



# TEORÍA DE GRAFOS: ALGORITMO FLOYD-WARSHALL

**Autores:**

Colectivo de Entrenadores ACM-ICPC UM

Marzo / 2019

ENTRENAMIENTO PARA CONCURSANTES  
ACM-ICPC DE LA UNIVERSIDAD DE  
MATANZAS

# Objetivos

- ▶ Familiarizar a los estudiantes con conceptos de Teoría de Grafos.
- ▶ Familiarizar a los estudiantes con representaciones computacionales de un grafo.
- ▶ Caracterizar el algortimo Floyd-Warshall.

# Objetivos

- ▶ Implementar el algoritmo Floyd-Warshall.
- ▶ Identificar problemas donde la solución sea el algoritmo Floyd-Warshall.

# Bibliografía

- ▶ *Manual de preparación para concursantes ACM-ICPC de la Universidad de Matanzas.* Publicado en el Entorno Virtual de Aprendizaje de la Universidad de Matanzas ( [eva.umcc.cu](http://eva.umcc.cu) ). Sección *Bibliografía*.

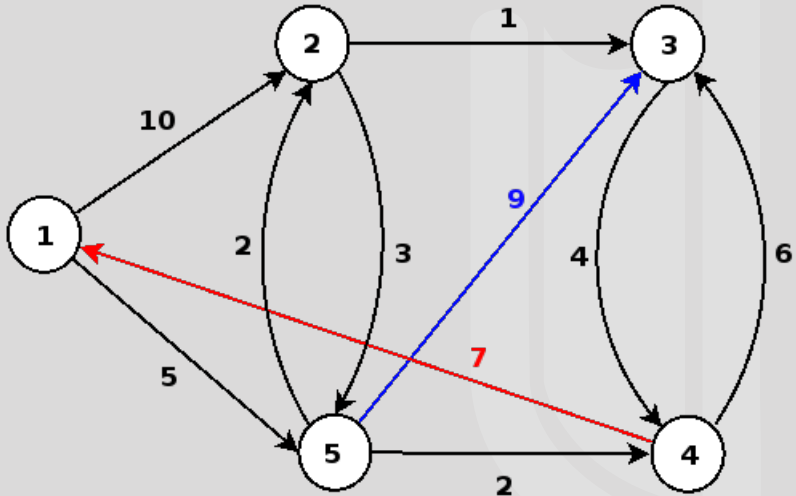
# Teoría de Grafos

La Teoría de grafo es un campo de estudio de las matemáticas y las ciencias de la computación, que estudia las propiedades de los grafos ...

# Teoría de Grafos

... estructuras que constan de dos partes, el conjunto de vértices, nodos o puntos; y el conjunto de aristas, líneas o lados (edges en inglés) que pueden ser orientados o no.

# Teoría de Grafos



# Representación de un grafo

- ▶ Existen diferentes formas de representar un grafo (simple), además de la geométrica y muchos métodos para almacenarlos en una computadora.
- ▶ La estructura de datos usada depende de las características del grafo y el algoritmo usado para manipularlo.



# Representación de un grafo

- ▶ Entre las estructuras más sencillas y usadas se encuentran las listas y las matrices, aunque frecuentemente se usa una combinación de ambas.

# Representación de un grafo

**Matriz de adyacencia:** El grafo está representado por una matriz cuadrada  $M$  de tamaño  $n^2$ , donde  $n$  es el número de vértices. Si hay una arista entre un vértice  $x$  y un vértice  $y$ , entonces el elemento  $m_{x,y}$  es 1, de lo contrario, es 0.

# Representación de un grafo

	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	1	0	0	1
2	0	0	0	1	0	1
3	0	0	0	0	1	0
4	0	1	0	1	0	0
5	0	0	1	1	1	0

# Representación de un grafo

**Matriz de costo:** El grafo está representado por una matriz cuadrada  $M$  de tamaño  $n^2$ , donde  $n$  es el número de vértices. Si hay una arista entre un vértice  $x$  y un vértice  $y$ , entonces el elemento  $m_{x,y}$  es igual al valor de ponderación de la arista o el peso, de lo contrario se coloca un valor que indica que no existe arista ( $\infty$ )

# Representación de un grafo

	0	1	2	3	4	5
0	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
1	$\infty$	$\infty$	10	$\infty$	$\infty$	5
2	$\infty$	$\infty$	$\infty$	1	$\infty$	3
3	$\infty$	$\infty$	$\infty$	$\infty$	4	$\infty$
4	$\infty$	7	$\infty$	6	$\infty$	$\infty$
5	$\infty$	$\infty$	2	9	2	$\infty$

# Algoritmo Floyd-Warshall

Es un algoritmo de análisis sobre grafos para encontrar el camino mínimo en grafos dirigidos ponderados.

El algoritmo encuentra el camino entre todos los pares de vértices en una única ejecución.

# Algoritmo Floyd-Warshall

El algoritmo de Floyd-Warshall es un ejemplo de programación dinámica.

Compara todos los posibles caminos a través del grafo entre cada par de vértices.

# Algoritmo Floyd-Warshall

Para que haya coherencia numérica, Floyd-Warshall supone que no hay ciclos negativos (aristas con peso o ponderación negativo).



# Algoritmo Floyd-Warshall

El objetivo del algoritmo es encontrar el camino mínimo desde cada vértice  $i$  a cada vértice  $j$  usando únicamente los vértices  $k$  tal que  $v_i \neq v_j$ ,  $v_j \neq v_k$  y  $v_i \neq v_k$ .

$$m[i][j] = \min(m[i][j], m[i][k] + m[k][j])$$

# Algoritmo Floyd-Warshall

Utiliza una copia de la matriz de costo del grafo sobre la cual realiza las operaciones y modificaciones quedando en esta estructura una vez finalizado el algoritmo el camino mínimo para cada par de vértices del grafo.

# Algoritmo Floyd-Warshall

El costo computacional es  $N^2$  siendo  $N$  la cantidad de vértices del grafo.

Su complejidad es  $N^3$  siendo  $N$  la cantidad de vértices del grafo.

Solo es recomendado utilizarlo en problema donde la cantidad de nodos es menor o igual que 100.

# Implementación Floyd-Warshall

```
inline void FloydWarshall(){  
    for(int i=1;i<=n;i++){  
        for(int j=1;j<=n;j++){  
            floyd[i][j] = dist[i][j];  
        }  
        floyd[i][i] = 0;  
    }  
    .....  
}
```

# Implementación Floyd-Warshall

```
....  
for(int k=1;k<=n;k++){  
    for(int i=1;i<=n;i++){  
        for(int j=1;j<=n;j++){  
            if(floyd[i][k]+floyd[k][j]<floyd[i][j]){  
                floyd[i][j] = floyd[i][k] + floyd[k][j]  
            };  
        }  
    }  
}  
}
```

# Problemas

- ▶ COJ 2081 - Saving Money
- ▶ COJ 3651 - Center of the City
- ▶ COJ 3542 - Farmer John and the Teleports

# Conclusiones

- ▶ El algoritmo encuentra el camino entre todos los pares de vértices en una única ejecución.
- ▶ El costo computacional es  $N^2$  siendo  $N$  la cantidad de vértices del grafo.

# Conclusiones

- ▶ Su complejidad es  $N^3$  siendo  $N$  la cantidad de vértices del grafo.
- ▶ Solo es recomendado utilizarlo en problema donde la cantidad de nodos es menor o igual que 100.



# UNIVERSIDAD DE MATANZAS

cosechando el saber

FIN