Recorridos en grafos: BFS vs. DFS

Algoritmo genérico de recorrido

Objetivos:

- Dado un nodo de partida, explorar tantos nodos restantes como sea posible
- No explorar ningún nodo más de una vez, es decir, lograr $O(M+N)$

```
function searchAlgorithm(grafo G, nodo n):
    n = explorado, todos los demás inexplorados
    mientras sea posible
        elegir una arista (u, v), siendo u el nodo actual y v un
        nodo no explorado, parar en caso de no haber ninguno
        v = explorado
```

Funciona tanto para grafos dirigidos como no dirigidos

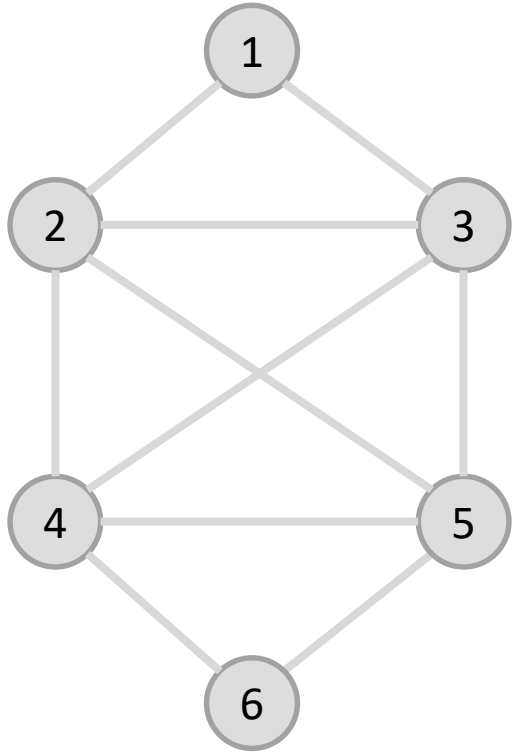
BFS versus DFS

¿Cómo elegir la arista (*u*,*v*) a partir del nodo actual *u*?

BFS (Breadth-First Search) o búsqueda en amplitud	**DFS (Depth-First Search) o búsqueda en profundidad**
• Explora los nodos por “capas” • Puede ejecutarse en *O*(*M*+*N*) usando una cola (FIFO) • Puede calcular “naturalmente” cantidad mínima de saltos entre nodos • Puede calcular los componentes conectados de un grafo no dirigido	• Explora los nodos “agresivamente” • Puede ejecutarse en *O(M+N)* usando una pila (LIFO) o usando recursión • Puede calcular los componentes fuertemente conectados de un grafo dirigido

BFS

```
function BFS(grafo G, nodo a):
   for each vertex u:
      u.explored = false
   a.explored = true
   q = new queue
   q.push(a)
   while (q is not empty):
      u = q.pop()
      for each edge (u, v){
         if v.explored = false{
            v.explored = true
            q.push(v)
```

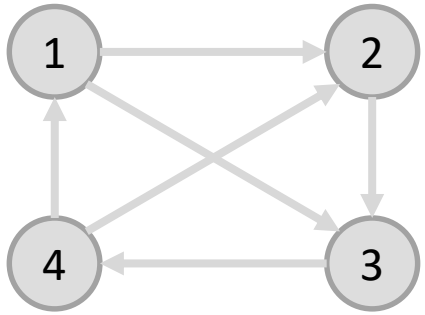


DFS

Versión recursiva

```
for each vertex u:
   u.explored = false

function DFS(grafo G, nodo u):
   u.explored = true
   for each edge (u, v):
      if v.explored = false:
         DFS(G, v)
```



Versión iterativa con pila

```
function DFS(grafo G, nodo a):
   for each vertex u:
      u.explored = false
   s = new stack
   s.push(a)
   while (s is not empty):
      u = s.pop()
      if u.explored = false:
         u.explored = true
         for each edge (u, v):
            s.push(v)
```