# Chapter 9 : Shortest Path Problem (Floyd-Warshall Algorithm) and Transitive Closure (Warshall's Algorithm)

Omar Saadi
omar.saadi@um6p.ma

UM6P - College of Computing

1st year of the engineering degree of UM6P-CS
Academic year 2024-2025

# Outline

# Outline

# Framework

- We have a digraph (or graph) $G = (V, E)$.
- Each edge $(u, v) \in E$ has a weight $w(u, v)$ (that can be negative), but $G$ **has no negative cycles**.

## All Pairs Shortest Path problem

For each pair of vertices $s, t$, we want to solve the problem :

$$\underset{P \text{ a } s,t\text{-path in } G}{\text{minimize}} \; w(P),$$

where $w(P)$ is the weight of the path $P$ given as the sum of the weights of the edges forming it.

# Outline

# Floyd-Warshall Algorithm

**Input :** A digraph (or graph) $G = (V, E)$ with $|V| = n$ vertices.

**Output :** *(If there is no negative cycles)* Distance $d(s, t)$ from each vertex $s$ to each vertex $t$ and a shortest path tree (from each starting vertex $s$) given by identifying the parent $P(s, t)$ of each vertex $t$ in the shortest path from $s$ to $t$.

# Floyd-Warshall Algorithm

**Key ideas :**

- The vertices of the digraph are identified with the integers from 1 to $n$ (i.e. $V = \{1, \cdots, n\}$).
- If $k$ is the biggest vertex appearing on an $s, t$-shortest path, then $d(s,t) = d(s,k) + d(k,t)$ and moreover, the sub-paths from $s$ to $k$ and from $k$ to $t$ only use vertices up to $k-1$ internally.
- For each $k$ going from 1 to $n$, we consider the problem of computing $d_k(s,t)$ which is the smallest weight of an $s, t$-path that only uses vertices $1, \cdots, k$ internally.

# Floyd-Warshall Algorithm

## Floyd-Warshall Algorithm

**Initialization :**

$\forall u \in V, \forall k \in \{0, \cdots, n\}, d_k(u,u) = 0, P_k(u,u) = \text{NONE}$

$\forall u, v \in V, u \neq v, \forall k \in \{1, \cdots, n\}, d_k(u,v) = +\infty, P_k(u,v) = \text{NONE}$

$\forall (u,v) \in E, d_0(u,v) = w(u,v), P_0(u,v) = u$

$\forall (u,v) \notin E, d_0(u,v) = +\infty, P_0(u,v) = \text{NONE}$

**Iterations :**

**for** $k$ from 1 to $n$ **do** :

  **for** $(u,v) \in V \times V$ **do** :

    **if** $d_{k-1}(u,k) + d_{k-1}(k,v) < d_{k-1}(u,v)$ **then**

    $d_k(u,v) = d_{k-1}(u,k) + d_{k-1}(k,v)$ and

    $P_k(u,v) = P_{k-1}(k,v)$,

    **else** $d_k(u,v) = d_{k-1}(u,v)$ and $P_k(u,v) = P_{k-1}(u,v)$.

**Return** $d_n(u,v)$ and $P_n(u,v)$ for all $(u,v) \in V \times V$.

# Correctness

## Proposition

If the digraph $G$ has no negative cycles, then $d_n(s,t) = d(s,t)$ for all couple of vertices $(s,t)$, and backtracking from $t$ to $s$ using the parent list $P_n$ yields a shortest path from $s$ to $t$.

## Proof

We prove by induction over $k \in \{0, \cdots, n\}$, that $d_k(s,t)$ is the minimal length of a path from $s$ to $t$ which is allowed to use internally only the vertices smaller of equal to $k$.

**Base case :** For $k = 0$, we initialize each $d_0(u,v)$ as the edge weight $w(u,v)$ if $(u,v) \in E$, else we set it to $+\infty$, so that $d_0(u,v)$ is exactly the length of a shortest path from $s$ to $t$ that don't use any other vertex internally. $P_0(u,v)$ is also properly initialized.

**Inductive step :** Let $k \in \{1, \cdots, n\}$ and suppose that the property is true up to $k-1$.

## Proof (Cont.)

Let $P$ be a shortest path from $u$ to $v$ using $1, \cdots, k$ as intermediate vertices. We can assume that $P$ is a simple path (since there are no negative cycles). There are two cases :

1. $P$ contains $k$ : In this case, we know that neither the path from $u$ to $k$ nor the path from $k$ to $v$ contains any vertices that are greater than $k-1$. Then, by the induction hypothesis we get $d_{k-1}(u,k) + d_{k-1}(k,v) = w(P) \leq d_{k-1}(u,v)$. Therefore, in the algorithm, $d_k(u,v) = d_{k-1}(u,k) + d_{k-1}(k,v) = w(P)$ and $P_k(u,v) = P_{k-1}(k,v)$ is the predecessor of v in a shortest path from $u$ to $v$ using only vertices between 1 and $k$ internally.
2. $P$ does not contain $k$ : In this case $P$ uses only vertices between 1 and $k-1$, then $w(P) = d_{k-1}(u,v) \leq d_{k-1}(u,k) + d_{k-1}(k,v)$. Therefore, from the algorithm, $d_k(u,v) = d_{k-1}(u,v) = w(P)$ and $P_k(u,v) = P_{k-1}(u,v)$ is an adequate update of the predecessor of $v$.

# Detection of negative cycles

## Proposition

The digraph contains a negative cycle if and only if Floyd-Warshall algorithm returns $d_n$ such that $d_n(u, u) < 0$ for some vertex $u$.

## Proof

If there is a cycle $C$ from $u$ to $u$ of negative weight, then $d_n(u, u)$ will be at most the weight of this cycle $C$, and hence, will be negative. Otherwise, and since $d_0(u, u) = 0$ in the beginning of the algorithm, no update can cause $d_n(u, u)$ to become negative.

# Complexity

**Time Complexity :** The runtime of the Floyd-Warshall algorithm is proportional to the size of the table $\{d_k(u,v)\}_{(k,u,v)\in V^3}$ (which is the same size of the table of predecessors $\{P_k(u,v)\}_{(k,u,v)\in V^3}$).
Indeed, filling each entry of the table $d$ (and also of $P$) only depends on at most two other entries filled in before it. Therefore, the time complexity is :

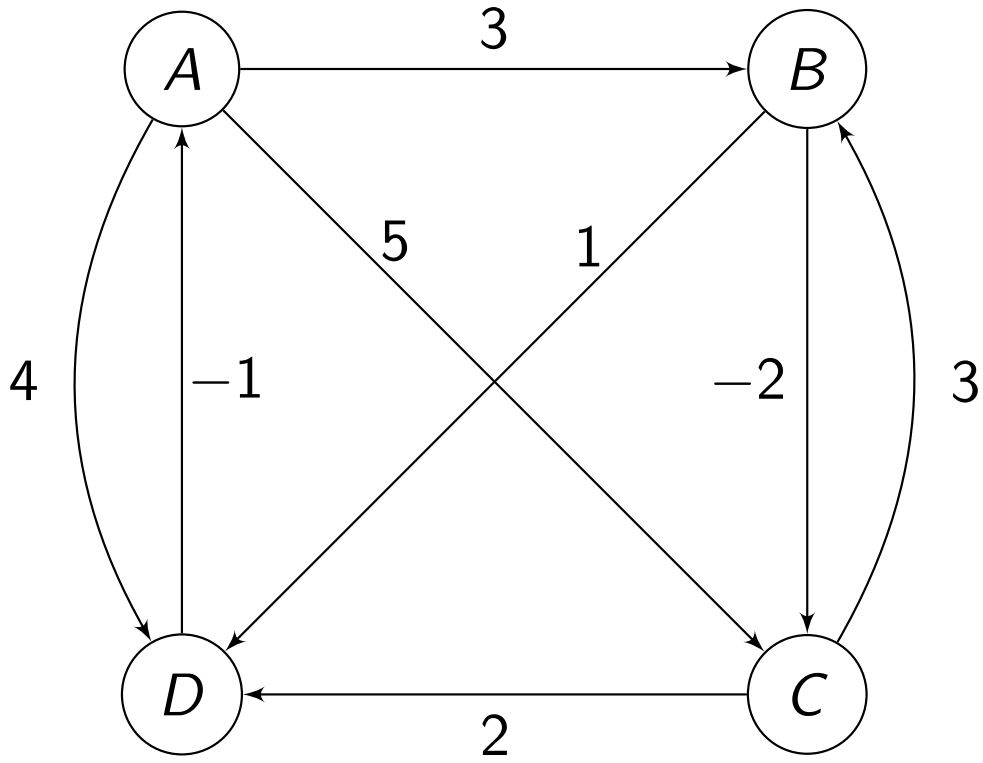
$$O\left(|V|^3\right).$$

**Space Complexity :** Note that we can choose to store only the table of the current iteration $(d_k(u,v), \forall (u,v) \in V^2)$ and the one of the previous iteration $(d_{k-1}(u,v), \forall (u,v) \in V^2)$ instead of storing all the tables $d_k, \forall k \in \{0, \cdots, n\}$.
Therefore, the space complexity is :

$$O\left(|V|^2\right).$$

# Example

In the following digraph, use Floyd-Warshall algorithm to find the distances between all pairs of vertices and the list of parents providing the shortest paths rooted at each vertex.



# Outline

# Transitive Closure of a digraph

## Definition 1

Given a digraph $G = (V, E)$, we call the **transitive closure** of $G$, the subset $T$ of $V \times V$ such that for all $(u, v) \in V \times V$, $(u, v) \in T$ if and only if there is a path from $u$ to $v$ in $G$.

## Definition 2

A digraph $G$ is **transitive** if it satisfies : for all vertices $u, v, w$, if $(u, v) \in E$ and $(v, w) \in E$ then $(u, w) \in E$.

## Lemma

The transitive closure $T$ of a graph $G$ gives the smallest set of edges that needs to be added to make the graph $G$ transitive.

## Proof

Remark that if a path from $u$ to $v$ exists in $G$ then necessarily the edge $(u, v)$ needs to be added to $G$ to make it transitive and that adding all such edges suffices to make the digraph transitive.

# Warshall's Algorithm

**Input :** A digraph $G = (V, E)$ with $|V| = n$ vertices.

**Output :** The set $T$ of couples $(u, v) \in V \times V$, such that there exists a path from $u$ to $v$ in the graph $G$.

**Key ideas :** Similar procedure as Floyd-Warshall algorithm.

- The vertices of the digraph are identified with the integers from 1 to $n$ (i.e. $V = \{1, \cdots, n\}$).
- If $(u, v) \in T$ and $k$ is the biggest vertex appearing on an $u, v$-path, then $(u, k) \in T$ and $(k, v) \in T$ and moreover, the sub-paths from $u$ to $k$ and from $k$ to $v$ only use vertices up to $k - 1$ internally.
- For each $k$ going from 1 to $n$, we consider the problem of identifying if there is a $u$,$v$-path that only uses vertices $1, \cdots, k$ internally. If such path exists we put $c_k(u, v) = 1$, otherwise we put $c_k(u, v) = 0$.

## Warshall's Algorithm

**Initialization :**

$\forall u, v \in V, \forall k \in \{1, \cdots, n\}, c_k(u, v) = 0$

$\forall (u, v) \in E, c_0(u, v) = 1$

$\forall (u, v) \notin E, c_0(u, v) = 0$

**Iterations :**

**for** $k$ from 1 to $n$ **do** :

  **for** $(u, v) \in V \times V$ **do*** :

    **if** $c_{k-1}(u, v) = 1$ or $(c_{k-1}(u, k) = 1$ and $c_{k-1}(k, v) = 1)$

    **then** $c_k(u, v) = 1$,

**Return** $c_n(u, v)$ for all $(u, v) \in V \times V$.

* : This is similar to writing the following (with boolean variables) :

$$c_k(u, v) = c_{k-1}(u, v) \textbf{ or } \left[c_{k-1}(u, k) \textbf{ and } c_{k-1}(k, v)\right].$$

# Correctness and complexity of Warshall's algorithm

**Proposition**

For all $(u, v) \in V \times V$, we have $c_n(u, v) = 1$ if and only if there exists a path from $u$ to $v$ in the digraph $G$, i.e. $T = \{(u, v) \in V^2 \mid c_n(u, v) = 1\}$.

**Proof :** Similar to the proof for Floyd-Warshall algorithm.

Similar to Floyd-Warshall algorithm, we have the following complexities for Warshall's algorithm :

- **Time Complexity :** $O(|V|^3)$.
- **Space Complexity :** $O(|V|^2)$.