
              min{dist(u, v, k − 1), dist(u, k, k − 1) + dist(k, v, k − 1)};
return dist(·, ·, n);
```

$$\mathrm{dist}(u,v,0) = \begin{cases} w_{u,v} & \text{if } (u,v) \in E \\ \infty & \text{otherwise} \end{cases};$$

$$\mathrm{dist}(u,v,k) = \min\{\mathrm{dist}(u,v,k-1), \mathrm{dist}(u,k,k-1) + \mathrm{dist}(k,v,k-1)\};$$

Running time: $O(n^3) = O(|V|^3)$