

Grafos

MSc Edson Ticona Zegarra

Campamento de Programación

Contenido

Introducción

Representación

Búsqueda en grafos

BFS

DFS

Árboles

Contenido

Introducción

Representación

Búsqueda en grafos

BFS

DFS

Árboles

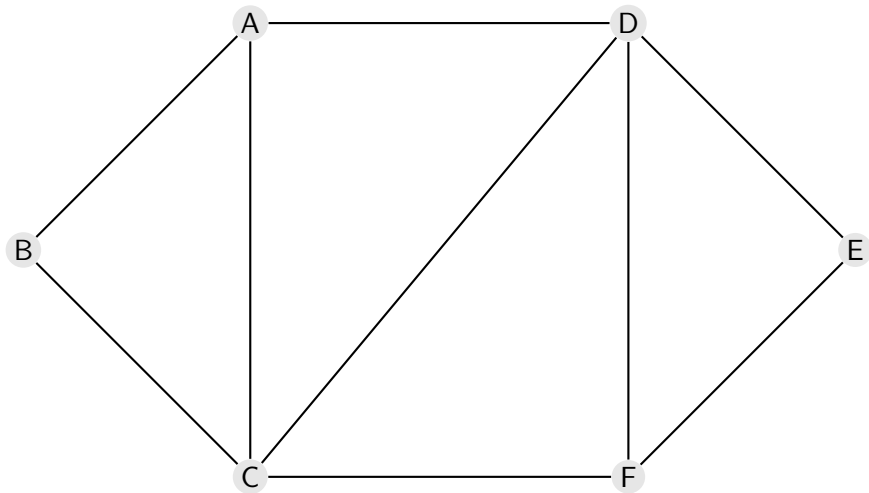
Grafo

- Un grafo es una estructura de datos definida por un par de conjuntos: los vértices V y las aristas E .

Grafo

- ▶ Un grafo es una estructura de datos definida por un par de conjuntos: los vértices V y las aristas E .
- ▶ Toda arista conecta un par de vértices, entonces decimos que $G = (V, E)$

Grafo



Grafo

- ▶ Para el grafo anterior, $V = \{A, B, C, D, E, F\}$

Grafo

- ▶ Para el grafo anterior, $V = \{A, B, C, D, E, F\}$
- ▶ $E =$
 $\{(D, A), (B, A), (B, C), (A, C), (D, C), (F, C), (F, E), (D, E), (D, F)\}$

Grafo

- ▶ Para el grafo anterior, $V = \{A, B, C, D, E, F\}$
- ▶ $E =$
 $\{(D, A), (B, A), (B, C), (A, C), (D, C), (F, C), (F, E), (D, E), (D, F)\}$
- ▶ formalmente $G = (V, E)$

Grafo

- Los grafos son útiles para representar: redes de comunicaciones, caminos, redes sociales, etc

Grafo

- ▶ Los grafos son útiles para representar: redes de comunicaciones, caminos, redes sociales, etc
- ▶ Usualmente consideramos a los grafos como no direccionados, es decir, si existe un vértice (u, v) , se puede ir de u a v y viceversa

Grafo

- ▶ Los grafos son útiles para representar: redes de comunicaciones, caminos, redes sociales, etc
- ▶ Usualmente consideramos a los grafos como no direccionados, es decir, si existe un vértice (u, v) , se puede ir de u a v y viceversa
- ▶ Por otro lado, si consideramos grafos direccionados, gráficamente utilizamos flechas para indicar la dirección y una arista (u, v) indica que se puede ir de u a v , pero **no** de v a u . En este caso usualmente utilizamos el término *arco* y no arista, así también decimos que $G = (V, A)$

Grafo

- ▶ Los grafos son útiles para representar: redes de comunicaciones, caminos, redes sociales, etc
- ▶ Usualmente consideramos a los grafos como no direccionados, es decir, si existe un vértice (u, v) , se puede ir de u a v y viceversa
- ▶ Por otro lado, si consideramos grafos direccionados, gráficamente utilizamos flechas para indicar la dirección y una arista (u, v) indica que se puede ir de u a v , pero **no** de v a u . En este caso usualmente utilizamos el término *arco* y no arista, así también decimos que $G = (V, A)$
- ▶ Entre cualquier par de vértices u y v , solo existe a lo mucho una arista para grafos no direccionados.

Grafo

- ▶ Los grafos son útiles para representar: redes de comunicaciones, caminos, redes sociales, etc
- ▶ Usualmente consideramos a los grafos como no direccionados, es decir, si existe un vértice (u, v) , se puede ir de u a v y viceversa
- ▶ Por otro lado, si consideramos grafos direccionados, gráficamente utilizamos flechas para indicar la dirección y una arista (u, v) indica que se puede ir de u a v , pero **no** de v a u . En este caso usualmente utilizamos el término *arco* y no arista, así también decimos que $G = (V, A)$
- ▶ Entre cualquier par de vértices u y v , solo existe a lo mucho una arista para grafos no direccionados.
- ▶ Para grafos direccionados, para cualquier par de vértices existe a lo mucho una arista (u, v) y una arista (v, u)

Grafo

- ▶ Adicionalmente una arista (u, v) puede recibir un *peso*, que puede representar el costo para ir de u a v y se representa por $c_{u,v}$

Grafo

- ▶ Adicionalmente una arista (u, v) puede recibir un *peso*, que puede representar el costo para ir de u a v y se representa por $c_{u,v}$
- ▶ A dichos grafos se les conoce como grafos con pesos

Grafo

- ▶ Adicionalmente una arista (u, v) puede recibir un *peso*, que puede representar el costo para ir de u a v y se representa por $c_{u,v}$
- ▶ A dichos grafos se les conoce como grafos con pesos
- ▶ Se conoce como grafo *conectado* a todo grafo cuyos vértices tienen al menos un camino entre sí

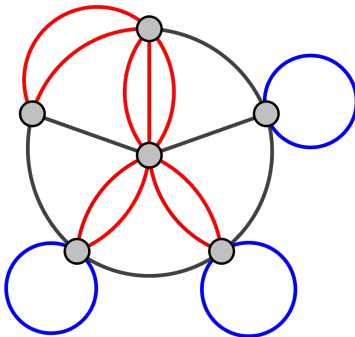
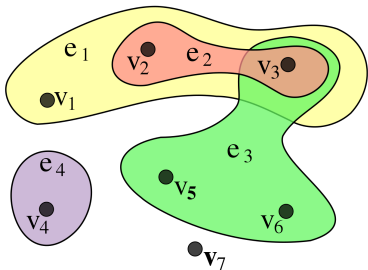
Grafo

- ▶ Adicionalmente una arista (u, v) puede recibir un *peso*, que puede representar el costo para ir de u a v y se representa por $c_{u,v}$
- ▶ A dichos grafos se les conoce como grafos con pesos
- ▶ Se conoce como grafo *conectado* a todo grafo cuyos vértices tienen al menos un camino entre sí
- ▶ Un grafo cuyos vértices representan puntos en el plano se les conocen como grafos geométricos

Grafo

- ▶ Adicionalmente una arista (u, v) puede recibir un *peso*, que puede representar el costo para ir de u a v y se representa por $c_{u,v}$
- ▶ A dichos grafos se les conoce como grafos con pesos
- ▶ Se conoce como grafo *conectado* a todo grafo cuyos vértices tienen al menos un camino entre sí
- ▶ Un grafo cuyos vértices representan puntos en el plano se les conocen como grafos geométricos
- ▶ Grafos que contienen más de una arista por par de vértices se le conoce como multi-grafo. Grafos cuyas aristas conectan más de dos vértices se les conocen como hipergrafos.

Multigrafos e Hipergrafos



Fuente: Wikimedia. Creative Commons license.

Grafo

- Un grafo conectado con $|V| - 1$ aristas es un árbol

Grafo

- ▶ Un grafo conectado con $|V| - 1$ aristas es un árbol
- ▶ Un árbol es un caso particular de un grafo

Grafo

- ▶ Un grafo conectado con $|V| - 1$ aristas es un árbol
- ▶ Un árbol es un caso particular de un grafo
- ▶ Un grafo que tiene todas las posibles aristas se conoce como grafo *completo*

Grafo

- ▶ Un grafo conectado con $|V| - 1$ aristas es un árbol
- ▶ Un árbol es un caso particular de un grafo
- ▶ Un grafo que tiene todas las posibles aristas se conoce como grafo *completo*
- ▶ Un grafo es llamado de *denso* si es que $|E| \gg |V|$ y es llamado de *esparso* si es que $|E| < |V|$. Esta definición es a veces difusa y depende del problema.

Grafo

- ▶ Un grafo conectado con $|V| - 1$ aristas es un árbol
- ▶ Un árbol es un caso particular de un grafo
- ▶ Un grafo que tiene todas las posibles aristas se conoce como grafo *completo*
- ▶ Un grafo es llamado de *denso* si es que $|E| \gg |V|$ y es llamado de *esparso* si es que $|E| < |V|$. Esta definición es a veces difusa y depende del problema.
- ▶ Alternativamente, un grafo es esparso si $|E| = O(|V|)$ o $|E| = O(|V| \log |V|)$

Contenido

Introducción

Representación

Búsqueda en grafos

BFS

DFS

Árboles

Lista de adyacencia

- La representación básica de un grafo es con una lista de adyacencia o con una matriz de adyacencia

Lista de adyacencia

- ▶ La representación básica de un grafo es con una lista de adyacencia o con una matriz de adyacencia
- ▶ Una lista de adyacencia es un vector, con un elemento por vértice i . El elemento i -ésimo almacena a su vez otro vector que contiene la lista de vértices con los cuales el vértice i está conectado

Lista de adyacencia

- ▶ La representación básica de un grafo es con una lista de adyacencia o con una matriz de adyacencia
- ▶ Una lista de adyacencia es un vector, con un elemento por vértice i . El elemento i -ésimo almacena a su vez otro vector que contiene la lista de vértices con los cuales el vértice i está conectado
- ▶ Para el grafo mostrado anteriormente, la lista se verá: $G = \{[D, B, C], [A, C], [B, A, D, F], [A, C, E, F], [D, F], [D, E]\}$

Lista de adyacencia

- ▶ La representación básica de un grafo es con una lista de adyacencia o con una matriz de adyacencia
- ▶ Una lista de adyacencia es un vector, con un elemento por vértice i . El elemento i -ésimo almacena a su vez otro vector que contiene la lista de vértices con los cuales el vértice i está conectado
- ▶ Para el grafo mostrado anteriormente, la lista se verá: $G = \{[D, B, C], [A, C], [B, A, D, F], [A, C, E, F], [D, F], [D, E]\}$
- ▶ Para un grafo con pesos, cada elemento del vector puede contener un par, indicando el vértice al que se conecta la arista y el peso de dicha arista. Note que no hay ningún orden en particular al momento de guardar los vértices

Matriz de adyacencia

- ▶ Una matriz de adyacencia es una matriz, tal que la arista u, v será representada por un valor de 1 en la fila u y la columna v

Matriz de adyacencia

- Una matriz de adyacencia es una matriz, tal que la arista u, v será representada por un valor de 1 en la fila u y la columna v

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

Matriz de adyacencia

- Una matriz de adyacencia es un matriz, tal que la arista u, v será representada por un valor de 1 en la fila u y la columna v

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

- Para un grafo con pesos puede guardarse el peso directamente en la matriz. Note que la matriz tiene una diagonal con ceros

Matriz de adyacencia

- Una matriz de adyacencia es un matriz, tal que la arista u, v será representada por un valor de 1 en la fila u y la columna v

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

- Para un grafo con pesos puede guardarse el peso directamente en la matriz. Note que la matriz tiene una diagonal con ceros
- Un grafo no direccionado tiene una matriz simétrica

Representaciones

- ▶ Estás no son las únicas maneras de representar un grafo, aunque sí las más comunes

Representaciones

- ▶ Estás no son las únicas maneras de representar un grafo, aunque sí las más comunes
- ▶ El uso depende de la situación, algunos algoritmos o problemas se facilitan con una u otra representación

Representaciones

- ▶ Estás no son las únicas maneras de representar un grafo, aunque sí las más comunes
- ▶ El uso depende de la situación, algunos algoritmos o problemas se facilitan con una u otra representación
- ▶ Para un grafo no direccionado completo, ambas representaciones serán iguales

Representaciones

- ▶ Estás no son las únicas maneras de representar un grafo, aunque sí las más comunes
- ▶ El uso depende de la situación, algunos algoritmos o problemas se facilitan con una u otra representación
- ▶ Para un grafo no direccionado completo, ambas representaciones serán iguales
- ▶ La lista de adyacencia es más eficiente en el uso de memoria

Representaciones

- ▶ Estas no son las únicas maneras de representar un grafo, aunque sí las más comunes
- ▶ El uso depende de la situación, algunos algoritmos o problemas se facilitan con una u otra representación
- ▶ Para un grafo no direccionado completo, ambas representaciones serán iguales
- ▶ La lista de adyacencia es más eficiente en el uso de memoria
- ▶ Grafos densos son naturalmente mejor representados por una matriz de adyacencia; grafos esparsos por listas de adyacencia

Contenido

Introducción

Representación

Búsqueda en grafos

BFS

DFS

Árboles

Búsqueda

- ▶ Es una forma de visitar todos los vértices de un grafo hasta encontrar el valor deseado

Búsqueda

- ▶ Es una forma de visitar todos los vértices de un grafo hasta encontrar el valor deseado
- ▶ A diferencia de una arreglo, los grafos pueden ser recorridos de diversas maneras

Búsqueda

- ▶ Es una forma de visitar todos los vértices de un grafo hasta encontrar el valor deseado
- ▶ A diferencia de una arreglo, los grafos pueden ser recorridos de diversas maneras
- ▶ Cada forma tiene sus ventajas y sus aplicaciones

Breadth First Search (BFS): Búsqueda en amplitud

- ▶ Se comienza en un vértice v y se avanza como una “gota de agua”, recorriendo primero los vecinos de v , luego los vértices que están a dos saltos de v , luego los que están a tres saltos, y así hasta recorrer todo el grafo.

Breadth First Search (BFS): Búsqueda en amplitud

- ▶ Se comienza en un vértice v y se avanza como una “gota de agua”, recorriendo primero los vecinos de v , luego los vértices que están a dos saltos de v , luego los que están a tres saltos, y así hasta recorrer todo el grafo.
- ▶ La complejidad es $O(|V| + |E|)$

Breadth First Search (BFS): Búsqueda en amplitud

```
toVisit.push(u);      // toVisit puede ser una cola y se
                        comienza en un vértice arbitrario u
while !toVisit.empty() do
    u = toVisit.front() /* Adj(u) adyacentes a u          */
    for v ∈ Adj(u) do
        /* colorear los vértices                          */
        if !visited(v) then
            toVisit.push(v);
            dist[v] = dist [u] + 1 ;
            mark u as visited ;
        end
    end
    toVisit.pop()
end
```

Breadth First Search (BFS): Búsqueda en amplitud

- ¿Qué pasa si el grafo no esta conectado?

Breadth First Search (BFS): Búsqueda en amplitud

- ▶ ¿Qué pasa si el grafo no está conectado?
- ▶ ¿Cómo hallar la distancia mínima entre dos vértices para un grafo sin pesos?

Depth First Search (DFS): Búsqueda en profundidad

- ▶ Se comienza desde un vértice v y se trata de llegar lo más lejos posible en número de saltos, luego se regresa hasta el último nodo visitado que tenga vecinos aún sin visitar, y se vuelve a intentar llegar lo más lejos posible en número de saltos hasta recorrer todo el grafo.

Depth First Search (DFS): Búsqueda en profundidad

- ▶ Se comienza desde un vértice v y se trata de llegar lo más lejos posible en número de saltos, luego se regresa hasta el último nodo visitado que tenga vecinos aún sin visitar, y se vuelve a intentar llegar lo más lejos posible en número de saltos hasta recorrer todo el grafo.
- ▶ La complejidad es $O(|V| + |E|)$

Depth First Search (DFS): Búsqueda en profundidad

```
mark  $u$  as visited ;  
for  $v \in Adj(u)$  do  
  | if ! $visited(v)$  then  
  |   |  $DFS(v)$   
  | end  
end
```

Depth First Search (DFS): Búsqueda en profundidad

- ▶ ¿Qué pasa si el grafo no esta conectado?

Depth First Search (DFS): Búsqueda en profundidad

- ▶ ¿Qué pasa si el grafo no está conectado?
- ▶ ¿Cómo podríamos hacer el ordenamiento topológico en un grafo direccionado?

Depth First Search (DFS): Búsqueda en profundidad

- ▶ ¿Qué pasa si el grafo no está conectado?
- ▶ ¿Cómo podríamos hacer el ordenamiento topológico en un grafo direccionado?
- ▶ ¿Cómo hallamos los componentes fuertemente conectados para un grafo direccionado?

Árboles

- Un árbol es un grafo acíclico, conectado y no dirigido.

Árboles

- ▶ Un árbol es un grafo acíclico, conectado y no dirigido.
- ▶ Alternativamente, un árbol puede ser definido como un grafo no dirigido en el que existe exactamente un camino entre cualquier par de nodos.

Árboles

- ▶ Un árbol es un grafo acíclico, conectado y no dirigido.
- ▶ Alternativamente, un árbol puede ser definido como un grafo no dirigido en el que existe exactamente un camino entre cualquier par de nodos.
- ▶ Propiedades de un árbol:
 1. Un árbol con n nodos tiene exactamente $n - 1$ aristas

Árboles

- ▶ Un árbol es un grafo acíclico, conectado y no dirigido.
- ▶ Alternativamente, un árbol puede ser definido como un grafo no dirigido en el que existe exactamente un camino entre cualquier par de nodos.
- ▶ Propiedades de un árbol:
 1. Un árbol con n nodos tiene exactamente $n - 1$ aristas
 2. Si una única arista es agregada a un árbol, el grafo resultante contiene exactamente un ciclo

Árboles

- ▶ Un árbol es un grafo acíclico, conectado y no dirigido.
- ▶ Alternativamente, un árbol puede ser definido como un grafo no dirigido en el que existe exactamente un camino entre cualquier par de nodos.
- ▶ Propiedades de un árbol:
 1. Un árbol con n nodos tiene exactamente $n - 1$ aristas
 2. Si una única arista es agregada a un árbol, el grafo resultante contiene exactamente un ciclo
 3. Si una única arista es eliminada de un árbol, el grafo resultante no está conectado