RECORRIDO EN PROFUNDIDAD

UNIVERSIDAD
COMPLUTENSE
MADRID

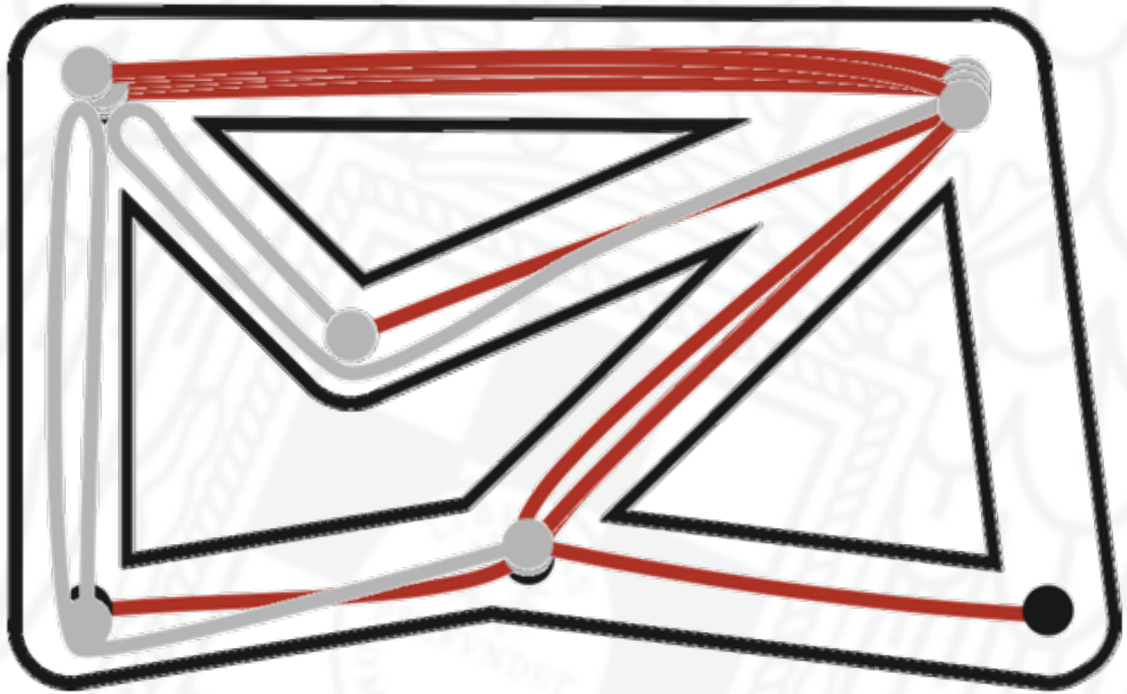
ALBERTO VERDEJO

Recorridos

- Muchas propiedades de los grafos pueden conocerse explorando sistemáticamente sus vértices y aristas.
- Algunas propiedades pueden averiguarse simplemente analizando todas las aristas.
- Muchas otras propiedades están relacionadas con caminos, por lo que interesa explorar el grafo siguiendo aristas. Es lo que se conoce como recorridos.

Recorrido en profundidad

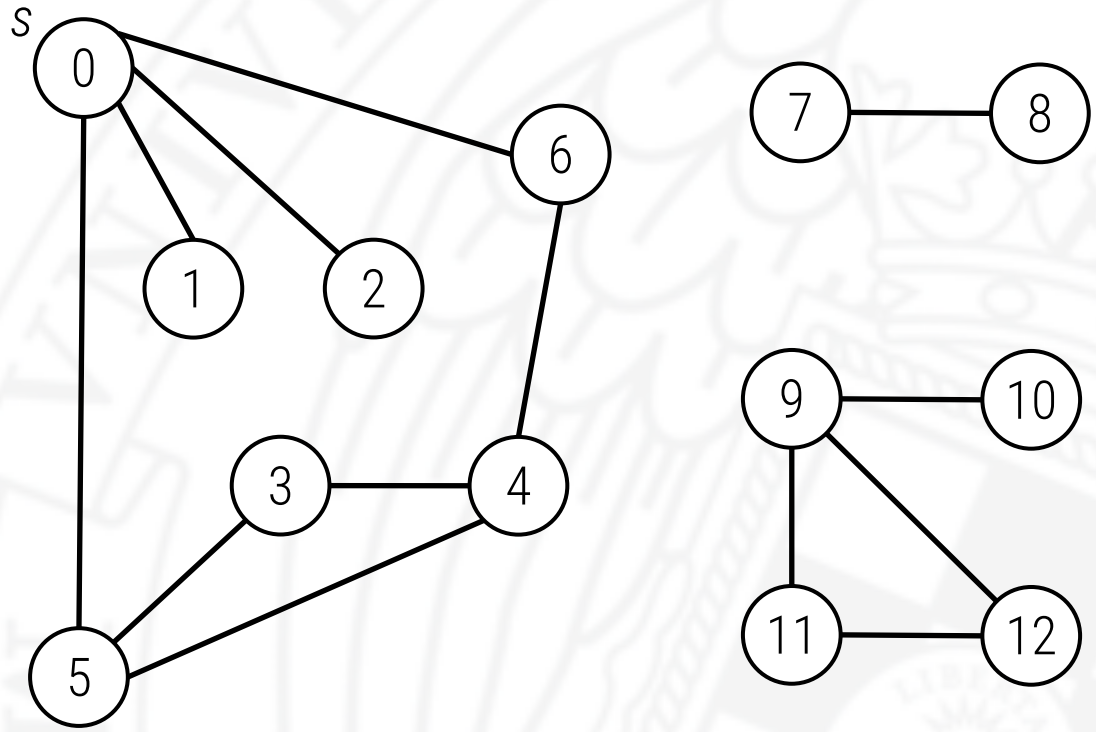
- El recorrido o búsqueda en profundidad (en inglés, *depth-first search*) imita la resolución de un laberinto.



Recorrido en profundidad

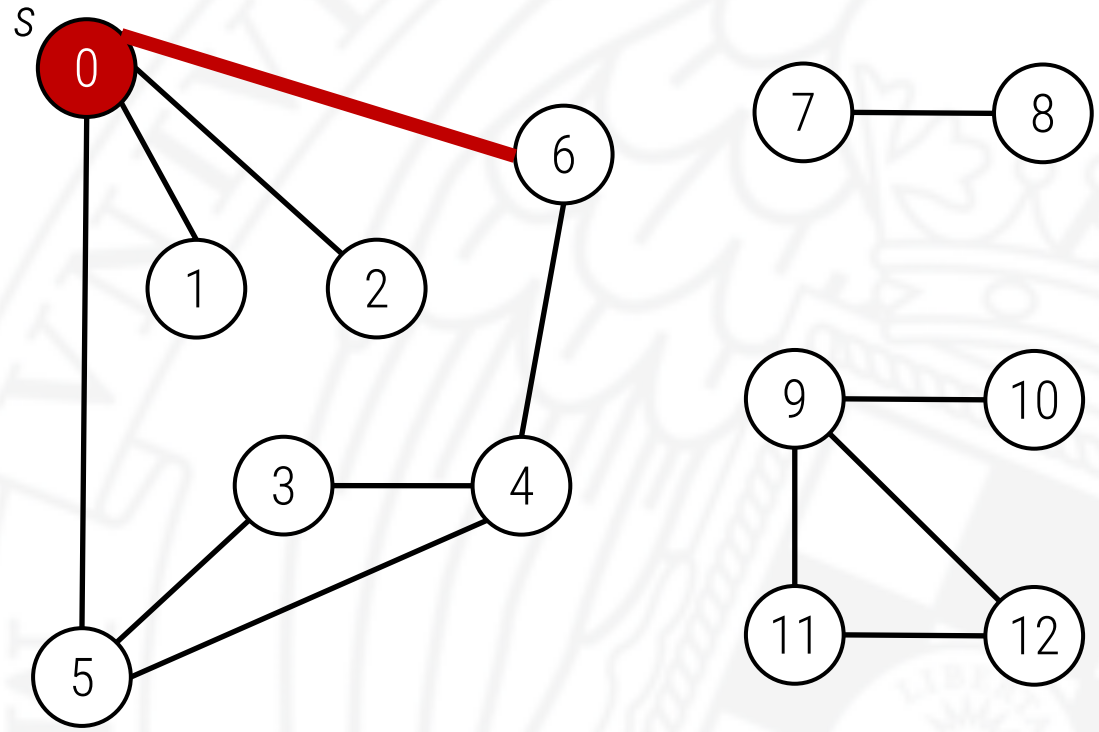
- Para recorrer un grafo utilizamos un algoritmo recursivo que va visitando vértices.
- Visitar un vértice *v* consiste en:
  - marcarlo como visitado;
  - *hacer algo con él*; y
  - visitar (recursivamente) todos los vértices adyacentes a *v* aún no visitados.

Recorrido en profundidad



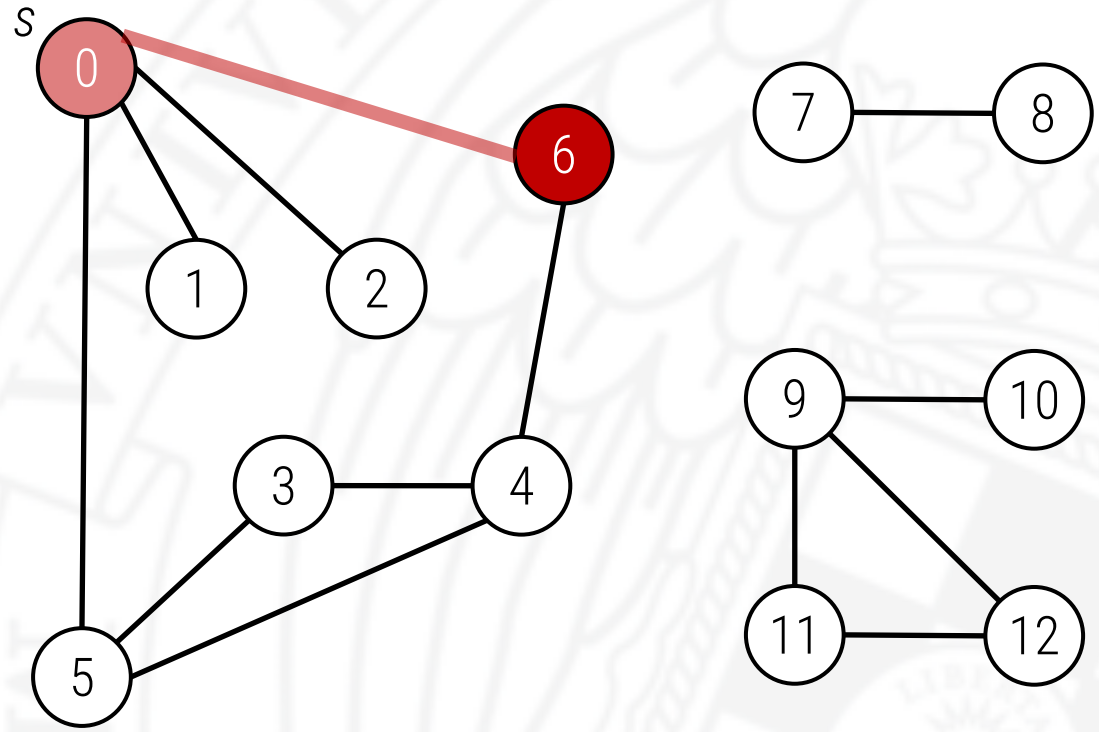
v	visitado	anterior
0	F	-
1	F	-
2	F	-
3	F	-
4	F	-
5	F	-
6	F	-
7	F	-
8	F	-
9	F	-
10	F	-
11	F	-
12	F	-

Recorrido en profundidad



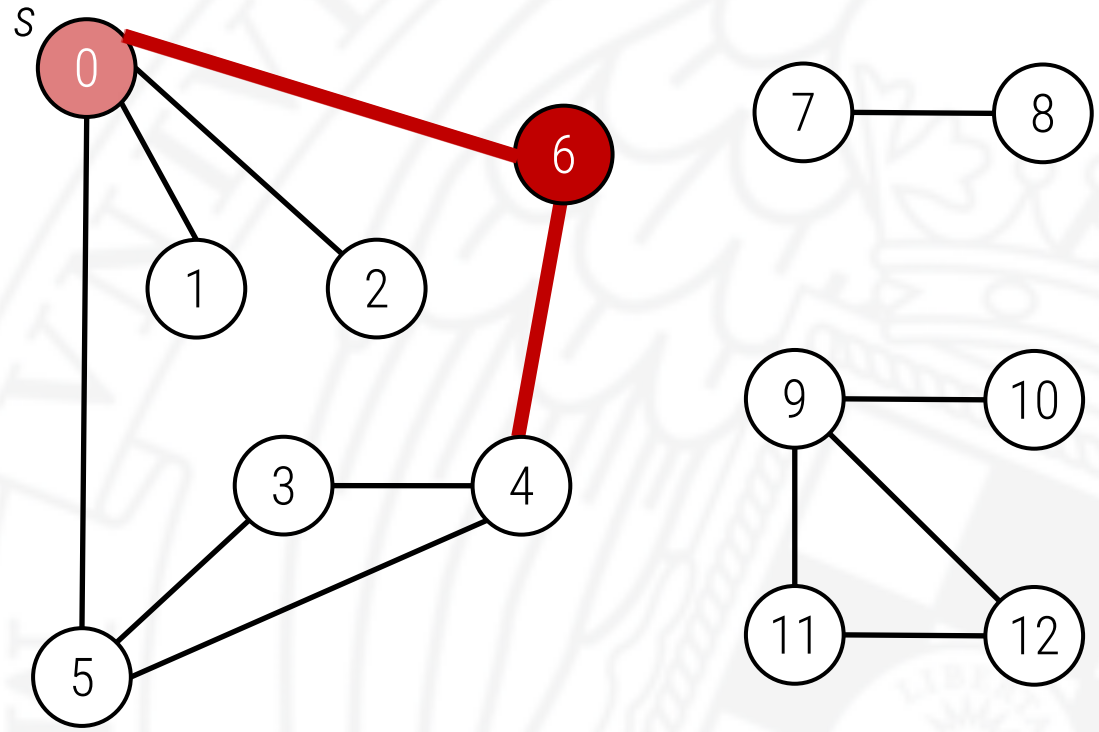
v	visitado	anterior
0	T	-
1	F	-
2	F	-
3	F	-
4	F	-
5	F	-
6	F	-
7	F	-
8	F	-
9	F	-
10	F	-
11	F	-
12	F	-

Recorrido en profundidad



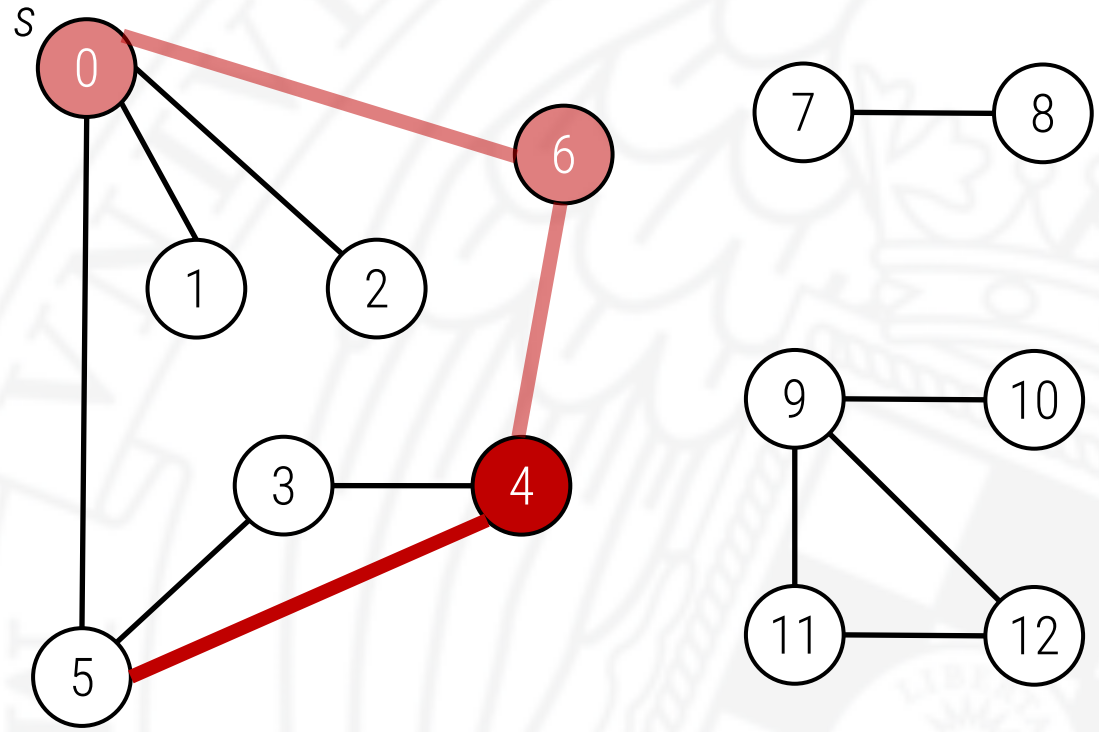
v	visitado	anterior
0	T	-
1	F	-
2	F	-
3	F	-
4	F	-
5	F	-
6	T	0
7	F	-
8	F	-
9	F	-
10	F	-
11	F	-
12	F	-

Recorrido en profundidad



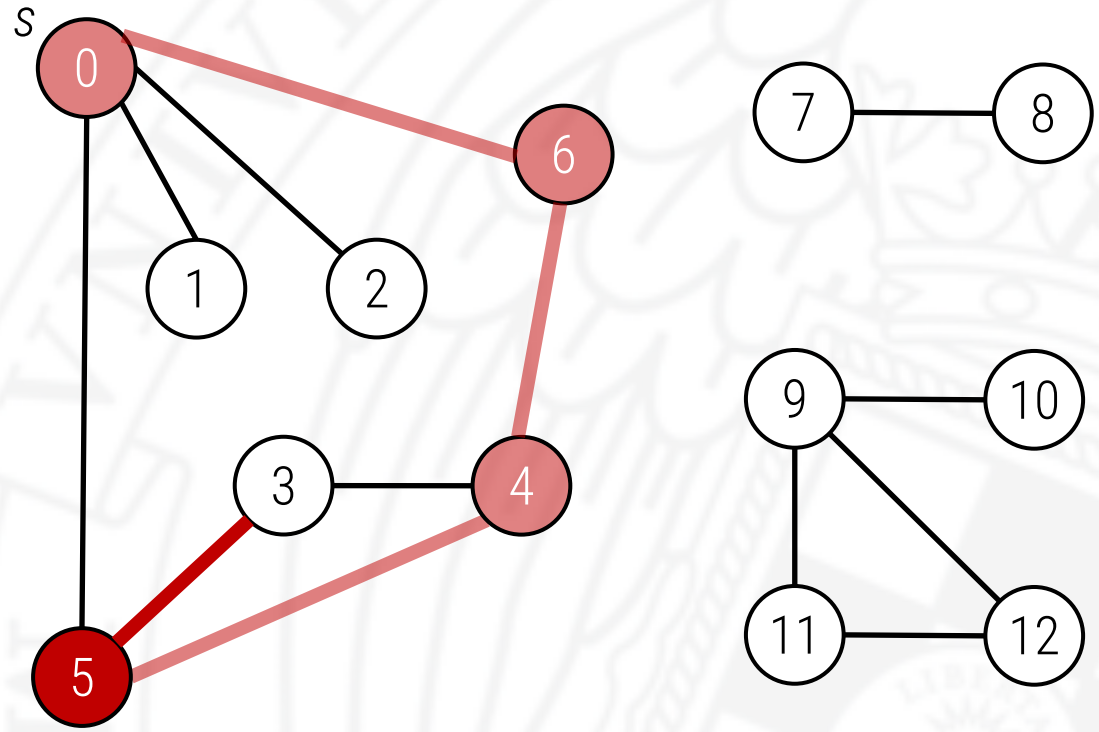
v	visitado	anterior
0	T	-
1	F	-
2	F	-
3	F	-
4	F	-
5	F	-
6	T	0
7	F	-
8	F	-
9	F	-
10	F	-
11	F	-
12	F	-

Recorrido en profundidad



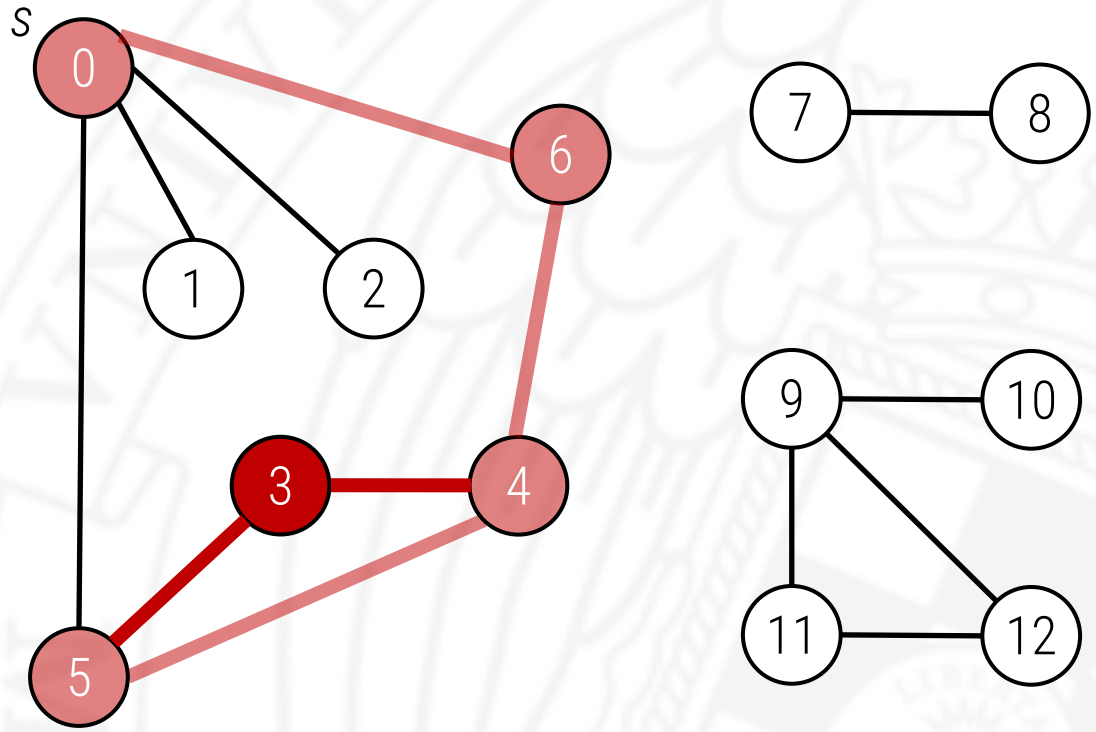
v	visitado	anterior
0	T	-
1	F	-
2	F	-
3	F	-
4	T	6
5	F	-
6	T	0
7	F	-
8	F	-
9	F	-
10	F	-
11	F	-
12	F	-

Recorrido en profundidad



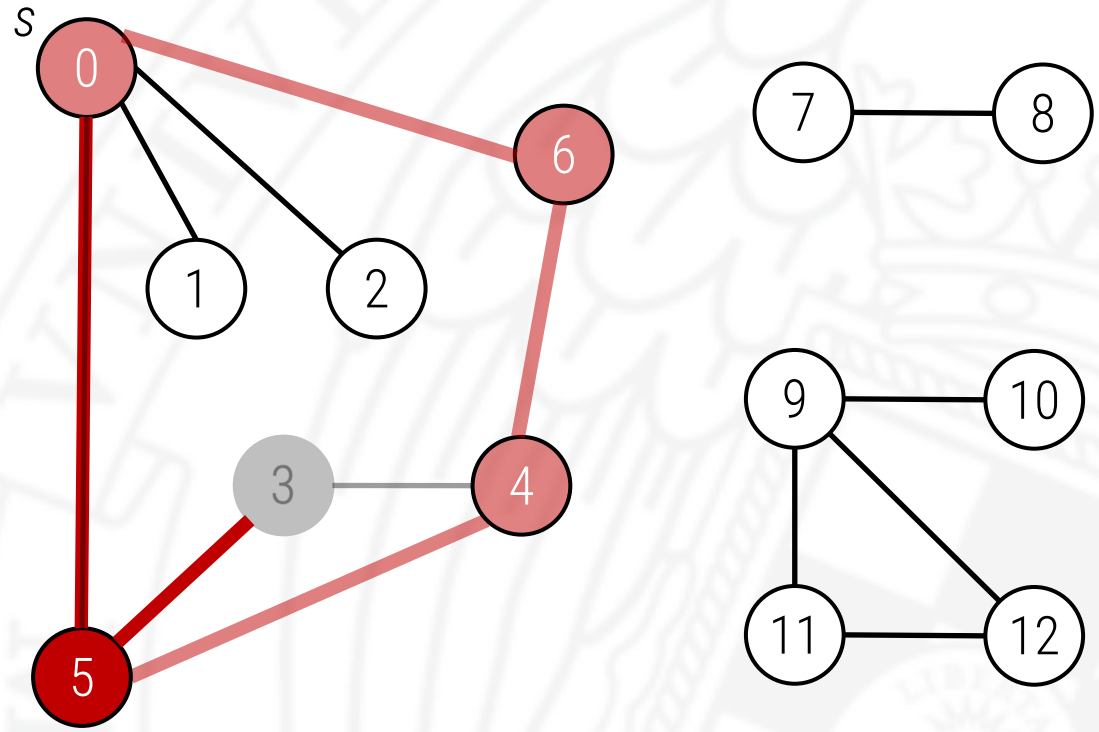
v	visitado	anterior
0	T	-
1	F	-
2	F	-
3	F	-
4	T	6
5	T	4
6	T	0
7	F	-
8	F	-
9	F	-
10	F	-
11	F	-
12	F	-

Recorrido en profundidad



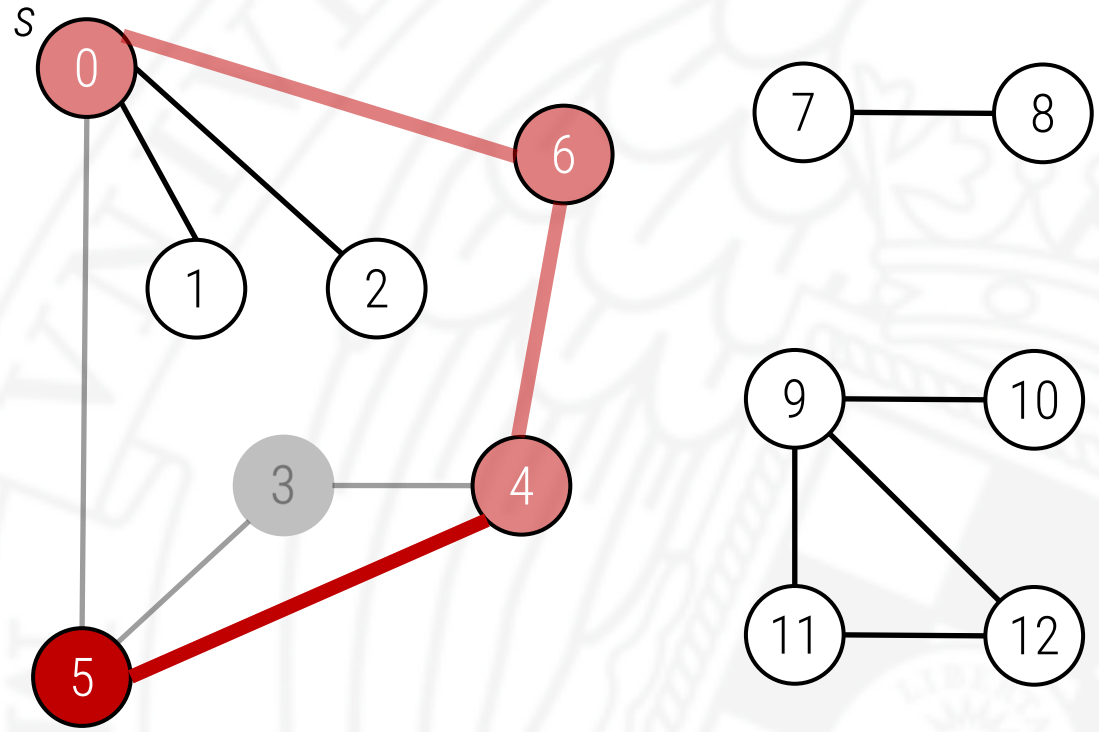
v	visitado	anterior
0	T	-
1	F	-
2	F	-
3	T	5
4	T	6
5	T	4
6	T	0
7	F	-
8	F	-
9	F	-
10	F	-
11	F	-
12	F	-

Recorrido en profundidad



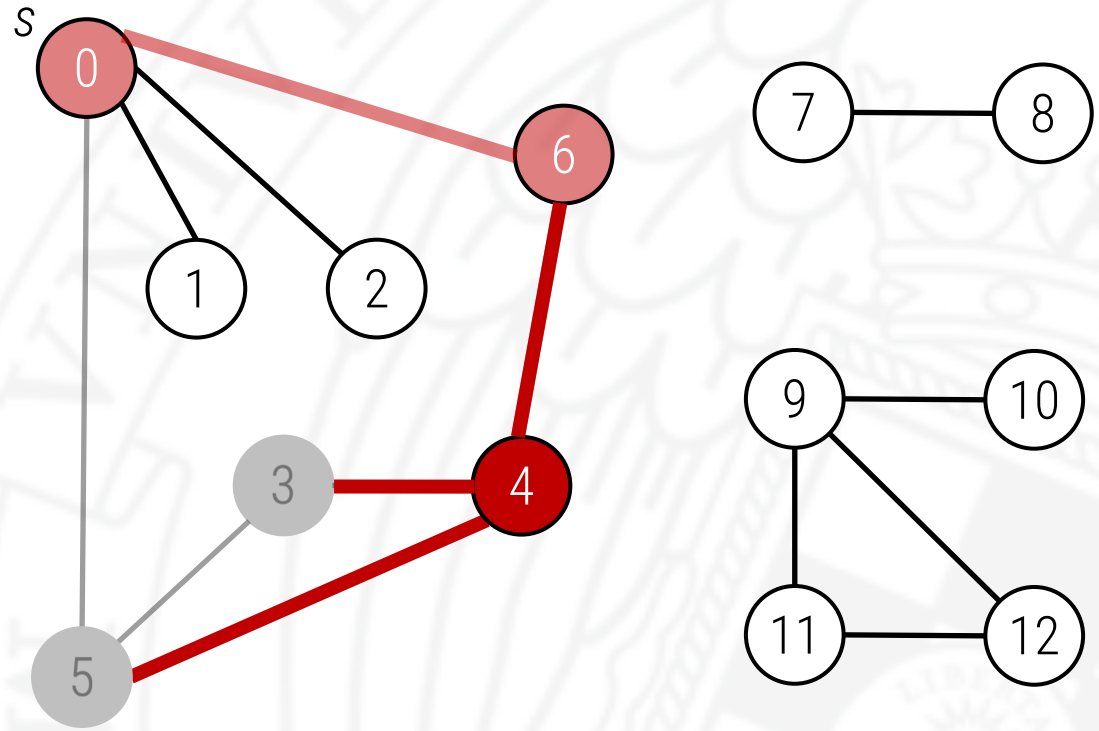
v	visitado	anterior
0	T	-
1	F	-
2	F	-
3	T	5
4	T	6
5	T	4
6	T	0
7	F	-
8	F	-
9	F	-
10	F	-
11	F	-
12	F	-

Recorrido en profundidad



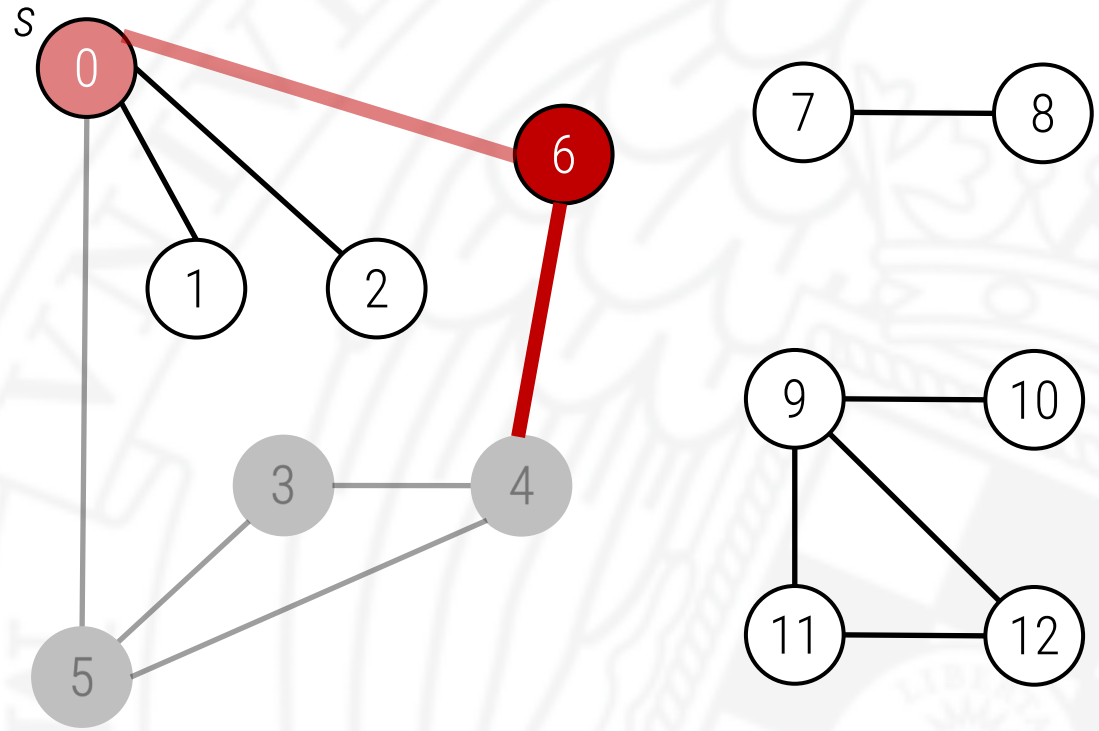
v	visitado	anterior
0	T	-
1	F	-
2	F	-
3	T	5
4	T	6
5	T	4
6	T	0
7	F	-
8	F	-
9	F	-
10	F	-
11	F	-
12	F	-

Recorrido en profundidad



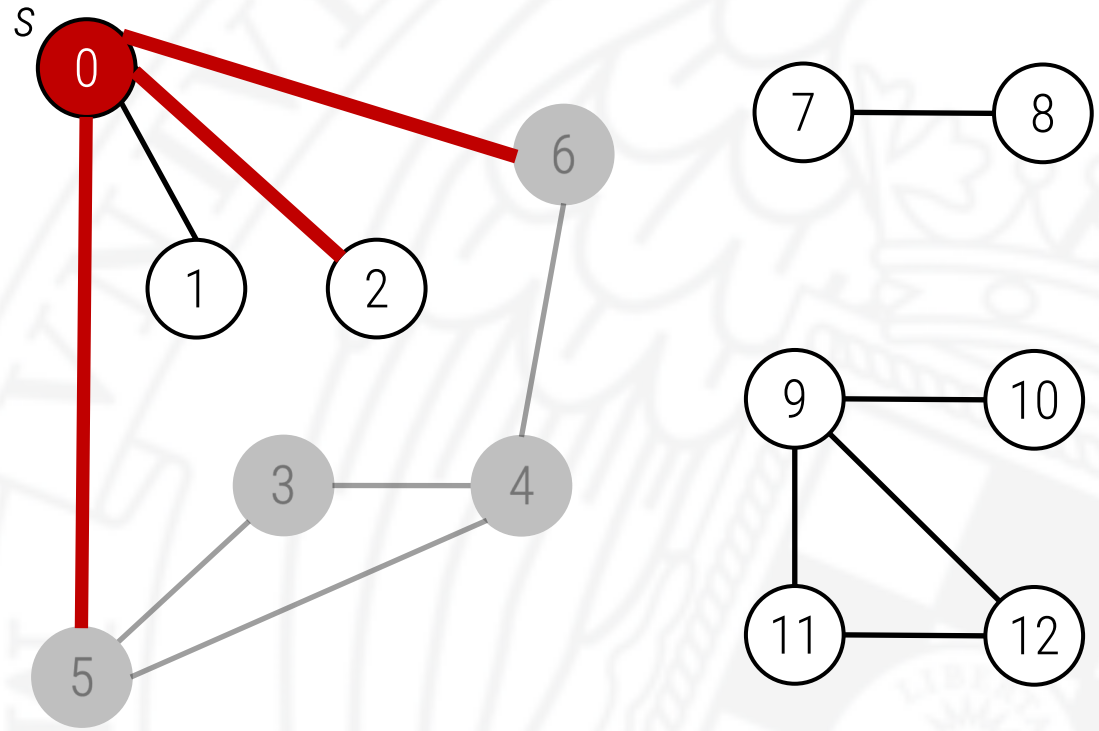
v	visitado	anterior
0	T	-
1	F	-
2	F	-
3	T	5
4	T	6
5	T	4
6	T	0
7	F	-
8	F	-
9	F	-
10	F	-
11	F	-
12	F	-

Recorrido en profundidad



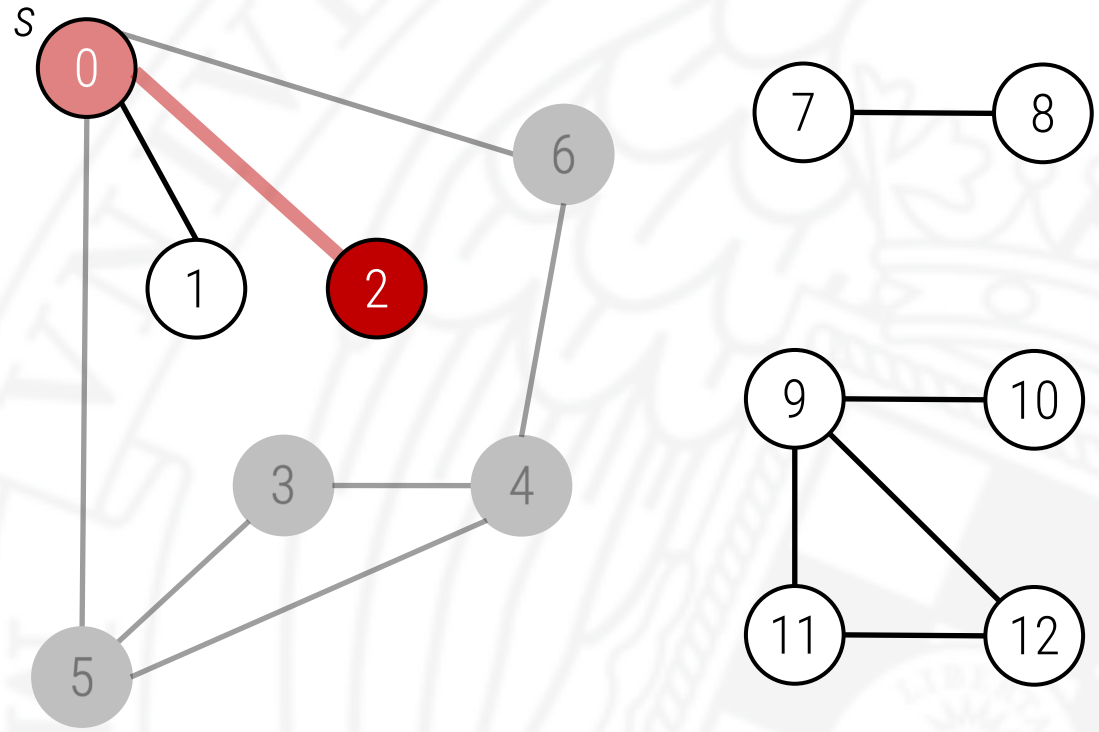
v	visitado	anterior
0	T	-
1	F	-
2	F	-
3	T	5
4	T	6
5	T	4
6	T	0
7	F	-
8	F	-
9	F	-
10	F	-
11	F	-
12	F	-

Recorrido en profundidad



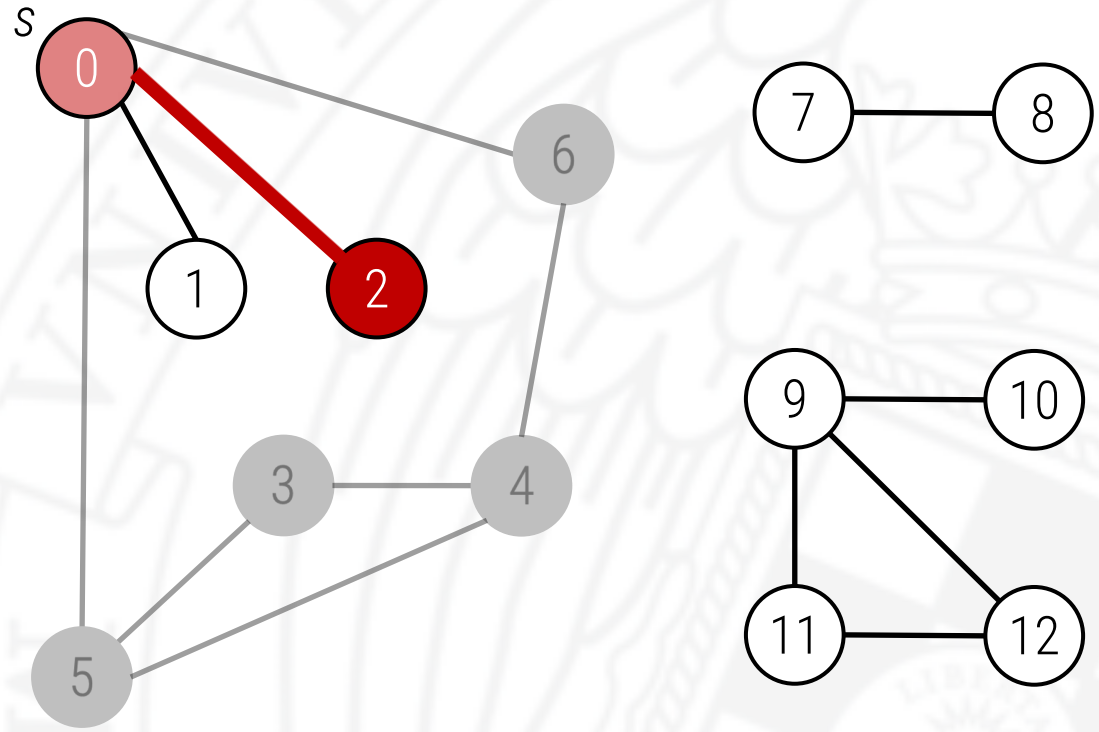
v	visitado	anterior
0	T	-
1	F	-
2	F	-
3	T	5
4	T	6
5	T	4
6	T	0
7	F	-
8	F	-
9	F	-
10	F	-
11	F	-
12	F	-

Recorrido en profundidad



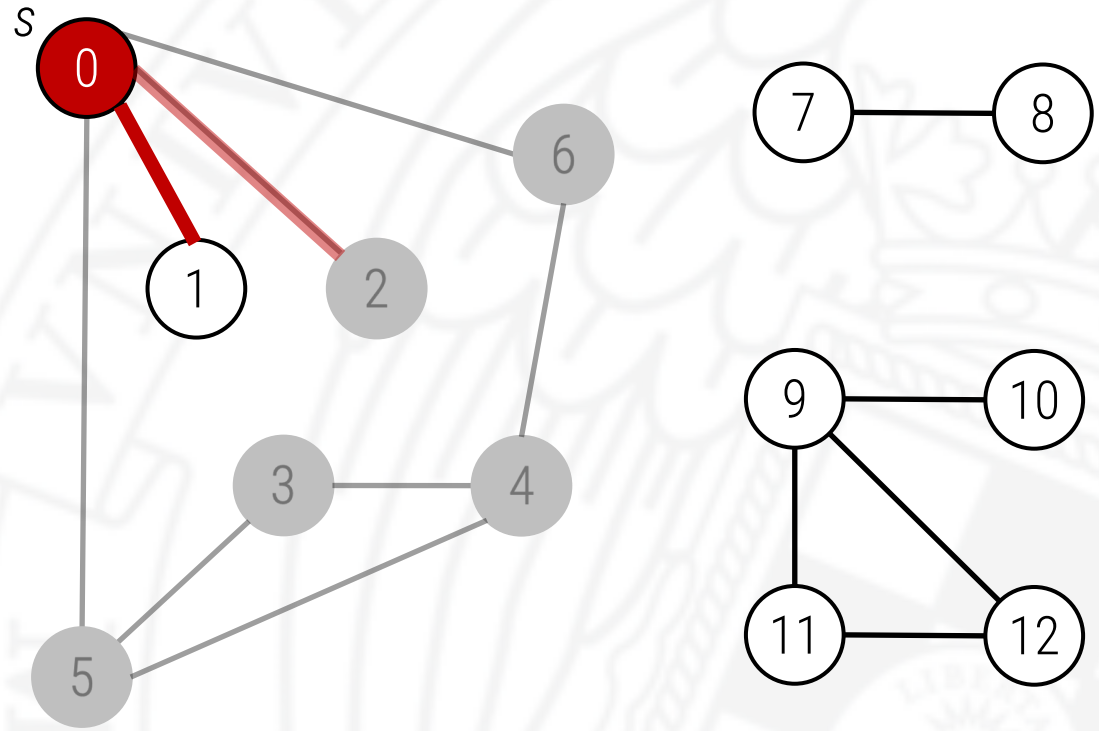
v	visitado	anterior
0	T	-
1	F	-
2	T	0
3	T	5
4	T	6
5	T	4
6	T	0
7	F	-
8	F	-
9	F	-
10	F	-
11	F	-
12	F	-

Recorrido en profundidad



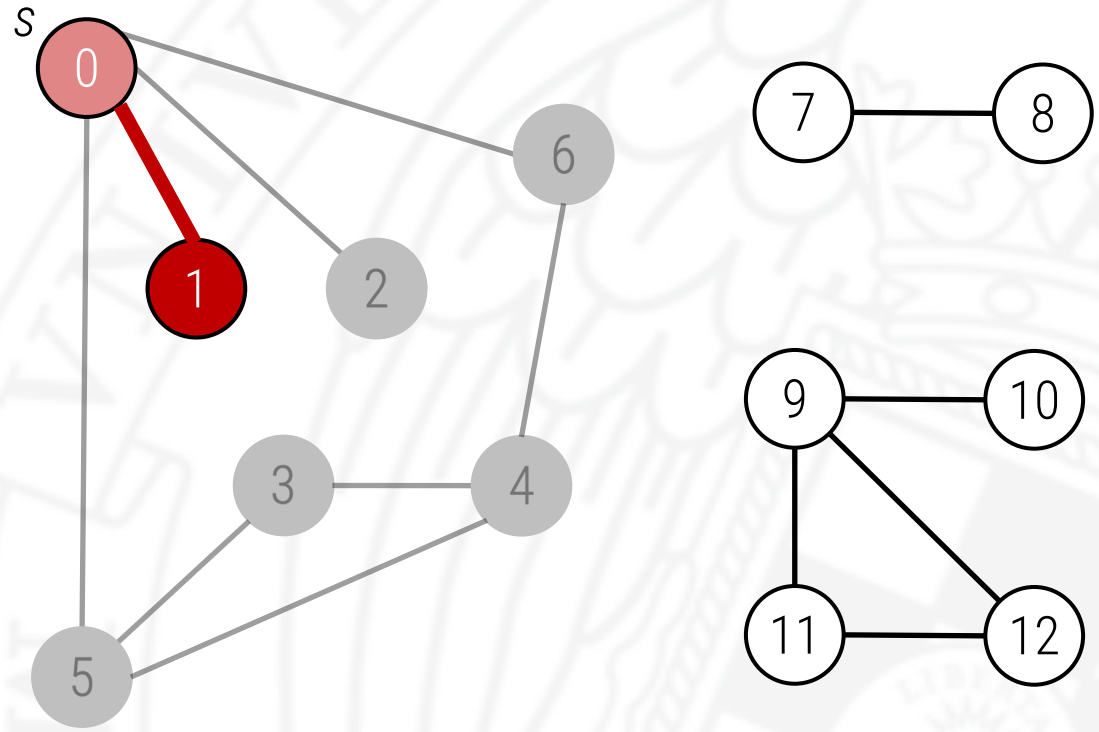
v	visitado	anterior
0	T	-
1	F	-
2	T	0
3	T	5
4	T	6
5	T	4
6	T	0
7	F	-
8	F	-
9	F	-
10	F	-
11	F	-
12	F	-

Recorrido en profundidad



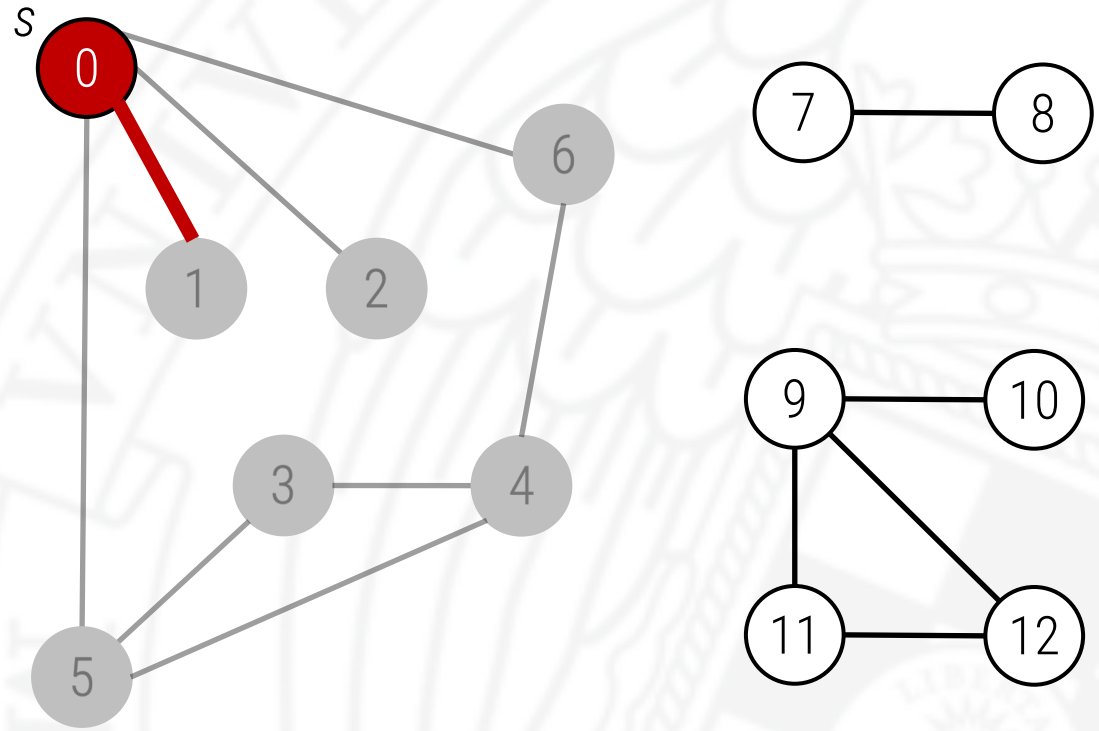
v	visitado	anterior
0	T	-
1	F	-
2	T	0
3	T	5
4	T	6
5	T	4
6	T	0
7	F	-
8	F	-
9	F	-
10	F	-
11	F	-
12	F	-

Recorrido en profundidad



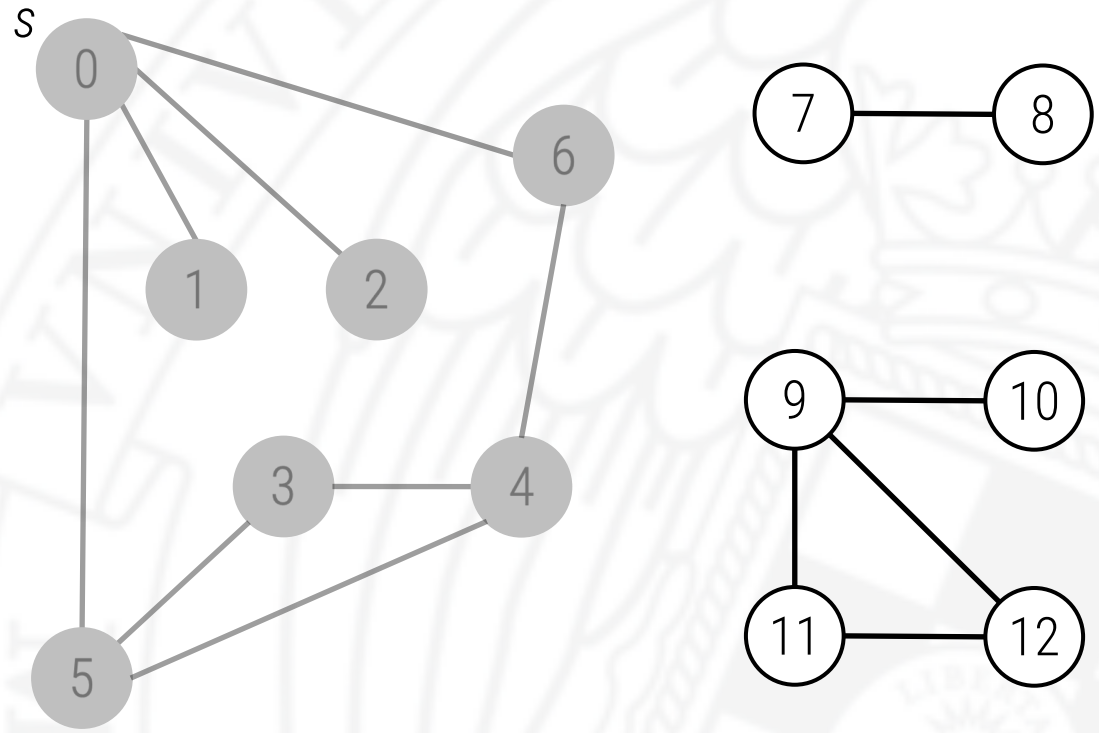
v	visitado	anterior
0	T	-
1	T	0
2	T	0
3	T	5
4	T	6
5	T	4
6	T	0
7	F	-
8	F	-
9	F	-
10	F	-
11	F	-
12	F	-

Recorrido en profundidad



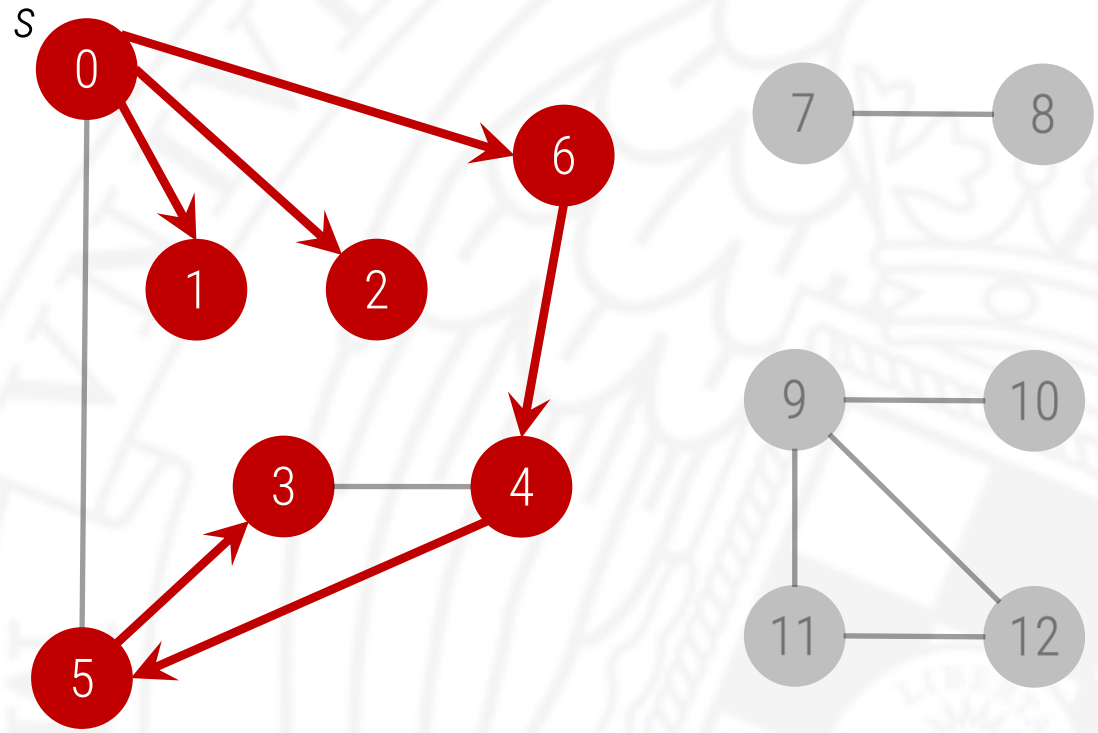
v	visitado	anterior
0	T	-
1	T	0
2	T	0
3	T	5
4	T	6
5	T	4
6	T	0
7	F	-
8	F	-
9	F	-
10	F	-
11	F	-
12	F	-

Recorrido en profundidad



v	visitado	anterior
0	T	-
1	T	0
2	T	0
3	T	5
4	T	6
5	T	4
6	T	0
7	F	-
8	F	-
9	F	-
10	F	-
11	F	-
12	F	-

Recorrido en profundidad



v	visitado	anterior
0	T	-
1	T	0
2	T	0
3	T	5
4	T	6
5	T	4
6	T	0
7	F	-
8	F	-
9	F	-
10	F	-
11	F	-
12	F	-

Caminos desde un origen

```
class CaminosDFS {
private:
    std::vector<bool> visit; // visit[v] = ¿hay camino de s a v?
    std::vector<int> ant;    // ant[v] = último vértice antes de llegar a v
    int s;                   // vértice origen

    void dfs(Grafo const& G, int v) {
        visit[v] = true;
        for (int w : G.ady(v)) {
            if (!visit[w]) {
                ant[w] = v;
                dfs(G, w);
            }
        }
    }
```

Caminos desde un origen

```
public:
    CaminosDFS(Grafo const& g, int s) : visit(g.V(), false),
                                        ant(g.V()), s(s) {
        dfs(g, s);
    }

    // ¿hay camino del origen a v?
    bool hayCamino(int v) const {
        return visit[v];
    }
```

Caminos desde un origen

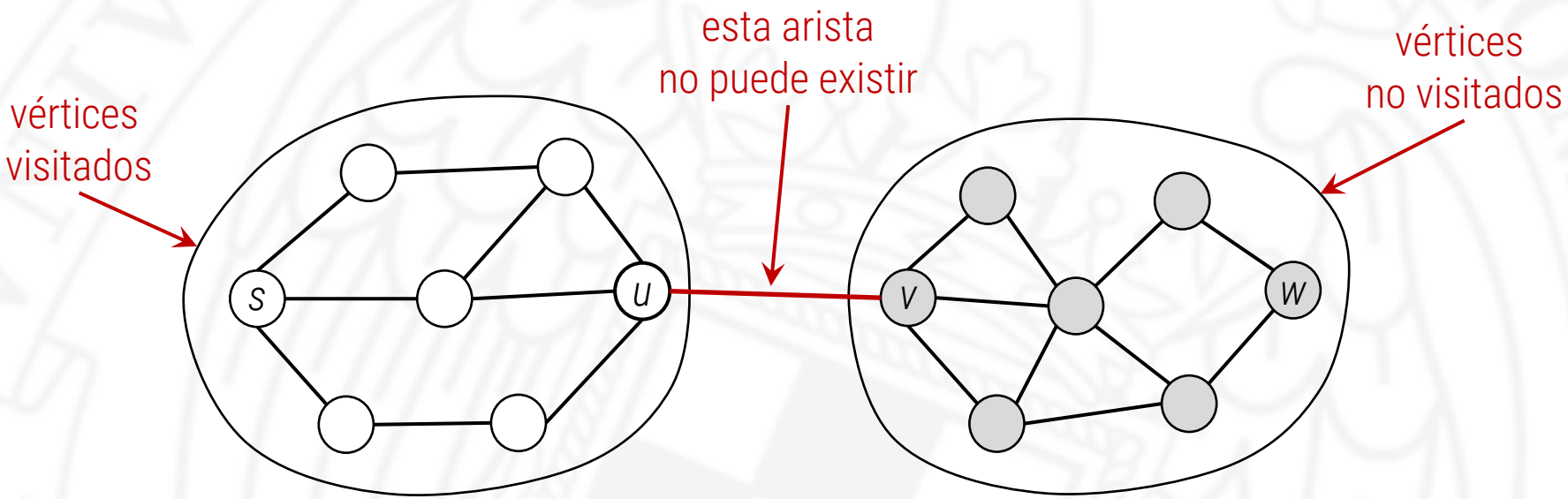
```
    using Camino = std::deque<int>;   // para representar caminos

    // devuelve un camino desde el origen a v (debe existir)
    Camino camino(int v) const {
        if (!hayCamino(v))
            throw std::domain_error("No existe camino");
        Camino cam;
        // recuperamos el camino retrocediendo
        for (int x = v; x != s; x = ant[x])
            cam.push_front(x);
        cam.push_front(s);
        return cam;
    }
};
```

Recorrido en profundidad: corrección y coste

- El recorrido en profundidad visita todos los vértices alcanzables desde el origen *s* en un tiempo proporcional a la suma de sus grados (más la inicialización de los vectores).
- Si *w* es visitado es porque está conectado a *s* (solo pasamos de un vértice a otro atravesando aristas).

Recorrido en profundidad: corrección y coste

- Si *w* está conectado a *s*, entonces termina siendo visitado.



- El coste está en $O(V + A)$ porque cada vértice se visita una única vez.