Representación de Grafos

Laboratorio de Algoritmos y Estructuras de Datos III

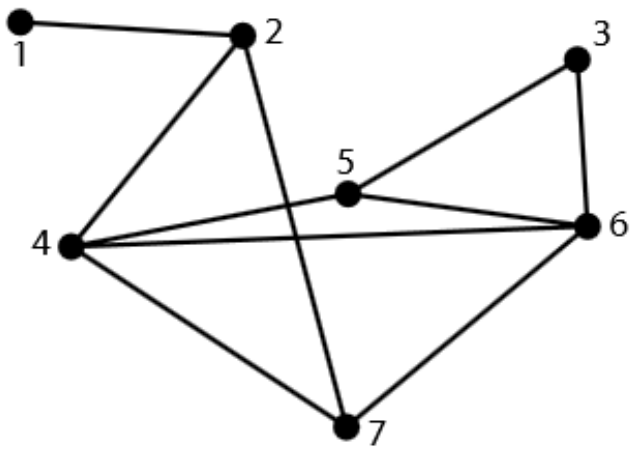
2^{do} Cuatrimestre de 2016

1. Matriz de adyacencia
2. Lista de adyacencia (o listas de vecinos)
3. Lista de incidencia (o lista de aristas)
4. Ejercicio

Matriz de adyacencia

Dado un grafo G con n vértices numerados de 1 a n, se define su matriz de adyacencia M como la matriz de tamaño $n \times n$ tal que $M_{ij} = 1$ si hay arista de i a j y $M_{ij} = 0$ en caso contrario.

Ej:



$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

Según el tipo de grafo

- Si el grafo es no dirigido, $M_{ij} = M_{ji}$ para cualquier $1 \leq i, j \leq n$. Luego la matriz es simétrica.
- Si el grafo es dirigido y hay arista de 1 a 2 pero no de 2 a 1, tendremos $M_{1,2} = 1$ pero $M_{2,1} = 0$.
- Si las aristas tienen peso, se puede poner en la matriz en lugar de unos, los pesos de las aristas. Recordando siempre guardar un valor especial para cuando no hay arista (lo que serían los ceros).

Matriz de adyacencia

Ventajas

Matriz de adyacencia

Ventajas

- Permite saber si existe o no arista entre dos nodos cualesquiera en O(1).
- Es muy fácil de implementar, $matrizAdy[i][j]$ guarda toda la información sobre la arista.

Matriz de adyacencia

Ventajas

- Permite saber si existe o no arista entre dos nodos cualesquiera en O(1).
- Es muy fácil de implementar, $matrizAdy[i][j]$ guarda toda la información sobre la arista.

Desventajas

Matriz de adyacencia

Ventajas

- Permite saber si existe o no arista entre dos nodos cualesquiera en O(1).
- Es muy fácil de implementar, $matrizAdy[i][j]$ guarda toda la información sobre la arista.

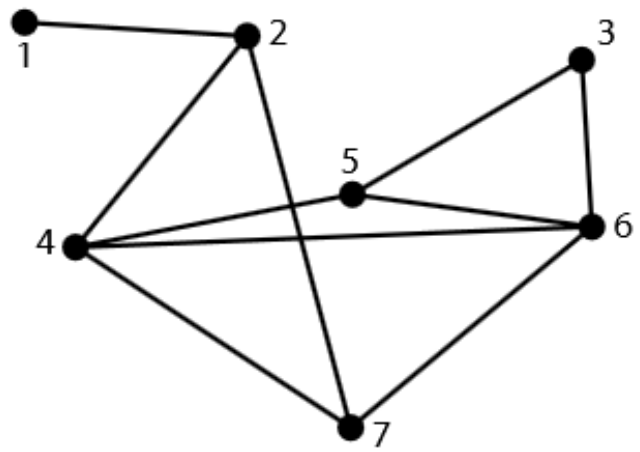
Desventajas

- La complejidad espacial: se necesitan n^2 casillas para representar un grafo de n nodos.
- Recorrer los vecinos de un nodo siempre tarda $O(n)$.

Lista de adyacencia

Coloquialmente la llamamos *lista de vecinos* pues para cada nodo guardamos la lista de nodos para los que existe una arista que los conecta (o sea, los vecinos).

Ej:



$$
\begin{aligned}
L_1 &: 2 \\
L_2 &: 1 \longrightarrow 4 \longrightarrow 7 \\
L_3 &: 5 \longrightarrow 6 \\
L_4 &: 2 \longrightarrow 5 \longrightarrow 6 \longrightarrow 7 \\
L_5 &: 3 \longrightarrow 4 \longrightarrow 6 \\
L_6 &: 3 \longrightarrow 4 \longrightarrow 5 \longrightarrow 7 \\
L_7 &: 2 \longrightarrow 4 \longrightarrow 6
\end{aligned}
$$

Lista de adyacencia

Nuevamente, con la misma idea también se pueden modelar grafos dirigidos y con pesos. En el caso de grafos con pesos, guardamos para cada nodo una lista de pares (nodo, peso) que indica un nodo vecino y el peso de la arista que los une.

En principio, la lista de vecinos no tiene por qué estar ordenada. Pero asumiendo que sí lo está, ¿Qué complejidad será necesaria para consultar si dos nodos tienen una arista que los une?

Lista de adyacencia

Nuevamente, con la misma idea también se pueden modelar grafos dirigidos y con pesos. En el caso de grafos con pesos, guardamos para cada nodo una lista de pares (nodo, peso) que indica un nodo vecino y el peso de la arista que los une.

En principio, la lista de vecinos no tiene por qué estar ordenada. Pero asumiendo que sí lo está, ¿Qué complejidad será necesaria para consultar si dos nodos tienen una arista que los une? **O(lg n)**

Lista de adyacencia

Nuevamente, con la misma idea también se pueden modelar grafos dirigidos y con pesos. En el caso de grafos con pesos, guardamos para cada nodo una lista de pares (nodo, peso) que indica un nodo vecino y el peso de la arista que los une.

En principio, la lista de vecinos no tiene por qué estar ordenada. Pero asumiendo que sí lo está, ¿Qué complejidad será necesaria para consultar si dos nodos tienen una arista que los une? **O(lg n)**

La complejidad espacial de esta representación será posiblemente mucho menor. ¿Cuánta memoria necesitaremos para un grafo de n nodos y m aristas?

Lista de adyacencia

Nuevamente, con la misma idea también se pueden modelar grafos dirigidos y con pesos. En el caso de grafos con pesos, guardamos para cada nodo una lista de pares (nodo, peso) que indica un nodo vecino y el peso de la arista que los une.

En principio, la lista de vecinos no tiene por qué estar ordenada. Pero asumiendo que sí lo está, ¿Qué complejidad será necesaria para consultar si dos nodos tienen una arista que los une? **O(lg n)**

La complejidad espacial de esta representación será posiblemente mucho menor. ¿Cuánta memoria necesitaremos para un grafo de n nodos y m aristas? **O(m+n)**

Lista de adyacencia

Ventajas

Lista de adyacencia

Ventajas

- Complejidad espacial lineal en el tamaño del grafo.
- Recorrer los vecinos de un nodo es $O(v)$ donde v es la cantidad de vecinos.

Lista de adyacencia

Ventajas

- Complejidad espacial lineal en el tamaño del grafo.
- Recorrer los vecinos de un nodo es $O(v)$ donde v es la cantidad de vecinos.

Desventajas

Lista de adyacencia

Ventajas

- Complejidad espacial lineal en el tamaño del grafo.
- Recorrer los vecinos de un nodo es $O(v)$ donde v es la cantidad de vecinos.

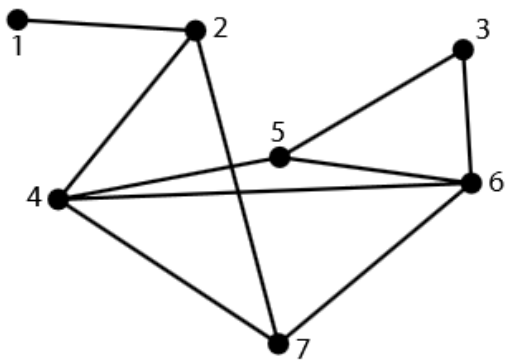
Desventajas

- Chequear si dos nodos son adyacentes ya no es $O(1)$.
- Más difícil de implementar y manejar que la matriz de adyacencia.

Lista de incidencia

La lista de incidencia de un grafo G se define como la cantidad de nodos de G y una lista con las aristas de G representadas por los dos vértices que conecta.

Ej:



Cant. vértices: 7

Aristas: (1,2),(2,4),(2,7),(3,5),
(3,6),(4,5),(4,6),(4,7),(5,6),(6,7)

Lista de incidencia

Ventajas

Lista de incidencia

Ventajas

- Complejidad espacial lineal en la cantidad de aristas.
- Fácil de implementar.
- Puedo ordenar las aristas con algun criterio.

Lista de incidencia

Ventajas

- Complejidad espacial lineal en la cantidad de aristas.
- Fácil de implementar.
- Puedo ordenar las aristas con algun criterio.

Desventajas

Lista de incidencia

Ventajas

- Complejidad espacial lineal en la cantidad de aristas.
- Fácil de implementar.
- Puedo ordenar las aristas con algun criterio.

Desventajas

- Chequear si dos nodos son adyacentes no es $O(1)$.

Y recorrer los vecinos?

Buscar triángulos

Dado un grafo G, el problema consiste en decir si existe un subgrafo K_3.
Dar un algoritmo para resolver el problema teniendo:

- Una matriz de adyacencia.
- Una lista de vecinos.
- Una lista de incidencia.