RECORRIDO EN PROFUNDIDAD

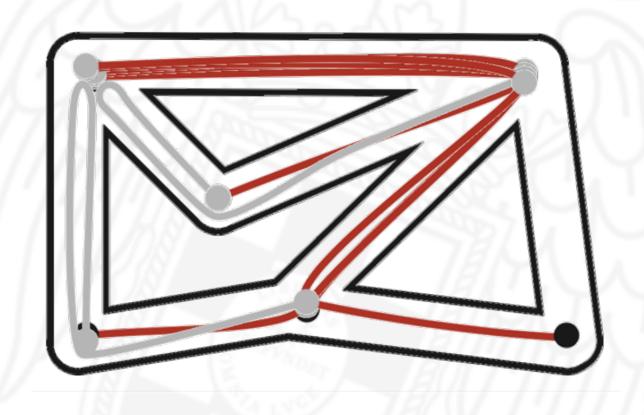


ALBERTO VERDEJO

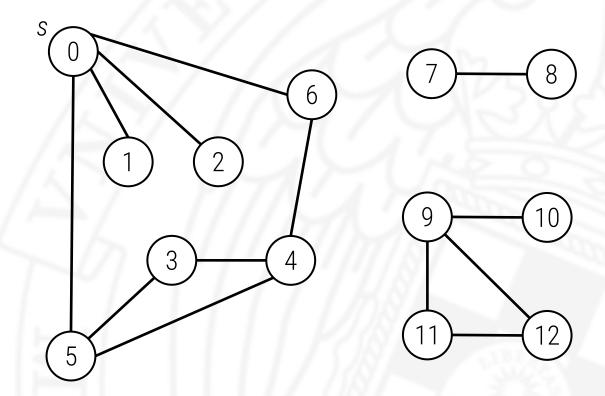
Recorridos

- 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.

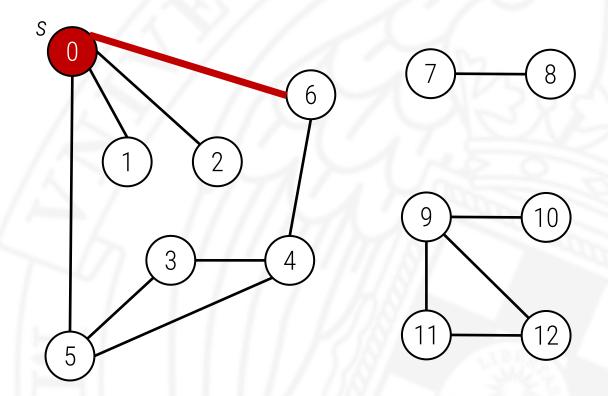
► El recorrido o búsqueda en profundidad (en inglés, *depth-first search*) imita la resolución de un laberinto.



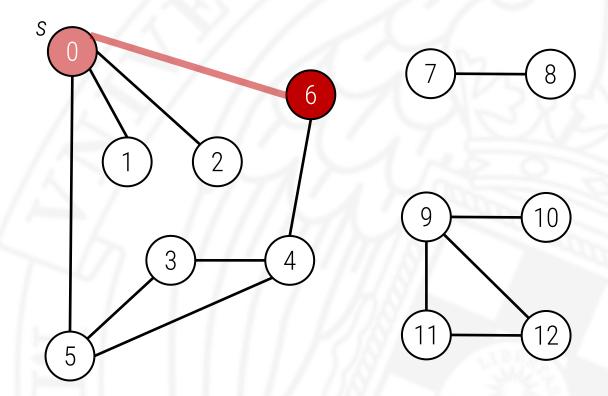
- Para recorrer un grafo utilizamos un algoritmo recursivo que va visitando vértices.
- ► Visitar un vértice *v* consiste en:
 - marcarlo como visitado;
 - ► hacer algo con él; y
 - visitar (recursivamente) todos los vértices adyacentes a v aún no visitados.



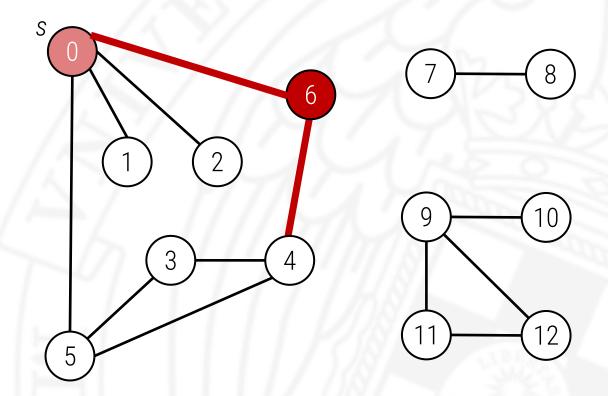
٧	visitado	anterior
0	F	-
1	F	\ -
2	F	7/-
3	F 2	_\
4	F	-
5	F F	TO.1
6	F	
7	/ F	
8	F F	
9) F	
10	//F	5
11	/ F /	
12	F/?	70 1



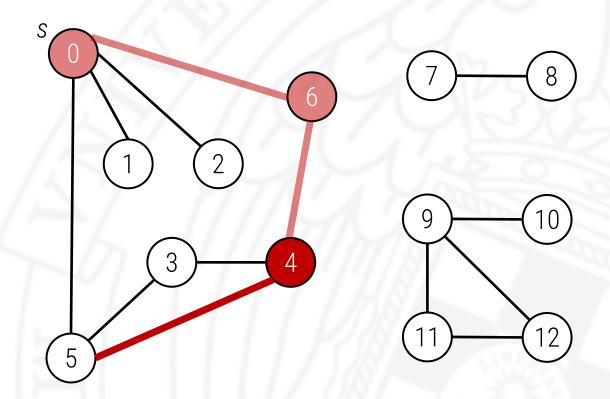
٧	visitado	anterior
0	1	-
1)\F	\ -
2	F	7/-
3	F 2	_\
4	\F\\	-
5	F \	-01
6	F F	
7	/\ F	
8	F	-
9	F F	
10	//F	
11	[/ F].	
12	F/2	70 1 1



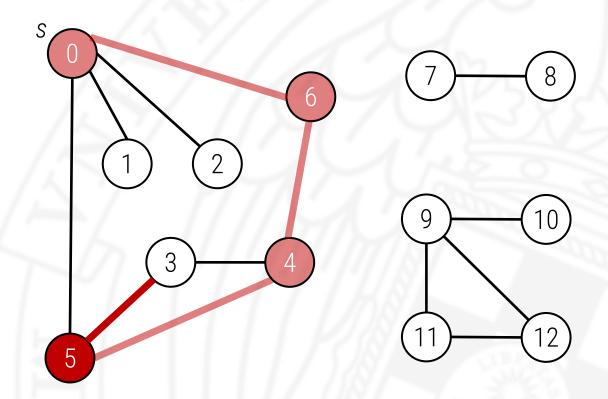
V	visitado	anterior
0	T	-
1	F	\ -
2	F	٦/-
3	F 2	_\
4	\F\\	-
5	F T	T.
6	T	0
7	F	\leq
8	F F	
9	F	اتسا
10	/ F	[-1]
11	/ F /	
12	F/?	70 1



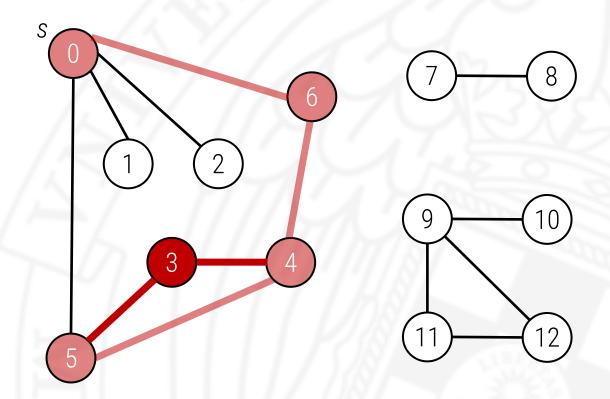
V	visitado	anterior
0	T	-
1	F	\ -
2	F	7/-
3	F 2	<u> </u>
4	F	-
5	F T	.01
6	T	0
7	/\ F	\leq
8	F F	
9	F	
10	/ F	5
11	/ F /	
12	F/?	70 1



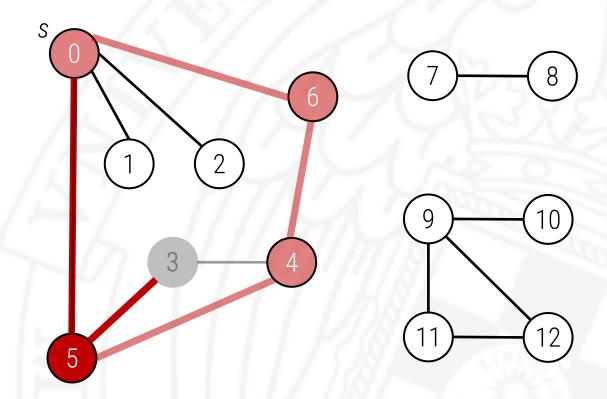
٧	visitado	anterior
0	1	-
1)\F	\ -
2	F	7/-
3	F 2	_\
4	\T \\	6
5	F T	-0,1
6	T	0
7	/ F	<
8	F	-
9	17 F	
10	//F	[-1]
11	/ F /.	
12	′ F/∂	70 1



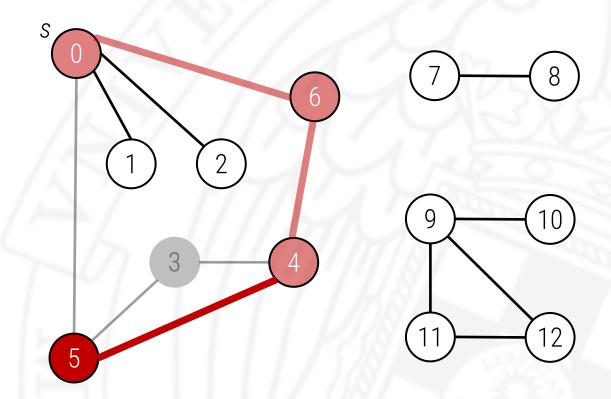
٧	visitado	anterior
0		-
1	F	\ -
2	F	7/-
3	F 2	_\
4	\T \\	6
5	T \	4
6	T T	0
7	F	\leq
8	F F	
9	F	
10	//F	[
11	/ F /	
12	′ F/∂	70 1 /



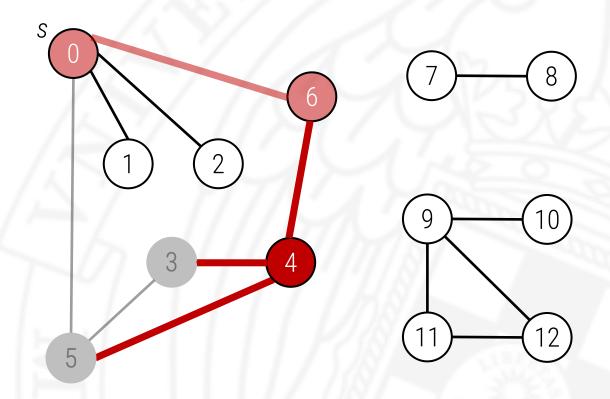
V	visitado	anterior
0	T	-
1	F	\ -
2	F	7/-
3	T/2	5
4	\T \\	6
5	Τ.	4
6	T T	0
7	F	<
8	F F	
9	F	
10	/ F	[-1]
11	/ F /	
12	F/?	70 1



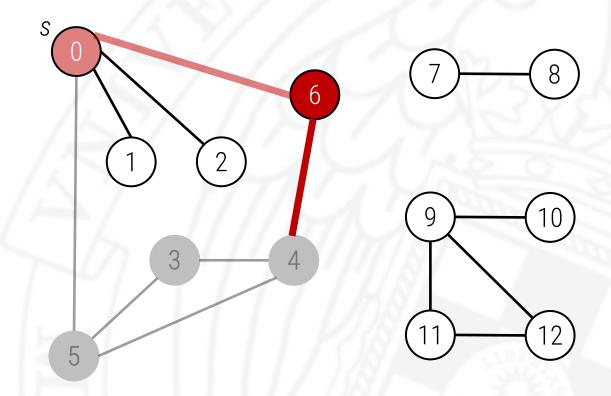
٧	visitado	anterior
0	1	-
1)\F	\ -
2	F	7/-
3	T/2	5
4	\T \\	6
5	T \	4
6	T T	0
7	F	<
8	F	
9	7 F	
10	//F	[-1]
11	/ F /.	2
12	′ F/∂	70 1



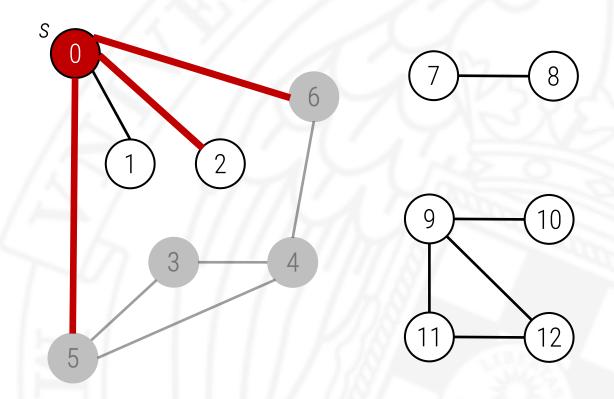
٧	visitado	anterior
0	1	-
1	F	\ -
2	F	7/-
3	T/2	5
4	\T \\	6
5	Τ.	4
6	T T	0
7	F	<
8	F	-
9	17 F	
10	//F	
11	/ F /.	
12	′ F/∂	70 1



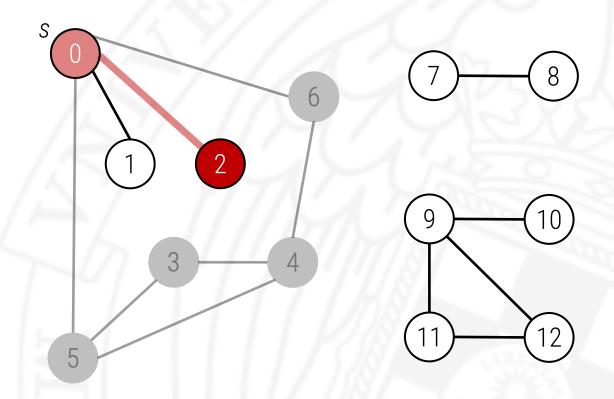
V	visitado	anterior
0	T	-
1	F	\ -
2	F	7/-
3	\ T\ 2	5
4	\T \\	6
5	Τ.	4
6	T T	0
7	F	<
8	F	
9	F F	
10		[
11	/ F /	
12	F/(70 1



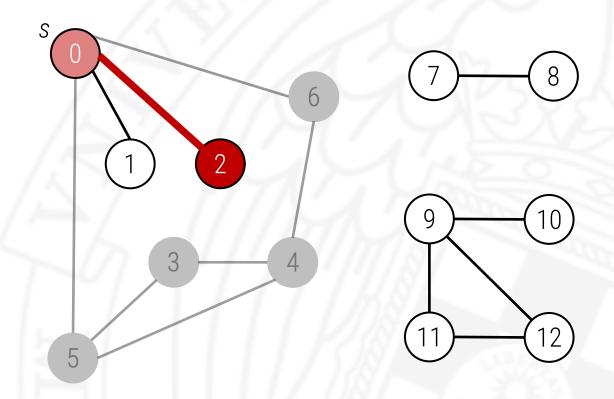
V	visitado	anterior
0		-
1	F	\ -
2	F	7/-
3	\ T\ 2	5
4	\T \\	6
5	Τ.	4
6	T T	0
7	F	<
8	F	-
9	F F	
10		1
11	/ F /.	
12	F/{	70 1



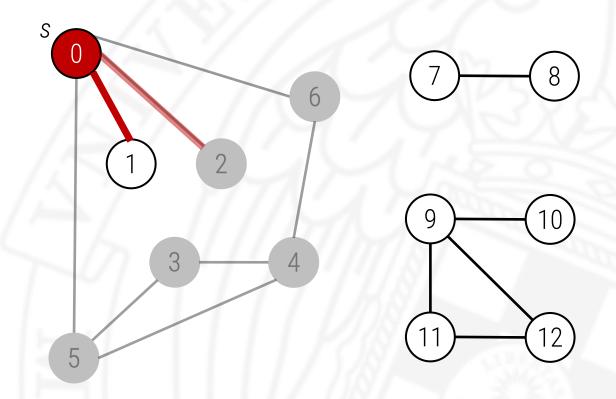
٧	visitado	anterior
0	T	-
1	F	\ -
2	F	7/-
3	/ T/ 2	5
4	\T \\	6
5	Τ.	4
6	T T	0
7	F	<
8	F	-
9	7 F	
10	//F /	1
11	/ F /.	
12	′ F/∂	70 1



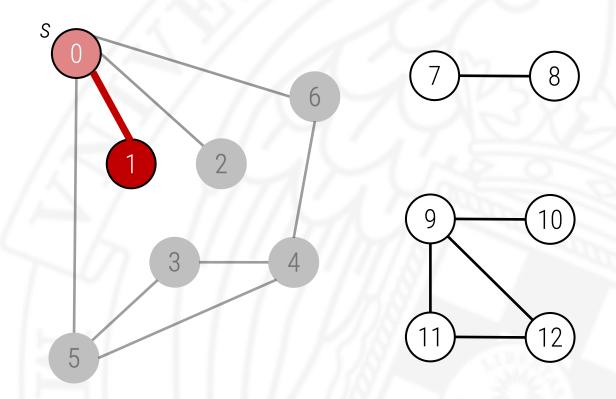
٧	visitado	anterior
0	T	-
1	F	\ -
2	1	0
3	T 2	0 5
4	\T \\	6
5	T \	4
6	T	0
7	F	<
8	F	-
9) F	
10	//F	1
11	/ F /.	
12	′ F/∂	70 1



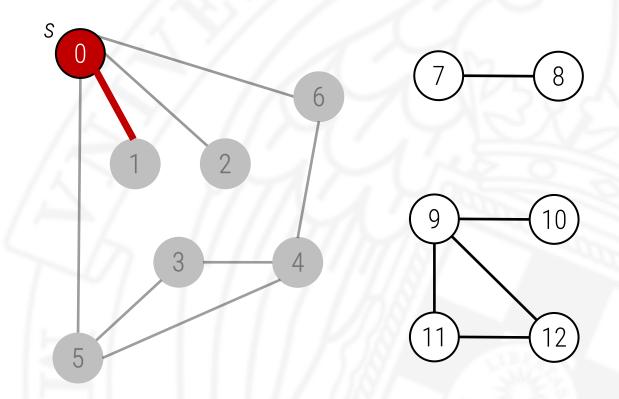
٧	visitado	anterior
0	T	-
1	F	\ -
2	1	0
3	T 2	0 5
4	\T \\	6
5	T \	4
6	T	0
7	F	<
8	F	-
9) F	
10	//F	1
11	/ F /.	
12	′ F/∂	70 1



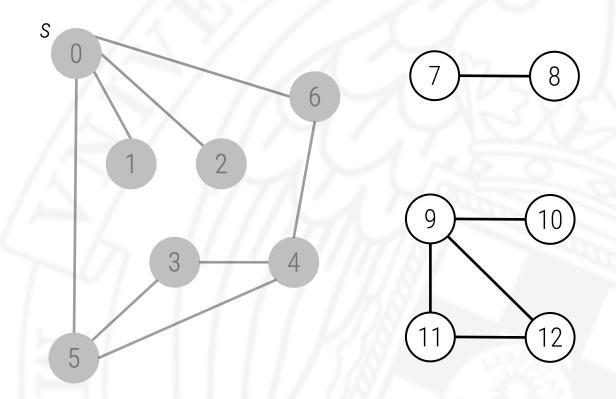
٧	visitado	anterior
0	T	-
1	F	\ -
2	T	0
3	T 2	0 5
4	\T \\	6
5	T \	4
6	T	0
7	F	<
8	F	-
9) F	
10	//F	1
11	/ F /.	
12	′ F/∂	70 1



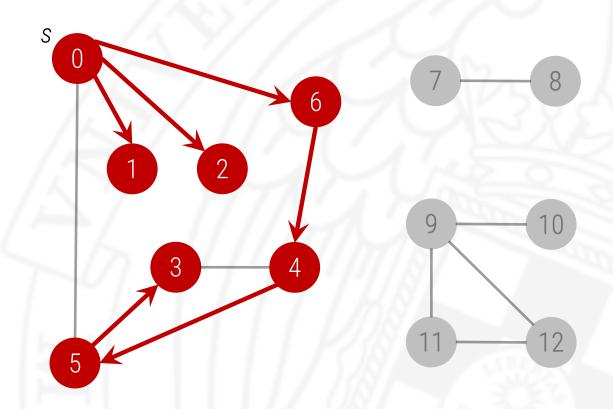
٧	visitado	anterior
0	T	-
1)\T	0
2	1	0
3	T/2	0 5
4	\T \\	6
5	Τ \	4
6	T	0
7	F	<
8	F	-
9	7 F	
10	//F	5
11	/ F /.	
12	′ F/∂	70 1



٧	visitado	anterior
0	T	-
1		0
2	T	0
3	T	0 5
4	\T \\	6
4 5	Τ \	4
6	T \	0
7	/\ F	<
8	F	
9	7) F	
10	//F	
11	/ F /	
12	/ F/(70°/



٧	visitado	anterior
0		-
1	T	0
2	T	0
3	T 2	0 5
4	\T \\	6
4 5	T	4
6	T	0
7	/\ F	
8	F	
9	F F	اتسا
10	/ F	1-1
11	/ F /.	
12	F/?	10



٧	visitado	anterior
0	1	-
1	1	0
2	T	0
3	T 2	0 5
4	\T \\	6
5	Τ.	4
6	T T	0
7	/\ F	
8	F	
9	F F	ائت
10	//F /	1
11	[/ F].	
12	/ F/(70°/

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

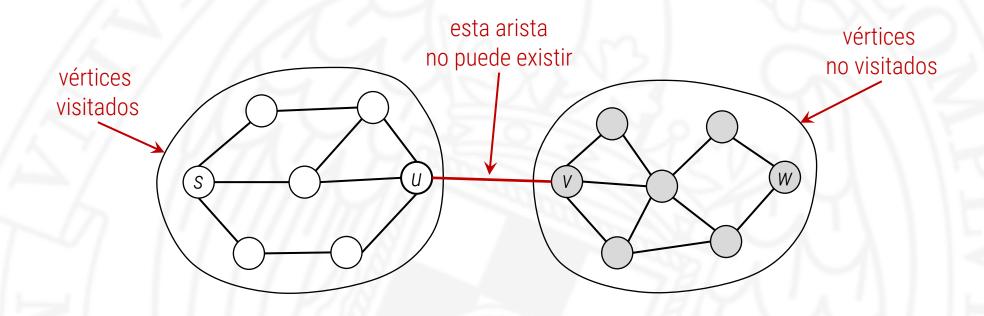
```
using Camino = std::deque<int>; // para representar caminos
// devuelve un camino desde el origen a v (debe existir)
Camino camino(int v) const {
  if (!hayCamino(v))
      throw std::domain_error("No existe camino");
   Camino cam;
   // recuperamos el camino retrocediendo
   for (int x = v; x != 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.