Caminos de costo mínimo entre todos los pares de vértices y el algoritmo de Floyd-Warshall

Guillermo Palma

Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información

Plan

1. Descripción del problema
2. Algoritmo de Floyd-Warshall
3. Clausura transitiva

Descripción del problema

Definición del problema a resolver

Definición del problema de los caminos de costo mínimo entre todos los pares de vértices

Dado un digrafo $G = (V, E)$ y una función de costo $w : E \to \mathbb{R}$ que asigna un costo real a cada uno de los lados de G, *el problema de los caminos de costo mínimo entre todos los pares de vértices, en inglés All-Pairs Shortest Paths Problem (APSPP)*, consiste en computar para cada par de vértices $u, v \in V$, el camino de costo mínimo y su valor desde u hasta v.

Sobre el APSPP

- Se presentan los valores de los caminos de costo mínimo en una matriz D, donde una casilla d_{ij} corresponde a la distancia desde el vértice i hasta el j.
- Se puede resolver con el algoritmo de Dijkstra, en $O(|V|^3)$ si se usa un arreglo como cola de prioridad, y en $O(|V||E|\log|V|)$ si usa un min-heap como cola de prioridad.
- Se puede resolver con el algoritmo de Bellman-Ford en $O(|V|^2|E|)$ y si el digrafo es denso $O(|V|^4)$.
- El algoritmo Floyd-Warshall resuelve el problema en $\Theta(|V|^3)$.
- El algoritmo Johnson resuelve el problema en $O(|V|^2\log|V| + |V||E|)$ y es buena opción para digrafos dispersos.
- Para el algoritmo de Floyd-Warshall se usará como representación una matriz de adyacencias.
- Para el algoritmo de Johnson se usará como representación una lista de adyacencias.

Sobre el APSPP

- Se presentan los valores de los caminos de costo mínimo en una matriz D, donde una casilla d_{ij} corresponde a la distancia desde el vértice i hasta el j.
- Se puede resolver con el algoritmo de Dijkstra, en $O(|V|^3)$ si se usa un arreglo como cola de prioridad, y en $O(|V||E|\log|V|)$ si usa un min-heap como cola de prioridad.
- Se puede resolver con el algoritmo de Bellman-Ford en $O(|V|^2|E|)$ y si el digrafo es denso $O(|V|^4)$.
- El algoritmo Floyd-Warshall resuelve el problema en $\Theta(|V|^3)$.
- El algoritmo Johnson resuelve el problema en $O(|V|^2\log|V| + |V||E|)$ y es buena opción para digrafos dispersos.
- Para el algoritmo de Floyd-Warshall se usará como representación una matriz de adyacencias.
- Para el algoritmo de Johnson se usará como representación una lista de adyacencias.

Sobre el APSPP

- Se presentan los valores de los caminos de costo mínimo en una matriz D, donde una casilla d_{ij} corresponde a la distancia desde el vértice i hasta el j.
- Se puede resolver con el algoritmo de Dijkstra, en $O(|V|^3)$ si se usa un arreglo como cola de prioridad, y en $O(|V||E| \log |V|)$ si usa un min-heap como cola de prioridad.
- Se puede resolver con el algoritmo de Bellman-Ford en $O(|V|^2|E|)$ y si el digrafo es denso $O(|V|^4)$.
- El algoritmo Floyd-Warshall resuelve el problema en $\Theta(|V|^3)$.
- El algoritmo Johnson resuelve el problema en $O(|V|^2 \log |V| + |V||E|)$ y es buena opción para digrafos dispersos.
- Para el algoritmo de Floyd-Warshall se usará como representación una matriz de adyacencias.
- Para el algoritmo de Johnson se usará como representación una lista de adyacencias.

Sobre el APSPP

- Se presentan los valores de los caminos de costo mínimo en una matriz D, donde una casilla d_{ij} corresponde a la distancia desde el vértice i hasta el j.
- Se puede resolver con el algoritmo de Dijkstra, en $O(|V|^3)$ si se usa un arreglo como cola de prioridad, y en $O(|V||E|\log|V|)$ si usa un min-heap como cola de prioridad.
- Se puede resolver con el algoritmo de Bellman-Ford en $O(|V|^2|E|)$ y si el digrafo es denso $O(|V|^4)$.
- El algoritmo Floyd-Warshall resuelve el problema en $\Theta(|V|^3)$.
- El algoritmo Johnson resuelve el problema en $O(|V|^2\log|V| + |V||E|)$ y es buena opción para digrafos dispersos.
- Para el algoritmo de Floyd-Warshall se usará como representación una matriz de adyacencias.
- Para el algoritmo de Johnson se usará como representación una lista de adyacencias.

Sobre el APSPP

- Se presentan los valores de los caminos de costo mínimo en una matriz D, donde una casilla d_{ij} corresponde a la distancia desde el vértice i hasta el j.
- Se puede resolver con el algoritmo de Dijkstra, en $O(|V|^3)$ si se usa un arreglo como cola de prioridad, y en $O(|V||E| \log |V|)$ si usa un min-heap como cola de prioridad.
- Se puede resolver con el algoritmo de Bellman-Ford en $O(|V|^2|E|)$ y si el digrafo es denso $O(|V|^4)$.
- El algoritmo Floyd-Warshall resuelve el problema en $\Theta(|V|^3)$.
- El algoritmo Johnson resuelve el problema en $O(|V|^2 \log |V| + |V||E|)$ y es buena opción para digrafos dispersos.
- Para el algoritmo de Floyd-Warshall se usará como representación una matriz de adyacencias.
- Para el algoritmo de Johnson se usará como representación una lista de adyacencias.

Sobre el APSPP

- Se presentan los valores de los caminos de costo mínimo en una matriz D, donde una casilla d_{ij} corresponde a la distancia desde el vértice i hasta el j.
- Se puede resolver con el algoritmo de Dijkstra, en $O(|V|^3)$ si se usa un arreglo como cola de prioridad, y en $O(|V||E|\log|V|)$ si usa un min-heap como cola de prioridad.
- Se puede resolver con el algoritmo de Bellman-Ford en $O(|V|^2|E|)$ y si el digrafo es denso $O(|V|^4)$.
- El algoritmo Floyd-Warshall resuelve el problema en $\Theta(|V|^3)$.
- El algoritmo Johnson resuelve el problema en $O(|V|^2\log|V| + |V||E|)$ y es buena opción para digrafos dispersos.
- Para el algoritmo de Floyd-Warshall se usará como representación una matriz de adyacencias.
- Para el algoritmo de Johnson se usará como representación una lista de adyacencias.

Sobre el APSPP

- Se presentan los valores de los caminos de costo mínimo en una matriz D, donde una casilla d_{ij} corresponde a la distancia desde el vértice i hasta el j.
- Se puede resolver con el algoritmo de Dijkstra, en $O(|V|^3)$ si se usa un arreglo como cola de prioridad, y en $O(|V||E|\log|V|)$ si usa un min-heap como cola de prioridad.
- Se puede resolver con el algoritmo de Bellman-Ford en $O(|V|^2|E|)$ y si el digrafo es denso $O(|V|^4)$.
- El algoritmo Floyd-Warshall resuelve el problema en $\Theta(|V|^3)$.
- El algoritmo Johnson resuelve el problema en $O(|V|^2\log|V| + |V||E|)$ y es buena opción para digrafos dispersos.
- Para el algoritmo de Floyd-Warshall se usará como representación una matriz de adyacencias.
- Para el algoritmo de Johnson se usará como representación una lista de adyacencias.

Notación y convenciones

- Los vértices están numerados $1, 2, \ldots, |V|$.
- Se tiene como entrada una matriz $W = (w_{ij})$ con los costos de arcos del digrafo $G = (V, E)$:

$$w_{ij} = \begin{cases} 0 & \text{si } i = j, \\ \text{el costo del arco } (i,j) & \text{si } i \neq j \wedge (i,j) \in E, \\ \infty & i \neq j \wedge (i,j) \neq E. \end{cases} \quad (1)$$

- El digrafo G puede tener lados con costos negativos, pero se asume que no hay ciclos negativos.
- Matriz de costos $D = (d_{ij})$, donde d_{ij} es el costo del camino de costo mínimo desde i hasta j ($d_{ij} = \delta(i,j)$).
- Matriz de predecesores $\Pi = (\pi_{ij})$, donde $\pi_{ij} = NIL$, si $i = j$ o no hay camino desde i hasta j, en caso contrario $\pi_{ij} =$ *predecesor de j en un camino de camino de costo mínimo desde i*.
- Un superíndice m indica el estado de la matriz en la iteración m, por ejemplo $D^{(3)} = (d_{ij}^{(3)})$, es el estado de la matriz en la iteración 3.

Notación y convenciones

- Los vértices están numerados $1, 2, \ldots, |V|$.
- Se tiene como entrada una matriz $W = (w_{ij})$ con los costos de arcos del digrafo $G = (V, E)$:

$$w_{ij} = \begin{cases} 0 & \text{si } i = j, \\ \text{el costo del arco } (i,j) & \text{si } i \neq j \wedge (i,j) \in E, \\ \infty & i \neq j \wedge (i,j) \neq E. \end{cases} \quad (1)$$

- El digrafo G puede tener lados con costos negativos, pero se asume que no hay ciclos negativos.
- Matriz de costos $D = (d_{ij})$, donde d_{ij} es el costo del camino de costo mínimo desde i hasta j ($d_{ij} = \delta(i,j)$).
- Matriz de predecesores $\Pi = (\pi_{ij})$, donde $\pi_{ij} = NIL$, si $i = j$ o no hay camino desde i hasta j, en caso contrario $\pi_{ij} =$ *predecesor de j en un camino de camino de costo mínimo desde i*.
- Un superíndice m indica el estado de la matriz en la iteración m, por ejemplo $D^{(3)} = (d_{ij}^{(3)})$, es el estado de la matriz en la iteración 3.

Notación y convenciones

- Los vértices están numerados $1, 2, \ldots, |V|$.
- Se tiene como entrada una matriz $W = (w_{ij})$ con los costos de arcos del digrafo $G = (V, E)$:

$$w_{ij} = \begin{cases} 0 & \text{si } i = j, \\ \text{el costo del arco } (i,j) & \text{si } i \neq j \wedge (i,j) \in E, \\ \infty & i \neq j \wedge (i,j) \neq E. \end{cases} \quad (1)$$

- El digrafo G puede tener lados con costos negativos, pero se asume que no hay ciclos negativos.
- Matriz de costos $D = (d_{ij})$, donde d_{ij} es el costo del camino de costo mínimo desde i hasta j ($d_{ij} = \delta(i,j)$).
- Matriz de predecesores $\Pi = (\pi_{ij})$, donde $\pi_{ij} = NIL$, si $i = j$ o no hay camino desde i hasta j, en caso contrario $\pi_{ij} =$ $\textit{predecesor de } j \textit{ en un camino de camino de costo mínimo desde } i$.
- Un superíndice m indica el estado de la matriz en la iteración m, por ejemplo $D^{(3)} = (d_{ij}^{(3)})$, es el estado de la matriz en la iteración 3.

Notación y convenciones

- Los vértices están numerados $1, 2, \ldots, |V|$.
- Se tiene como entrada una matriz $W = (w_{ij})$ con los costos de arcos del digrafo $G = (V, E)$:

$$w_{ij} = \begin{cases} 0 & \text{si } i = j, \\ \text{el costo del arco } (i,j) & \text{si } i \neq j \wedge (i,j) \in E, \\ \infty & i \neq j \wedge (i,j) \neq E. \end{cases} \tag{1}$$

- El digrafo G puede tener lados con costos negativos, pero se asume que no hay ciclos negativos.
- Matriz de costos $D = (d_{ij})$, donde d_{ij} es el costo del camino de costo mínimo desde i hasta j ($d_{ij} = \delta(i,j)$).
- Matriz de predecesores $\Pi = (\pi_{ij})$, donde $\pi_{ij} = NIL$, si $i = j$ o no hay camino desde i hasta j, en caso contrario $\pi_{ij} =$ *predecesor de j en un camino de camino de costo mínimo desde i.*
- Un superíndice m indica el estado de la matriz en la iteración m, por ejemplo $D^{(3)} = (d_{ij}^{(3)})$, es el estado de la matriz en la iteración 3.

Notación y convenciones

- Los vértices están numerados $1, 2, \ldots, |V|$.
- Se tiene como entrada una matriz $W = (w_{ij})$ con los costos de arcos del digrafo $G = (V, E)$:

$$w_{ij} = \begin{cases} 0 & \text{si } i = j, \\ \text{el costo del arco } (i,j) & \text{si } i \neq j \wedge (i,j) \in E, \\ \infty & i \neq j \wedge (i,j) \neq E. \end{cases} \tag{1}$$

- El digrafo G puede tener lados con costos negativos, pero se asume que no hay ciclos negativos.
- Matriz de costos $D = (d_{ij})$, donde d_{ij} es el costo del camino de costo mínimo desde i hasta j ($d_{ij} = \delta(i,j)$).
- Matriz de predecesores $\Pi = (\pi_{ij})$, donde $\pi_{ij} = NIL$, si $i = j$ o no hay camino desde i hasta j, en caso contrario $\pi_{ij} =$ *predecesor de j en un camino de camino de costo mínimo desde i.*
- Un superíndice m indica el estado de la matriz en la iteración m, por ejemplo $D^{(3)} = (d_{ij}^{(3)})$, es el estado de la matriz en la iteración 3.

Notación y convenciones

- Los vértices están numerados $1, 2, \ldots, |V|$.
- Se tiene como entrada una matriz $W = (w_{ij})$ con los costos de arcos del digrafo $G = (V, E)$:

$$w_{ij} = \begin{cases} 0 & \text{si } i = j, \\ \text{el costo del arco } (i,j) & \text{si } i \neq j \land (i,j) \in E, \\ \infty & i \neq j \land (i,j) \neq E. \end{cases} \tag{1}$$

- El digrafo G puede tener lados con costos negativos, pero se asume que no hay ciclos negativos.
- Matriz de costos $D = (d_{ij})$, donde d_{ij} es el costo del camino de costo mínimo desde i hasta j ($d_{ij} = \delta(i,j)$).
- Matriz de predecesores $\Pi = (\pi_{ij})$, donde $\pi_{ij} = NIL$, si $i = j$ o no hay camino desde i hasta j, en caso contrario $\pi_{ij} =$ *predecesor de j en un camino de camino de costo mínimo desde i*.
- Un superíndice m indica el estado de la matriz en la iteración m, por ejemplo $D^{(3)} = (d_{ij}^{(3)})$, es el estado de la matriz en la iteración 3.

Subgrafo predecesor para el APSPP

Definición de subgrafo predecesor

Se define el subgrafo predecesor del digrafo $G = (V, E)$ para cada $i \in V$, como el digrafo $G_{\pi,i} = (V_{\pi,i}, E_{\pi,i})$, donde:

$$V_{\pi,i} = \{j \in V : \pi_{ij} \neq NIL\} \cup \{i\}$$

y

$$E_{\pi,i} = \{(\pi_{ij}, j) : j \in V_{\pi,i} - \{i\}\}$$

Imprimiendo los caminos del APSPP

Procedimiento Print-All-Pairs-Shortest-Path(Π, i, j)

```
inicio
    si i = j entonces
        Imprimir(i)
    si no, si π_ij = NIL entonces
        Imprimir("No existe un camino desde i hasta j")
    en otro caso
        Print-All-Pairs-Shortest-Path(Π, i, π_ij) ;
        Imprimir(j)
```

Algoritmo de Floyd-Warshall

La estructura de un camino de costo mínimo

- Dado un camino simple $p = \langle v_1, v_2, \ldots, v_l \rangle$ un **vértice intermedio** es cualquier vértice de p diferente a v_1 y v_l.
- Sea $V = \{1, 2, \ldots, n\}$ los vértices de G.
- Sea $\{1, 2, \ldots, k\}$ un subconjunto de V.
- Para cada $i, j \in V$ consideramos todos los caminos desde i hasta j, cuyos **vértices intermedios** son $\{1, 2, \ldots, k\}$.
- Sea p el camino de menor costo entre los caminos indicados anteriormente.
- Floyd-Warshall explota la relación entre p y los caminos de costo mínimo entre i y j con los **vértices intermedios** $\{1, 2, \ldots, k-1\}$.
- La relación depende si k es o no es un **vértice intermedio**.

La estructura de un camino de costo mínimo

- Dado un camino simple $p = \langle v_1, v_2, \ldots, v_l \rangle$ un **vértice intermedio** es cualquier vértice de p diferente a v_1 y v_l.
- Sea $V = \{1, 2, \ldots, n\}$ los vértices de G.
- Sea $\{1, 2, \ldots, k\}$ un subconjunto de V.
- Para cada $i, j \in V$ consideramos todos los caminos desde i hasta j, cuyos **vértices intermedios** son $\{1, 2, \ldots, k\}$.
- Sea p el camino de menor costo entre los caminos indicados anteriormente.
- Floyd-Warshall explota la relación entre p y los caminos de costo mínimo entre i y j con los **vértices intermedios** $\{1, 2, \ldots, k-1\}$.
- La relación depende si k es o no es un **vértice intermedio**.

La estructura de un camino de costo mínimo

- Dado un camino simple $p = \langle v_1, v_2, \ldots, v_l \rangle$ un **vértice intermedio** es cualquier vértice de p diferente a v_1 y v_l.
- Sea $V = \{1, 2, \ldots, n\}$ los vértices de G.
- Sea $\{1, 2, \ldots, k\}$ un subconjunto de V.
- Para cada $i, j \in V$ consideramos todos los caminos desde i hasta j, cuyos **vértices intermedios** son $\{1, 2, \ldots, k\}$.
- Sea p el camino de menor costo entre los caminos indicados anteriormente.
- Floyd-Warshall explota la relación entre p y los caminos de costo mínimo entre i y j con los **vértices intermedios** $\{1, 2, \ldots, k-1\}$.
- La relación depende si k es o no es un **vértice intermedio**.

La estructura de un camino de costo mínimo

- Dado un camino simple $p = \langle v_1, v_2, \ldots, v_l \rangle$ un **vértice intermedio** es cualquier vértice de p diferente a v_1 y v_l.
- Sea $V = \{1, 2, \ldots, n\}$ los vértices de G.
- Sea $\{1, 2, \ldots, k\}$ un subconjunto de V.
- Para cada $i, j \in V$ consideramos todos los caminos desde i hasta j, cuyos **vértices intermedios** son $\{1, 2, \ldots, k\}$.
- Sea p el camino de menor costo entre los caminos indicados anteriormente.
- Floyd-Warshall explota la relación entre p y los caminos de costo mínimo entre i y j con los **vértices intermedios** $\{1, 2, \ldots, k-1\}$.
- La relación depende si k es o no es un **vértice intermedio**.

La estructura de un camino de costo mínimo

- Dado un camino simple $p = \langle v_1, v_2, \ldots, v_l \rangle$ un **vértice intermedio** es cualquier vértice de p diferente a v_1 y v_l.
- Sea $V = \{1, 2, \ldots, n\}$ los vértices de G.
- Sea $\{1, 2, \ldots, k\}$ un subconjunto de V.
- Para cada $i, j \in V$ consideramos todos los caminos desde i hasta j, cuyos **vértices intermedios** son $\{1, 2, \ldots, k\}$.
- Sea p el camino de menor costo entre los caminos indicados anteriormente.
- Floyd-Warshall explota la relación entre p y los caminos de costo mínimo entre i y j con los **vértices intermedios** $\{1, 2, \ldots, k-1\}$.
- La relación depende si k es o no es un **vértice intermedio**.

La estructura de un camino de costo mínimo

- Dado un camino simple $p = \langle v_1, v_2, \ldots, v_l \rangle$ un **vértice intermedio** es cualquier vértice de p diferente a v_1 y v_l.
- Sea $V = \{1, 2, \ldots, n\}$ los vértices de G.
- Sea $\{1, 2, \ldots, k\}$ un subconjunto de V.
- Para cada $i, j \in V$ consideramos todos los caminos desde i hasta j, cuyos **vértices intermedios** son $\{1, 2, \ldots, k\}$.
- Sea p el camino de menor costo entre los caminos indicados anteriormente.
- Floyd-Warshall explota la relación entre p y los caminos de costo mínimo entre i y j con los **vértices intermedios** $\{1, 2, \ldots, k-1\}$.
- La relación depende si k es o no es un **vértice intermedio**.

La estructura de un camino de costo mínimo

- Dado un camino simple $p = \langle v_1, v_2, \ldots, v_l \rangle$ un **vértice intermedio** es cualquier vértice de p diferente a v_1 y v_l.
- Sea $V = \{1, 2, \ldots, n\}$ los vértices de G.
- Sea $\{1, 2, \ldots, k\}$ un subconjunto de V.
- Para cada $i, j \in V$ consideramos todos los caminos desde i hasta j, cuyos **vértices intermedios** son $\{1, 2, \ldots, k\}$.
- Sea p el camino de menor costo entre los caminos indicados anteriormente.
- Floyd-Warshall explota la relación entre p y los caminos de costo mínimo entre i y j con los **vértices intermedios** $\{1, 2, \ldots, k-1\}$.
- La relación depende si k es o no es un **vértice intermedio**.

La estructura de un camino de costo mínimo, cont.

Caso 1: *k no es un vértice intermedio de el camino p*:

- Todos los **vértices intermedios** de p están en $\{1, 2, \ldots, k-1\}$.
- El camino de costo mínimo desde i hasta j con los **vértices intermedios** $\{1, 2, \ldots, k-1\}$, es también el camino de costo mínimo desde i hasta j con todos los **vértices intermedios** $\{1, 2, \ldots, k\}$.

La estructura de un camino de costo mínimo, cont.

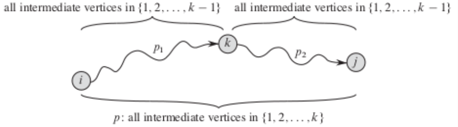
Caso 1: *k no es un vértice intermedio de el camino p*:

- Todos los **vértices intermedios** de p están en $\{1, 2, \ldots, k-1\}$.
- El camino de costo mínimo desde i hasta j con los **vértices intermedios** $\{1, 2, \ldots, k-1\}$, es también el camino de costo mínimo desde i hasta j con todos los **vértices intermedios** $\{1, 2, \ldots, k\}$.

La estructura de un camino de costo mínimo, cont.

Caso 2: *k es un vértice intermedio de el camino p*:

- Se descompone p en $i \overset{p_1}{\rightsquigarrow} k \overset{p_2}{\rightsquigarrow} j$

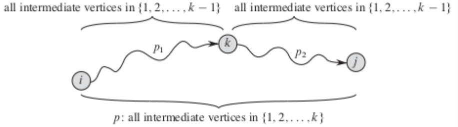


- Por Lema *los subcaminos de los caminos de costo mínimo son caminos de costo mínimo*, p_1 es un camino de costo mínimo.
- p_1 contiene todos los vértices **vértices intermedios** en el conjunto $\{1, 2, \ldots, k\}$.
- Todos los **vértices intermedios** de p_1 están en el conjunto $\{1, 2, \ldots, k-1\}$ porque k es un extremo.
- Por lo tanto, p_1 contiene todos los **vértices intermedios** en el conjunto $\{1, 2, \ldots, k-1\}$.
- p_2 es un camino de costo mínimo desde k hasta j, que contiene todos los **vértices intermedios** en el conjunto $\{1, 2, \ldots, k-1\}$.

La estructura de un camino de costo mínimo, cont.

Caso 2: *k es un vértice intermedio de el camino p*:

- Se descompone p en $i \overset{p_1}{\leadsto} k \overset{p_2}{\leadsto} j$

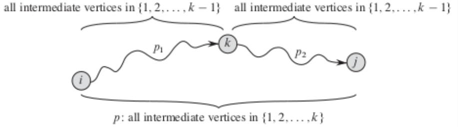


- Por Lema *los subcaminos de los caminos de costo mínimo son caminos de costo mínimo*, p_1 es un camino de costo mínimo.
- p_1 contiene todos los vértices **vértices intermedios** en el conjunto $\{1, 2, \ldots, k\}$.
- Todos los **vértices intermedios** de p_1 están en el conjunto $\{1, 2, \ldots, k-1\}$ porque k es un extremo.
- Por lo tanto, p_1 contiene todos los **vértices intermedios** en el conjunto $\{1, 2, \ldots, k-1\}$.
- p_2 es un camino de costo mínimo desde k hasta j, que contiene todos los **vértices intermedios** en el conjunto $\{1, 2, \ldots, k-1\}$.

La estructura de un camino de costo mínimo, cont.

Caso 2: *k es un vértice intermedio de el camino p*:

- Se descompone p en $i \overset{p_1}{\leadsto} k \overset{p_2}{\leadsto} j$

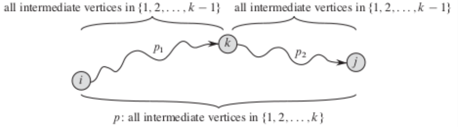


- Por Lema *los subcaminos de los caminos de costo mínimo son caminos de costo mínimo*, p_1 es un camino de costo mínimo.
- p_1 contiene todos los vértices **vértices intermedios** en el conjunto $\{1, 2, \ldots, k\}$.
- Todos los **vértices intermedios** de p_1 están en el conjunto $\{1, 2, \ldots, k-1\}$ porque k es un extremo.
- Por lo tanto, p_1 contiene todos los **vértices intermedios** en el conjunto $\{1, 2, \ldots, k-1\}$.
- p_2 es un camino de costo mínimo desde k hasta j, que contiene todos los **vértices intermedios** en el conjunto $\{1, 2, \ldots, k-1\}$.

La estructura de un camino de costo mínimo, cont.

Caso 2: *k es un vértice intermedio de el camino p*:

- Se descompone p en $i \overset{p_1}{\leadsto} k \overset{p_2}{\leadsto} j$

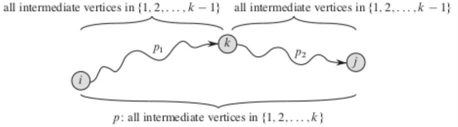


- Por Lema *los subcaminos de los caminos de costo mínimo son caminos de costo mínimo*, p_1 es un camino de costo mínimo.
- p_1 contiene todos los vértices **vértices intermedios** en el conjunto $\{1, 2, \ldots, k\}$.
- Todos los **vértices intermedios** de p_1 están en el conjunto $\{1, 2, \ldots, k - 1\}$ porque k es un extremo.
- Por lo tanto, p_1 contiene todos los **vértices intermedios** en el conjunto $\{1, 2, \ldots, k - 1\}$.
- p_2 es un camino de costo mínimo desde k hasta j, que contiene todos los **vértices intermedios** en el conjunto $\{1, 2, \ldots, k - 1\}$.

La estructura de un camino de costo mínimo, cont.

Caso 2: *k es un vértice intermedio de el camino p*:

- Se descompone p en $i \overset{p_1}{\leadsto} k \overset{p_2}{\leadsto} j$

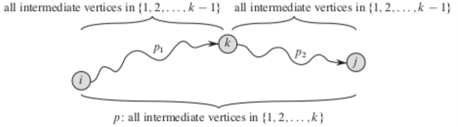


- Por Lema *los subcaminos de los caminos de costo mínimo son caminos de costo mínimo*, p_1 es un camino de costo mínimo.
- p_1 contiene todos los vértices **vértices intermedios** en el conjunto $\{1, 2, \ldots, k\}$.
- Todos los **vértices intermedios** de p_1 están en el conjunto $\{1, 2, \ldots, k-1\}$ porque k es un extremo.
- Por lo tanto, p_1 contiene todos los **vértices intermedios** en el conjunto $\{1, 2, \ldots, k-1\}$.
- p_2 es un camino de costo mínimo desde k hasta j, que contiene todos los **vértices intermedios** en el conjunto $\{1, 2, \ldots, k-1\}$.

La estructura de un camino de costo mínimo, cont.

Caso 2: *k es un vértice intermedio de el camino p*:

- Se descompone p en $i \overset{p_1}{\leadsto} k \overset{p_2}{\leadsto} j$



- Por Lema *los subcaminos de los caminos de costo mínimo son caminos de costo mínimo*, p_1 es un camino de costo mínimo.
- p_1 contiene todos los vértices **vértices intermedios** en el conjunto $\{1, 2, \ldots, k\}$.
- Todos los **vértices intermedios** de p_1 están en el conjunto $\{1, 2, \ldots, k-1\}$ porque k es un extremo.
- Por lo tanto, p_1 contiene todos los **vértices intermedios** en el conjunto $\{1, 2, \ldots, k-1\}$.
- p_2 es un camino de costo mínimo desde k hasta j, que contiene todos los **vértices intermedios** en el conjunto $\{1, 2, \ldots, k-1\}$.

Una solución recursiva al APSPP

- Sea $d_{ij}^{(k)}$ el costo desde del camino de costo mínimo desde i hasta j para el cual todos los **vértices intermedios** están en el conjunto $\{1, 2, \ldots, k\}$.
- Se define $d_{ij}^{(k)}$ como:

$$d_{ij}^{(k)} = \begin{cases} w_{ij} & \text{si } k = 0, \\ \text{mín}(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}) & \text{si } k \geq 1. \end{cases} \quad (2)$$

- Para cualquier camino todos los **vértices intermedios** están en el conjunto $\{1, 2, \ldots, n\}$.
- Entonces, $D^{(n)} = (d_{ij}^{(n)})$, tal que $d_{ij}^{(n)} = \delta(i, j)$ para todo $i, j \in V$.

Una solución recursiva al APSPP

- Sea $d_{ij}^{(k)}$ el costo desde del camino de costo mínimo desde i hasta j para el cual todos los **vértices intermedios** están en el conjunto $\{1, 2, \ldots, k\}$.
- Se define $d_{ij}^{(k)}$ como:

$$d_{ij}^{(k)} = \begin{cases} w_{ij} & \text{si } k = 0, \\ \text{mín}(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}) & \text{si } k \geq 1. \end{cases} \tag{2}$$

- Para cualquier camino todos los **vértices intermedios** están en el conjunto $\{1, 2, \ldots, n\}$.
- Entonces, $D^{(n)} = (d_{ij}^{(n)})$, tal que $d_{ij}^{(n)} = \delta(i, j)$ para todo $i, j \in V$.

Una solución recursiva al APSPP

- Sea $d_{ij}^{(k)}$ el costo desde del camino de costo mínimo desde i hasta j para el cual todos los **vértices intermedios** están en el conjunto $\{1, 2, \ldots, k\}$.
- Se define $d_{ij}^{(k)}$ como:

$$d_{ij}^{(k)} = \begin{cases} w_{ij} & \text{si } k = 0, \\ \text{mín}(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}) & \text{si } k \geq 1. \end{cases} \quad (2)$$

- Para cualquier camino todos los **vértices intermedios** están en el conjunto $\{1, 2, \ldots, n\}$.
- Entonces, $D^{(n)} = (d_{ij}^{(n)})$, tal que $d_{ij}^{(n)} = \delta(i, j)$ para todo $i, j \in V$.

Una solución recursiva al APSPP

- Sea $d_{ij}^{(k)}$ el costo desde del camino de costo mínimo desde i hasta j para el cual todos los **vértices intermedios** están en el conjunto $\{1, 2, \ldots, k\}$.
- Se define $d_{ij}^{(k)}$ como:

$$d_{ij}^{(k)} = \begin{cases} w_{ij} & \text{si } k = 0, \\ \text{mín}(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}) & \text{si } k \geq 1. \end{cases} \tag{2}$$

- Para cualquier camino todos los **vértices intermedios** están en el conjunto $\{1, 2, \ldots, n\}$.
- Entonces, $D^{(n)} = (d_{ij}^{(n)})$, tal que $d_{ij}^{(n)} = \delta(i, j)$ para todo $i, j \in V$.

Función Floyd-Warshall(W)

inicio
- $n \leftarrow W.numDeFilas$;
- $D^{(0)} \leftarrow W$;
- **para** $k \leftarrow 1$ **a** n **hacer**
  - Crear una matriz $D^{(k)} = (d_{ij}^{(k)})$ de dimensión $n \times n$;
  - **para** $i \leftarrow 1$ **a** n **hacer**
    - **para** $j \leftarrow 1$ **a** n **hacer**
      - $d_{ij}^{(k)} \leftarrow \min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)})$;
- **devolver** $D^{(n)}$;

Análisis del algoritmo de Floyd-Warshall

- Linea 2 es $O(1)$.
- La línea 3 es una copia de una matriz es $O(|V|^2)$.
- Suponemos que la línea 6, creación de la matriz, es $O(1)$.
- La línea 8 es $O(1)$.
- Las operaciones dominantes son las líneas 4-8.
- Las las líneas 4-8 contienen tres ciclos **para**, desde 1 hasta n.
- Por lo tanto, por las líneas 4-8 el algoritmo de Floyd-Warshall es $\Theta(|V|^3)$.

Análisis del algoritmo de Floyd-Warshall

- Linea 2 es $O(1)$.
- La línea 3 es una copia de una matriz es $O(|V|^2)$.
- Suponemos que la línea 6, creación de la matriz, es $O(1)$.
- La línea 8 es $O(1)$.
- Las operaciones dominantes son las líneas 4-8.
- Las las líneas 4-8 contienen tres ciclos **para**, desde 1 hasta n.
- Por lo tanto, por las líneas 4-8 el algoritmo de Floyd-Warshall es $\Theta(|V|^3)$.

Análisis del algoritmo de Floyd-Warshall

- Linea 2 es $O(1)$.
- La línea 3 es una copia de una matriz es $O(|V|^2)$.
- Suponemos que la línea 6, creación de la matriz, es $O(1)$.
- La línea 8 es $O(1)$.
- Las operaciones dominantes son las líneas 4-8.
- Las las líneas 4-8 contienen tres ciclos **para**, desde 1 hasta n.
- Por lo tanto, por las líneas 4-8 el algoritmo de Floyd-Warshall es $\Theta(|V|^3)$.

Análisis del algoritmo de Floyd-Warshall

- Linea 2 es $O(1)$.
- La línea 3 es una copia de una matriz es $O(|V|^2)$.
- Suponemos que la línea 6, creación de la matriz, es $O(1)$.
- La línea 8 es $O(1)$.
- Las operaciones dominantes son las líneas 4-8.
- Las las líneas 4-8 contienen tres ciclos **para**, desde 1 hasta n.
- Por lo tanto, por las líneas 4-8 el algoritmo de Floyd-Warshall es $\Theta(|V|^3)$.

Análisis del algoritmo de Floyd-Warshall

- Linea 2 es $O(1)$.
- La línea 3 es una copia de una matriz es $O(|V|^2)$.
- Suponemos que la línea 6, creación de la matriz, es $O(1)$.
- La línea 8 es $O(1)$.
- Las operaciones dominantes son las líneas 4-8.
- Las las líneas 4-8 contienen tres ciclos **para**, desde 1 hasta n.
- Por lo tanto, por las líneas 4-8 el algoritmo de Floyd-Warshall es $\Theta(|V|^3)$.

Análisis del algoritmo de Floyd-Warshall

- Linea 2 es $O(1)$.
- La línea 3 es una copia de una matriz es $O(|V|^2)$.
- Suponemos que la línea 6, creación de la matriz, es $O(1)$.
- La línea 8 es $O(1)$.
- Las operaciones dominantes son las líneas 4-8.
- Las las líneas 4-8 contienen tres ciclos **para**, desde 1 hasta n.
- Por lo tanto, por las líneas 4-8 el algoritmo de Floyd-Warshall es $\Theta(|V|^3)$.

Análisis del algoritmo de Floyd-Warshall

- Linea 2 es $O(1)$.
- La línea 3 es una copia de una matriz es $O(|V|^2)$.
- Suponemos que la línea 6, creación de la matriz, es $O(1)$.
- La línea 8 es $O(1)$.
- Las operaciones dominantes son las líneas 4-8.
- Las las líneas 4-8 contienen tres ciclos **para**, desde 1 hasta n.
- Por lo tanto, por las líneas 4-8 el algoritmo de Floyd-Warshall es $\Theta(|V|^3)$.

Construcción de los caminos de costo mínimo

- Se computa la matriz Π mientras el algoritmo de Floyd-Warshall computa $D^{(k)}$.
- Se computa una secuencia de matrices $\Pi^{(0)}, \Pi^{(1)}, \ldots, \Pi^{(n)}$, donde $\Pi^{(n)} = \Pi$.
- Recordar que $\Pi = \Pi^{(n)} = (\pi_{ij}^{(n)})$.
- Cuando $k = 0$, el camino desde i hasta j no tiene vértices intermedios:

$$\pi_{ij}^{(0)} = \begin{cases} NIL & \text{si } i = j \vee w_{ij} = \infty, \\ i & \text{si } i \neq j \vee w_{ij} < \infty. \end{cases} \tag{3}$$

- Si $k \geq 1$, hay un camino $i \rightsquigarrow k \rightsquigarrow j$ donde $k \neq j$, entonces:

$$\pi_{ij}^{(k)} = \begin{cases} \pi_{ij}^{(k-1)} & \text{si } d_{ij}^{(k-1)} < d_{ik}^{(k-1)} + d_{kj}^{(k-1)}, \\ \pi_{kj}^{(k-1)} & \text{si } d_{ij}^{(k-1)} \geq d_{ik}^{(k-1)} + d_{kj}^{(k-1)}. \end{cases} \tag{4}$$

Construcción de los caminos de costo mínimo

- Se computa la matriz Π mientras el algoritmo de Floyd-Warshall computa $D^{(k)}$.
- Se computa una secuencia de matrices $\Pi^{(0)}, \Pi^{(1)}, \ldots, \Pi^{(n)}$, donde $\Pi^{(n)} = \Pi$.
- Recordar que $\Pi = \Pi^{(n)} = (\pi_{ij}^{(n)})$.
- Cuando $k = 0$, el camino desde i hasta j no tiene vértices intermedios:

$$\pi_{ij}^{(0)} = \begin{cases} NIL & \text{si } i = j \vee w_{ij} = \infty, \\ i & \text{si } i \neq j \vee w_{ij} < \infty. \end{cases} \tag{3}$$

- Si $k \geq 1$, hay un camino $i \rightsquigarrow k \rightsquigarrow j$ donde $k \neq j$, entonces:

$$\pi_{ij}^{(k)} = \begin{cases} \pi_{ij}^{(k-1)} & \text{si } d_{ij}^{(k-1)} < d_{ik}^{(k-1)} + d_{kj}^{(k-1)}, \\ \pi_{kj}^{(k-1)} & \text{si } d_{ij}^{(k-1)} \geq d_{ik}^{(k-1)} + d_{kj}^{(k-1)}. \end{cases} \tag{4}$$

Construcción de los caminos de costo mínimo

- Se computa la matriz Π mientras el algoritmo de Floyd-Warshall computa $D^{(k)}$.
- Se computa una secuencia de matrices $\Pi^{(0)}, \Pi^{(1)}, \ldots, \Pi^{(n)}$, donde $\Pi^{(n)} = \Pi$.
- Recordar que $\Pi = \Pi^{(n)} = (\pi_{ij}^{(n)})$.
- Cuando $k = 0$, el camino desde i hasta j no tiene vértices intermedios:

$$\pi_{ij}^{(0)} = \begin{cases} NIL & \text{si } i = j \vee w_{ij} = \infty, \\ i & \text{si } i \neq j \vee w_{ij} < \infty. \end{cases} \tag{3}$$

- Si $k \geq 1$, hay un camino $i \rightsquigarrow k \rightsquigarrow j$ donde $k \neq j$, entonces:

$$\pi_{ij}^{(k)} = \begin{cases} \pi_{ij}^{(k-1)} & \text{si } d_{ij}^{(k-1)} < d_{ik}^{(k-1)} + d_{kj}^{(k-1)}, \\ \pi_{kj}^{(k-1)} & \text{si } d_{ij}^{(k-1)} \geq d_{ik}^{(k-1)} + d_{kj}^{(k-1)}. \end{cases} \tag{4}$$

Construcción de los caminos de costo mínimo

- Se computa la matriz Π mientras el algoritmo de Floyd-Warshall computa $D^{(k)}$.
- Se computa una secuencia de matrices $\Pi^{(0)}, \Pi^{(1)}, \ldots, \Pi^{(n)}$, donde $\Pi^{(n)} = \Pi$.
- Recordar que $\Pi = \Pi^{(n)} = (\pi_{ij}^{(n)})$.
- Cuando $k = 0$, el camino desde i hasta j no tiene vértices intermedios:

$$\pi_{ij}^{(0)} = \begin{cases} NIL & \text{si } i = j \vee w_{ij} = \infty, \\ i & \text{si } i \neq j \vee w_{ij} < \infty. \end{cases} \tag{3}$$

- Si $k \geq 1$, hay un camino $i \rightsquigarrow k \rightsquigarrow j$ donde $k \neq j$, entonces:

$$\pi_{ij}^{(k)} = \begin{cases} \pi_{ij}^{(k-1)} & \text{si } d_{ij}^{(k-1)} < d_{ik}^{(k-1)} + d_{kj}^{(k-1)}, \\ \pi_{kj}^{(k-1)} & \text{si } d_{ij}^{(k-1)} \geq d_{ik}^{(k-1)} + d_{kj}^{(k-1)}. \end{cases} \tag{4}$$

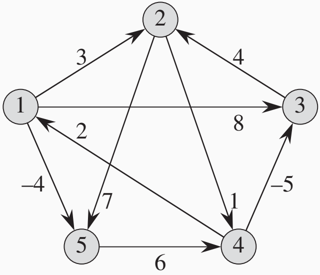
Construcción de los caminos de costo mínimo

- Se computa la matriz Π mientras el algoritmo de Floyd-Warshall computa $D^{(k)}$.
- Se computa una secuencia de matrices $\Pi^{(0)}, \Pi^{(1)}, \ldots, \Pi^{(n)}$, donde $\Pi^{(n)} = \Pi$.
- Recordar que $\Pi = \Pi^{(n)} = (\pi_{ij}^{(n)})$.
- Cuando $k = 0$, el camino desde i hasta j no tiene vértices intermedios:

$$\pi_{ij}^{(0)} = \begin{cases} NIL & \text{si } i = j \vee w_{ij} = \infty, \\ i & \text{si } i \neq j \vee w_{ij} < \infty. \end{cases} \tag{3}$$

- Si $k \geq 1$, hay un camino $i \rightsquigarrow k \rightsquigarrow j$ donde $k \neq j$, entonces:

$$\pi_{ij}^{(k)} = \begin{cases} \pi_{ij}^{(k-1)} & \text{si } d_{ij}^{(k-1)} < d_{ik}^{(k-1)} + d_{kj}^{(k-1)}, \\ \pi_{kj}^{(k-1)} & \text{si } d_{ij}^{(k-1)} \geq d_{ik}^{(k-1)} + d_{kj}^{(k-1)}. \end{cases} \tag{4}$$



(a) Digrafo de entrada.

$$W = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

(b) Matriz de costos W.

Figura 1: En la Figura 8(a) se muestra el digrafo de entrada para el algoritmo de Floyd-Warshall y en la Figura 8(b) muestra su matriz W de costos correspondiente. Fuente [1].

Ejemplo del algoritmo de Floyd-Warshall, cont.

$$D^{(0)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

(a) Matriz de distancias $D^{(0)}$.

$$\Pi^{(0)} = \begin{pmatrix} \text{NIL} & 1 & 1 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & \text{NIL} & \text{NIL} \\ 4 & \text{NIL} & 4 & \text{NIL} & \text{NIL} \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

(b) Matriz de predecesores $\Pi^{(0)}$.

Figura 2: Matrices $D^{(0)}$ y $\Pi^{(0)}$ computadas por el algoritmo de Floyd-Warshall para el grafo de la Figura 1. Fuente [1].

Ejemplo del algoritmo de Floyd-Warshall, cont.

$$D^{(1)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \quad \Pi^{(1)} = \begin{pmatrix} \text{NIL} & 1 & 1 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & \text{NIL} & \text{NIL} \\ 4 & 1 & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

(a) Matriz de distancias $D^{(1)}$. (b) Matriz de predecesores $\Pi^{(1)}$.

Figura 3: Matrices $D^{(1)}$ y $\Pi^{(1)}$ computadas por el algoritmo de Floyd-Warshall para el grafo de la Figura 1. Fuente [1].

Ejemplo del algoritmo de Floyd-Warshall, cont.

$$D^{(2)} = \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

(a) Matriz de distancias $D^{(2)}$.

$$\Pi^{(2)} = \begin{pmatrix} \text{NIL} & 1 & 1 & 2 & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & 2 & 2 \\ 4 & 1 & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

(b) Matriz de predecesores $\Pi^{(2)}$.

Figura 4: Matrices $D^{(2)}$ y $\Pi^{(2)}$ computadas por el algoritmo de Floyd-Warshall para el grafo de la Figura 1. Fuente [1].

Ejemplo del algoritmo de Floyd-Warshall, cont.

$$D^{(3)} = \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

(a) Matriz de distancias $D^{(3)}$.

$$\Pi^{(3)} = \begin{pmatrix} \text{NIL} & 1 & 1 & 2 & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & 2 & 2 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

(b) Matriz de predecesores $\Pi^{(3)}$.

Figura 5: Matrices $D^{(3)}$ y $\Pi^{(3)}$ computadas por el algoritmo de Floyd-Warshall para el grafo de la Figura 1. Fuente [1].

Ejemplo del algoritmo de Floyd-Warshall, cont.

$$D^{(4)} = \begin{pmatrix} 0 & 3 & -1 & 4 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix}$$

(a) Matriz de distancias $D^{(4)}$.

$$\Pi^{(4)} = \begin{pmatrix} \text{NIL} & 1 & 4 & 2 & 1 \\ 4 & \text{NIL} & 4 & 2 & 1 \\ 4 & 3 & \text{NIL} & 2 & 1 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ 4 & 3 & 4 & 5 & \text{NIL} \end{pmatrix}$$

(b) Matriz de predecesores $\Pi^{(4)}$.

Figura 6: Matrices $D^{(4)}$ y $\Pi^{(4)}$ computadas por el algoritmo de Floyd-Warshall para el grafo de la Figura 1. Fuente [1].

Ejemplo del algoritmo de Floyd-Warshall, cont.

$$D^{(5)} = \begin{pmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix}$$

(a) Matriz de distancias $D^{(5)}$.

$$\Pi^{(5)} = \begin{pmatrix} \text{NIL} & 3 & 4 & 5 & 1 \\ 4 & \text{NIL} & 4 & 2 & 1 \\ 4 & 3 & \text{NIL} & 2 & 1 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ 4 & 3 & 4 & 5 & \text{NIL} \end{pmatrix}$$

(b) Matriz de predecesores $\Pi^{(5)}$.

Figura 7: Matrices $D^{(5)}$ y $\Pi^{(5)}$ computadas por el algoritmo de Floyd-Warshall para el grafo de la Figura 1. Fuente [1].

Clausura transitiva

Clausura transitiva de un digrafo

Definición de la clausura transitiva de un digrafo

Dado un digrafo $G = (V, E)$ con un conjunto de vértices $V = \{1, 2, \ldots, n\}$, se define la *clausura transitiva* de un digrafo G como el grafo $G^* = (V, E^*)$, donde

$$E^* = \{(i, j) : \text{si hay un camino desde el vértice } i \text{ hasta el vértice } j \text{ en } G\}$$

Computando la clausura transitiva de un digrafo

- Se construye matrices de alcance $T^{(k)} = (t_{ij}^{(k)})$, en orden creciente de k de forma recursiva.
- Si $k = 0$ se tiene que:

$$t_{ij}^{(0)} = \begin{cases} 0 & \text{si } i \neq j \wedge (i,j) \notin E, \\ 1 & \text{si } i = j \vee (i,j) \in E. \end{cases} \tag{5}$$

- Si $k \geq 1$ se tiene que:

$$t_{ij}^{(k)} = t_{ij}^{(k-1)} \vee (t_{ik}^{(k-1)} \wedge t_{kj}^{(k-1)}). \tag{6}$$

Computando la clausura transitiva de un digrafo

- Se construye matrices de alcance $T^{(k)} = (t_{ij}^{(k)})$, en orden creciente de k de forma recursiva.
- Si $k = 0$ se tiene que:

$$t_{ij}^{(0)} = \begin{cases} 0 & \text{si } i \neq j \wedge (i,j) \notin E, \\ 1 & \text{si } i = j \vee (i,j) \in E. \end{cases} \tag{5}$$

- Si $k \geq 1$ se tiene que:

$$t_{ij}^{(k)} = t_{ij}^{(k-1)} \vee (t_{ik}^{(k-1)} \wedge t_{kj}^{(k-1)}). \tag{6}$$

Computando la clausura transitiva de un digrafo

- Se construye matrices de alcance $T^{(k)} = (t_{ij}^{(k)})$, en orden creciente de k de forma recursiva.
- Si $k = 0$ se tiene que:

$$t_{ij}^{(0)} = \begin{cases} 0 & \text{si } i \neq j \wedge (i,j) \notin E, \\ 1 & \text{si } i = j \vee (i,j) \in E. \end{cases} \tag{5}$$

- Si $k \geq 1$ se tiene que:

$$t_{ij}^{(k)} = t_{ij}^{(k-1)} \vee (t_{ik}^{(k-1)} \wedge t_{kj}^{(k-1)}). \tag{6}$$

Algoritmo de la clausura transitiva de un digrafo

Función Clausura-Transitiva($G = (V, E)$)

inicio
  $n \leftarrow |V|$;
  Sea $T^{(0)} = (t_{ij}^{(0)})$ una nueva matriz $n \times n$;
  para $i \leftarrow 1$ **a** n **hacer**
    para $j \leftarrow 1$ **a** n **hacer**
      si $i = j \vee (i, j) \in E$ **entonces** $t_{ij}^{(0)} = 1$;
      en otro caso $t_{ij}^{(0)} = 0$;
  para $k \leftarrow 1$ **a** n **hacer**
    Sea $T^{(k)} = (t_{ij}^{(k)})$ una nueva matriz $n \times n$;
    para $i \leftarrow 1$ **a** n **hacer**
      para $j \leftarrow 1$ **a** n **hacer**
        $t_{ij}^{(k)} = t_{ij}^{(k-1)} \vee (t_{ik}^{(k-1)} \wedge t_{kj}^{(k-1)})$;
  devolver $T^{(n)}$;

Análisis del algoritmo de la clausura transitiva

- La creación de la matriz de las líneas 3 y 9 es $O(1)$.
- La línea 12 es $O(1)$.
- Las operaciones dominantes son los tres ciclos **para** de las líneas 8-12.
- Por lo tanto, el algoritmo es $\Theta(|V|^3)$.

Análisis del algoritmo de la clausura transitiva

- La creación de la matriz de las líneas 3 y 9 es $O(1)$.
- La línea 12 es $O(1)$.
- Las operaciones dominantes son los tres ciclos **para** de las líneas 8-12.
- Por lo tanto, el algoritmo es $\Theta(|V|^3)$.

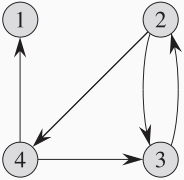
Análisis del algoritmo de la clausura transitiva

- La creación de la matriz de las líneas 3 y 9 es $O(1)$.
- La línea 12 es $O(1)$.
- Las operaciones dominantes son los tres ciclos **para** de las líneas 8-12.
- Por lo tanto, el algoritmo es $\Theta(|V|^3)$.

Análisis del algoritmo de la clausura transitiva

- La creación de la matriz de las líneas 3 y 9 es $O(1)$.
- La línea 12 es $O(1)$.
- Las operaciones dominantes son los tres ciclos **para** de las líneas 8-12.
- Por lo tanto, el algoritmo es $\Theta(|V|^3)$.

Ejemplo del algoritmo de la clausura transitiva



(a) Digrafo.

$$T^{(0)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{pmatrix} \quad T^{(1)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{pmatrix} \quad T^{(2)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix}$$

$$T^{(3)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix} \quad T^{(4)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

(b) Matrices de alcance $T^{(k)}$ en cada iteración k.

Figura 8: Digrafo de entrada y las matrices de alcance $T^{(k)}$ computadas por el algoritmo Clausura-Transitiva. Fuente [1].

Referencias

[1] T. Cormen, C. Leirserson, R. Rivest, and C. Stein.
***Introduction to Algorithms*.**
McGraw Hill, 3ra edition, 2009.