DECSAI
Departamento de Ciencias de la Computación e I.A.
Universidad de Granada

Exploración de grafos
Análisis y Diseño de Algoritmos

Exploración de grafos

- Grafos
- Recorridos sobre grafos
  - Búsqueda primero en profundidad
  - Búsqueda primero en anchura
- Backtracking ("vuelta atrás")
  - Descripción general
  - Espacio de soluciones
  - Implementación
  - Ejemplos
- Branch & Bound ("ramificación y poda")
  - Descripción general
  - Estrategias de ramificación
  - Implementación
  - Ejemplos

Grafos

Grafo G = (V,E)

- V: Conjunto de vértices o nodos del grafo.
- $E \subseteq VxV$: Conjunto de aristas o arcos.

Tipos de grafos

- **Grafos no dirigidos**: Aristas (no orientadas).

  $(v,w) = (w,v)$

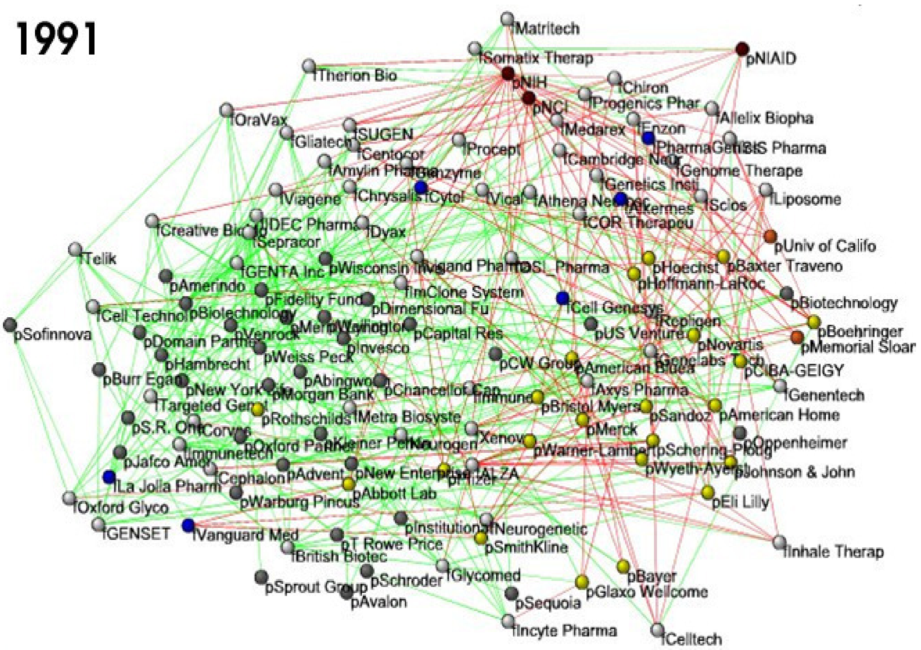
- **Grafos dirigidos**: Arcos (con dirección).

  $(v,w) \neq (w,v)$

Grafos

Ejemplo: US Biotech Industry

1991



Grafos - Definiciones

Grafos no dirigidos

- **Grado** de un vértice:
  Número de aristas que lo contienen.

Grafos dirigidos

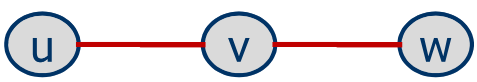
- **Grado de salida** de un vértice v:
  Número de arcos cuyo vértice inicial es v.

- **Grado de entrada** de un vértice v:
  Número de arcos cuyo vértice final es v.

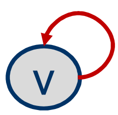
Grafos - Definiciones

- **Nodos/vértices adyacentes**:
  Vértices conectados por una arista (o un arco).

- **Aristas/arcos adyacentes**:
  Arcos/aristas con un vértice común.



Bucle: Arco/arista cuyos vértices inicial y final coinciden.



Grafos - Definiciones

- **Camino** [path]:
  Sucesión de arcos adyacentes tal que el vértice final de cada arco coincide con el inicial del siguiente.

  Secuencia $(w_1, w_2, ..., w_k) \in V$
  tal que $(w_1, w_2), (w_2, w_3), ..., (w_{k-1}, w_k) \in E$.

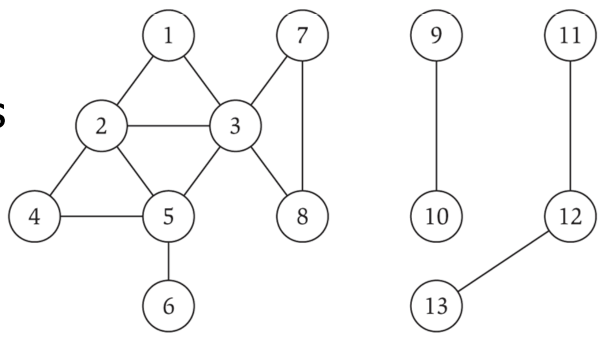
  - Longitud del camino:
    Número de arcos del camino (k-1).

- **Circuito** (o ciclo):
  Camino que empieza y acaba en el mismo vértice.

Grafos - Definiciones

- **Grafo conexo**:
  Un grafo no dirigido es un grafo conexo si para todo par de nodos u y v existe una camido de u a v.

- **Componentes conexas**:
  Cada uno de los conjuntos maximales conexos.



Grafos - Definiciones

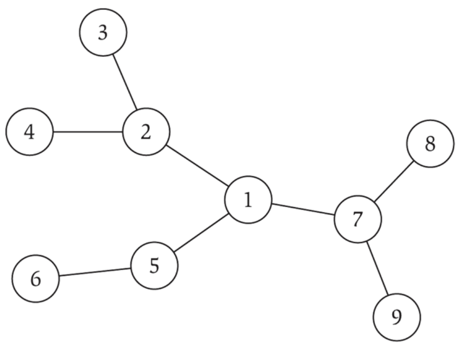
Tipos de grafos

- **Grafo etiquetado**:
  Cada arista y/o vértice tiene asociada una etiqueta/valor.
- **Grafo ponderado** = Grafo con pesos:
  Grafo etiquetado en el que existe un valor numérico asociado a cada arista o arco.
- **Multigrafo**:
  Grafo en el que se permite que entre dos vértices exista más de una arista o arco.

Grafos - Definiciones

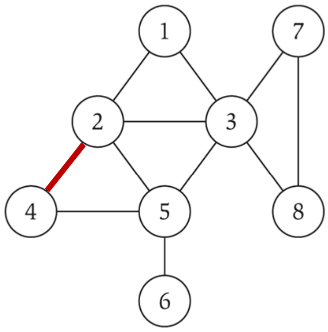
- **Árbol**:
  Grafo conexo que no contiene ciclos.

  Teorema
  Sea G un grafo de n nodos. Cualquier pareja de las siguientes afirmaciones implica la tercera:
  - G es conexo.
  - G no contiene ciclos.
  - G tiene n-1 aristas.



Grafos - Representación

Representación mediante matrices de adyacencia



	1	2	3	4	5	6	7	8
1	0	1	1	0	0	0	0	0
2	1	0	1	1	1	0	0	0
3	1	1	0	0	1	0	1	1
4	0	1	0	1	1	0	0	0
5	0	1	1	1	0	1	0	0
6	0	0	0	0	1	0	0	0
7	0	0	1	0	0	0	0	1
8	0	0	1	0	0	0	1	0

Matriz de adyacencia de tamaño |V|x|V|

- A[u][v] = 1 si $(u, v) \in E$.
- A[u][v] = 0 si $(u, v) \notin E$.

Grafos - Representación

Representación mediante matrices de adyacencia

Ventaja:

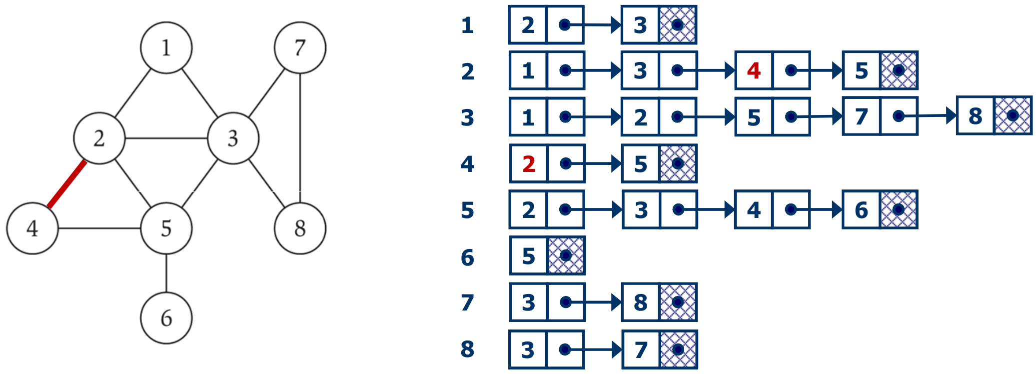
- Acceso eficiente a una arista, $\Theta(1)$.

Inconvenientes:

- $\Theta(|V|^2)$ en identificar todas las aristas.
- Espacio proporcional a $|V|^2$
  (se desperdicia memoria si el grafo es poco denso).

Grafos - Representación

Representación mediante listas de adyacencia



Array de listas enlazadas de nodos adyacentes.

Grafos - Representación

Representación mediante listas de adyacencia

Ventajas:

- Espacio proporcional a |V|+|E|.
  (representación adecuada para grafos poco densos).
- $\Theta(|V|+|E|)$ en identificar todas las aristas.

Inconvenientes:

- Se tarda O(grado(u)) en comprobar si $(u,v) \in E$.
- Ineficiente para encontrar los arcos que llegan a un nodo (solución: usar estructuras de listas múltiples).

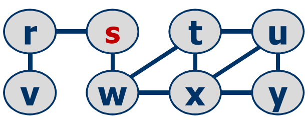
Recorridos sobre grafos

Se parte de un nodo dado y se visitan los vértices del grafo de manera ordenada y sistemática, pasando de un vértice a otro a través de las aristas del grafo.

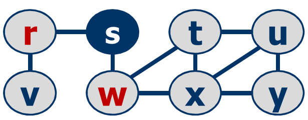
Tipos de recorridos:

- **Búsqueda primero en profundidad:**
  Equivalente al recorrido en preorden de un árbol.
- **Búsqueda primero en anchura:**
  Equivalente al recorrido de un árbol por niveles.

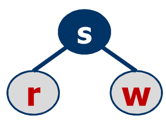
Búsqueda primero en profundidad
[DFS: Depth-First Search]

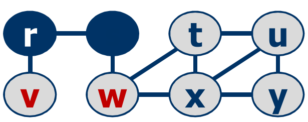


Pila S={}

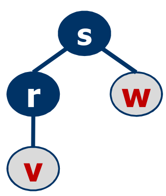


Pila S={s}





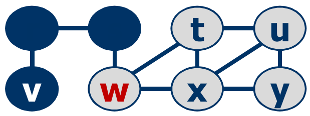
Pila S={r,s}



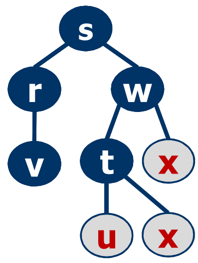
Recorridos sobre grafos

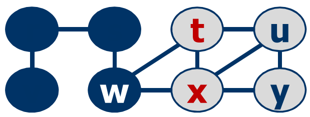
Búsqueda primero en profundidad

[DFS: Depth-First Search]

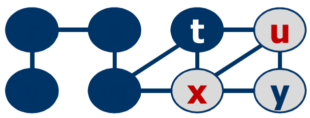


Pila S={v,r,s}





Pila S={w,s}

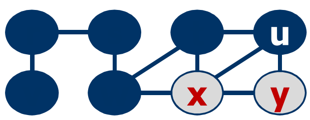


Pila S={t,w,s}

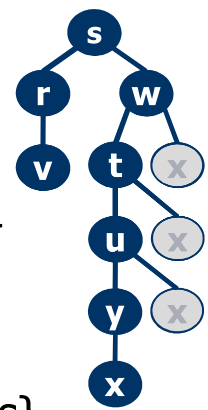
Recorridos sobre grafos

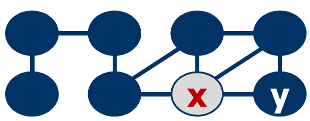
Búsqueda primero en profundidad

[DFS: Depth-First Search]

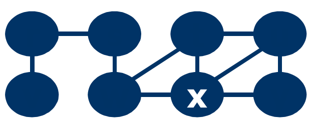


Pila S={u,t,w,s}





Pila S={y,u,t,w,s}



Pila S={x,y,u,t,w,s}

Recorridos sobre grafos

Búsqueda primero en profundidad
[DFS: Depth-First Search]

- Es necesario llevar la cuenta de los nodos visitados (y no visitados).
- El recorrido **no es único**: depende del vértice inicial y del orden de visita de los vértices adyacentes.
- El orden de visita de unos nodos puede interpretarse como un árbol: **árbol de expansión en profundidad** asociado al grafo.

Recorridos sobre grafos

Búsqueda primero en profundidad
[DFS: Depth-First Search]

```
función DFS (Grafo G(V,E))
{
  for (i=0; i<V.length; i++)
      visitado[i] = false;

  for (i=0; i<V.length; i++)
      if (!visitado[i])
         DFS(G,i);
}
```

Recorridos sobre grafos

Búsqueda primero en profundidad

[DFS: Depth-First Search]

```
función DFS (Grafo G(V,E), int i)
{
  visitado[i] = true;

  foreach (v[j] adyacente a v[i])
          if (!visitado[j])
             DFS(G,j);
}
```

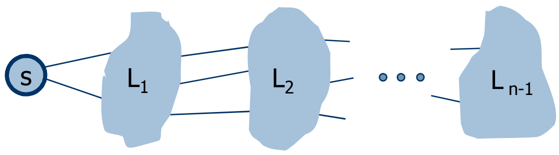
$O(|V|+|E|)$

si usamos la representación basada en listas de adyacencia.

Recorridos sobre grafos

Búsqueda primero en anchura

[BFS: Breadth-First Search]



Exploración desde s por niveles:

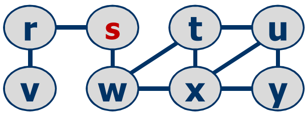
- L_0 = { s }.
- L_1 = Nodos adyacentes a L_0.
- L_2 = Nodos adyacentes a L_1 que no pertenecen ni a L_0 ni a L_1.
- L_{i+1} = Nodos que, sin pertenecer a ningún nivel anterior, están conectados con L_i a través de una arista.

Teorema:

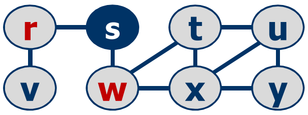
L_i contiene todos los nodos que están a distancia i de s.

Recorridos sobre grafos

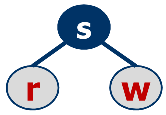
Búsqueda primero en anchura
[BFS: Breadth-First Search]

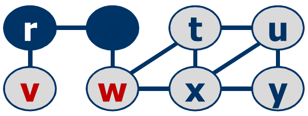


Cola Q={s}

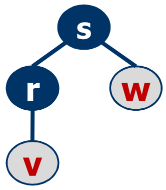


Cola Q={r,w}



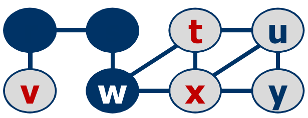


Cola Q={w,v}

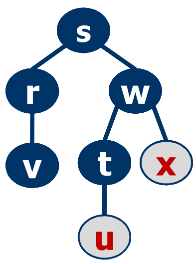


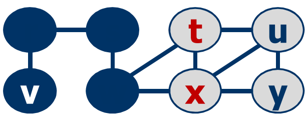
Recorridos sobre grafos

Búsqueda primero en anchura
[BFS: Breadth-First Search]

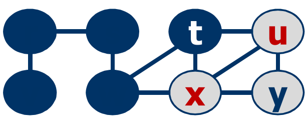


Cola Q={v,t,x}





Cola Q={t,x}

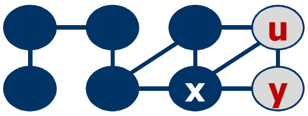


Cola Q={x,u}

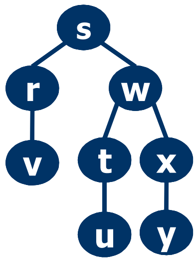
Recorridos sobre grafos

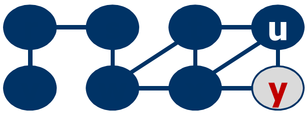
Búsqueda primero en anchura

[BFS: Breadth-First Search]

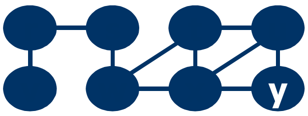


Cola Q={u,y}





Cola Q={y}



Cola Q={}

Recorridos sobre grafos

Búsqueda primero en anchura

[BFS: Breadth-First Search]

- Empezando en un nodo v:
  - Primero se visita v.
  - Luego se visitan todos sus vértices adyacentes.
  - A continuación, los adyacentes a éstos…
    y así sucesivamente.
- El algoritmo utiliza una **cola de vértices.**

Recorridos sobre grafos

Búsqueda primero en anchura
[BFS: Breadth-First Search]

```
función BFS (Grafo G(V,E))
{
  for (i=0; i<V.length; i++)
      visitado[i] = false;

  for (i=0; i<V.length; i++)
      if (!visitado[i])
         BFS(G,i);
}
```


Recorridos sobre grafos

Búsqueda primero en anchura
[BFS: Breadth-First Search]

O(|V|+|E|)
si usamos la representación basada en listas de adyacencia

```
función BFS (Grafo G(V,E), int v)
{
  Cola Q;

  visitado[v]=true; Q.add(v);

  while (!Q.empty()) {
        x = Q.extract();
        foreach (v[y] adyacente a v[x])
                if (!visitado[y]) {
                   visitado[y]=true; Q.add(y);
                }
  }
}
```

Recorridos sobre grafos

Aplicaciones de los recorridos sobre grafos

- **Comprobar si un grafo es bipartito:** Un grafo no dirigido G=(V,A) es bipartito si sus vértices se pueden separar en dos conjuntos disjuntos V_1 y V_2 ($V=V_1 \cup V_2$, $V_1 \cap V_2 = \emptyset$) de tal forma que todas las aristas de G unen vértices de un conjunto con vértices de otro.
- **Detección de las componentes fuertemente conexas de un grafo:** Una componente fuertemente conexa de un grafo G=(V,A) es el máximo conjunto de vértices $U \subseteq V$ tal que para cada par de vértices $u, v \in U$, existen caminos en G desde u hasta v y viceversa.

Recorridos sobre grafos

Aplicaciones de los recorridos sobre grafos

- **"Flood fill"** (coloreado por inundación)