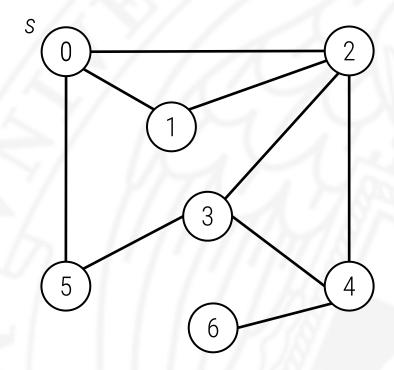
# RECORRIDO EN ANCHURA

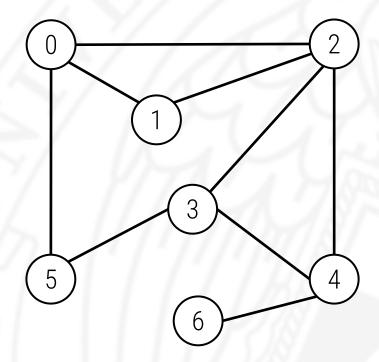


**ALBERTO VERDEJO** 

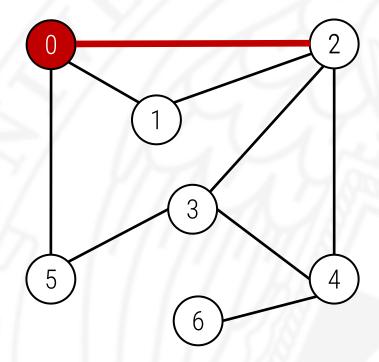
- ► En ocasiones estamos interesados en encontrar el camino más corto desde un origen s a otro vértice v (o a todos los vértices conectados a s).
- ► El recorrido en anchura (en inglés, breadth-first search) logra eso: primero visita todos los vértices alcanzables siguiendo una arista (a distancia 1); luego visita todos los vértices alcanzables utilizando dos aristas (a distancia 2); y así sucesivamente.
- Para lograrlo utiliza una cola donde guardar los vértices alcanzados pero que aún no se han explorado sus adyacentes.



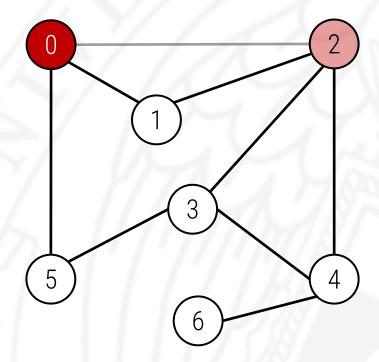
V	visitado	anterior	distancia
0	F	D/7/(	$\supset$ $f$
1	F	// - /	1
2	F	/) <del>/</del> /	
3	WEL	'\-\	1101
4	F	71 <del>-</del> 1	
5	NE	' ) l	
6	JF /	74 I	-3



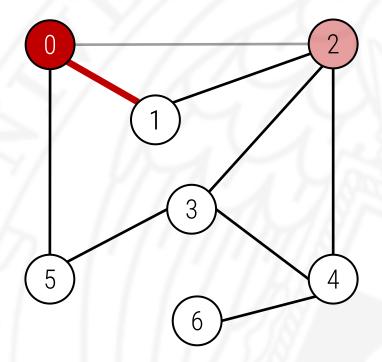
٧	visitado	anterior	distancia
0	T	7/-/(	0
1	F	// - //	1
2	F	1) - (	1
3	WE!	1/-/	1501
4	F	41 <del>-</del> 1	
5	)E/	' ) l	-
6	(F   '	77-I	-3



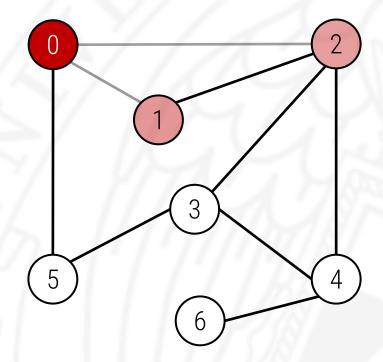
٧	visitado	anterior	distancia
0	T	7/-/(	0
1	F	// - //	1
2	F	1) - (	1
3	WE!	1/-/	1501
4	F	41 <del>-</del> 1	
5	)E/	' ) l	-
6	(F   '	77-I	-3



٧	visitado	anterior	distancia
0	T	D/ <u>-</u> //	0
1	E	// - //	1
2	Т	0	1
3	WE!	'\-\	1101
4	F	71 <del>-</del> 1	
5	DE/	' ) l	
6	OF I	77A I	

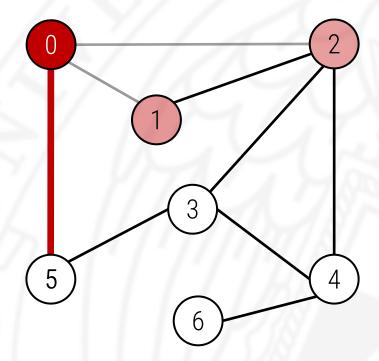


V	visitado	anterior	distancia
0	TV	7/7/	0
1	F	// - //	1
2	Т	0	1
3	VE/	1/-/	1101
4	F	41 <del>-</del> 1	-
5	)E/	' ) l	
6	F	77 I	1.7



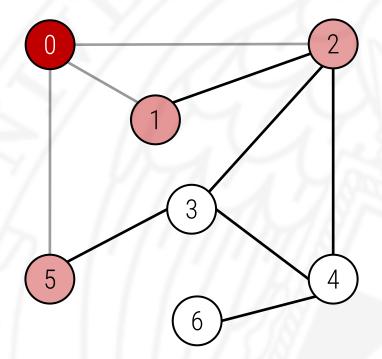
٧	visitado	anterior	distancia
0	T	7/7/	0
1	T	0	1
2	T	0	1
3	WEL	1/-/	11-01
4	F	71 <del>-</del> 1	
5	)E/	' ) l	
6	OF I	74 1	

Cola: 2 1



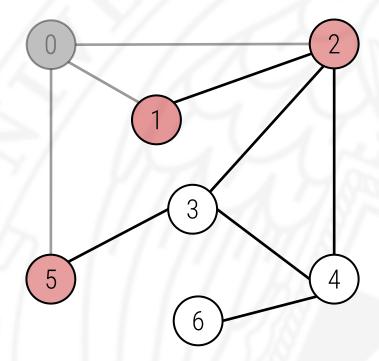
V	visitado	anterior	distancia
0	T	7/7/	0
1	T	0	1
2	Т	0	1
3	WE!	,/-/	1101
4	F	71 <del>-</del> 1	
5	()E/	' ) l	
6	F	74 I	1.73

Cola: 2 1



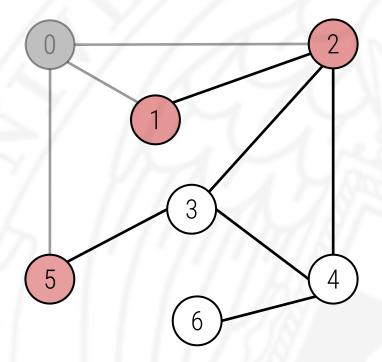
visitado	anterior	distancia
TV	7//(	0
	0	1
T	0	1
7/E/	1/-/	1101
F	71 <del>-</del> 1	
M	0	1
_(F   )	77A I	-3
	T T F T	T - T 0 T 0 F - F - T 0

Cola: 2 1 5



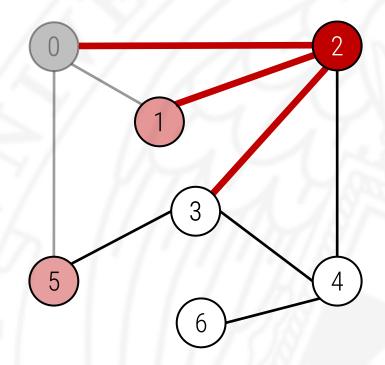
٧	visitado	anterior	distancia
0	T	7/7/	0
1		0	1
2	T	0	1
3	W	1/-/	1101
4	F	71 <del>-</del> 1	
5		0	1
6	(F /	77A I	1.7

Cola: 2 1 5



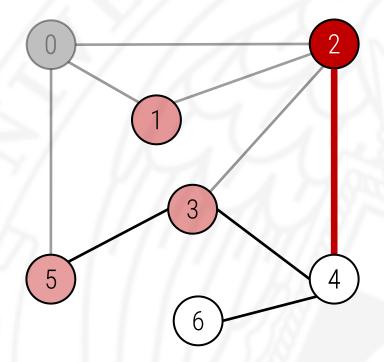
٧	visitado	anterior	distancia
0	TV	D/ <u>-</u> //	0
1		0	1
2	T	0	1
3	WEL	1/-/	1101
4	F	71 <del>-</del> 1	
5	DI/	0	1
6	OF I	77A I	

Cola: 2 1 5



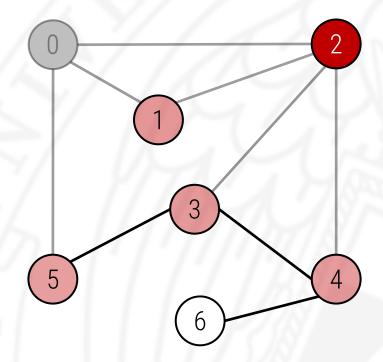
٧	visitado	anterior	distancia
0	T	7/7/	0
1		0	1
2	T	0	1
3	WEL	1/-/	101
4	F	71- I	-
5	M	0	1
6	(F)	77 I	-

Cola: 1 5



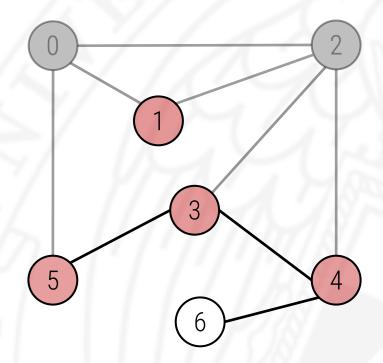
V	visitado	anterior	distancia
0	T	7//	0
1	T	0	1
2	T	0	1
3	V.F.(	2	2
4	F	71 <del>-</del> 1	
5	) T/	0	1
6	F	74 I	

Cola: 1 5 3



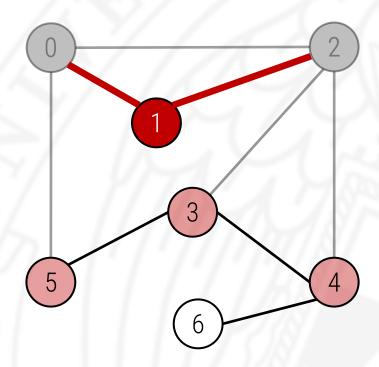
V	visitado	anterior	distancia
0	T	7/7/	0
1		0	1
2	T	0	1
3	7/5/	2	2
4	T	2	2
5	M	0	1
6	(F /	77 I	

Cola: 1 5 3 4



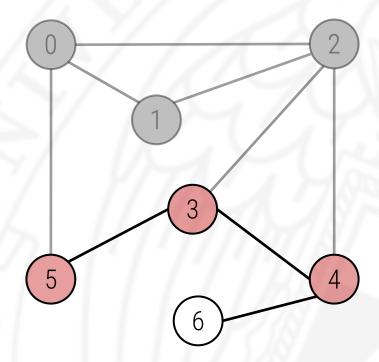
V	visitado	anterior	distancia
0	TV	7/7/	0
1		0	1
2	T	0	1
3	W.	2	2
4	1	2	2
5	171	0	1
6	CF I	77 I	- 1

Cola: 1 5 3 4



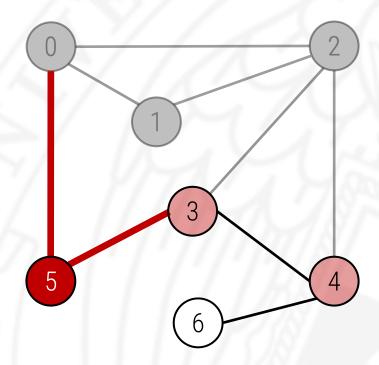
V	visitado	anterior	distancia
0	TV	7/7/	0
1	Ţ	0	1
2	T	0	1
3	V.F.	2	2
4	T	2	2
5	N	0	1
6	JF /	. TA 1	17

Cola: 5 3 4



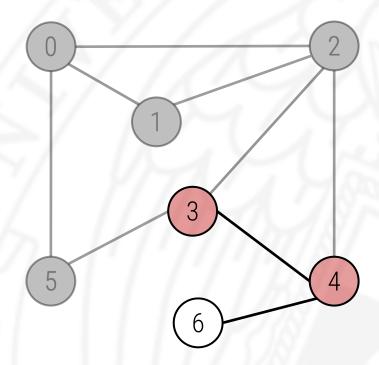
٧	visitado	anterior	distancia
0	TV	7/7/	0
1		0	1
2	T	0	1
3	7/5/	2	2
4	T	2	2
5	DI/	0	1
6	F /	77A I	

Cola: 5 3 4



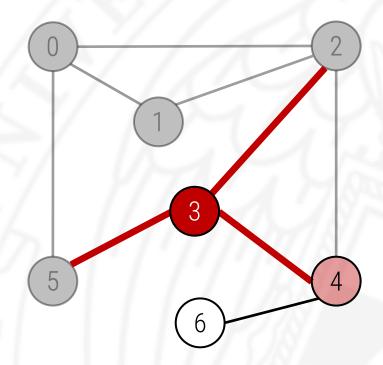
V	visitado	anterior	distancia
0	TV	7/7/	0
1	Ţ	0	1
2	T	0	1
3	VE/	2	2
4	T	2	2
5	M	0	1
6	F /	77 I	1.7

Cola: 3 4

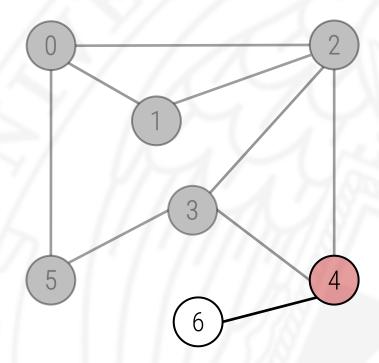


V	visitado	anterior	distancia
0	TV	7/7/	0
1	Ţ	0	1
2	T	0	1
3	VE/	2	2
4	T	2	2
5	M	0	1
6	F /	77 I	1.7

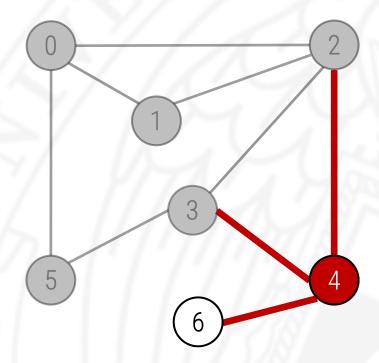
Cola: 3 4



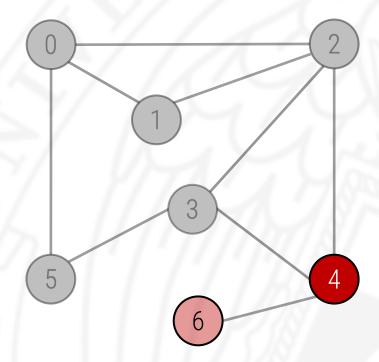
V	visitado	anterior	distancia
0	TV	7/7/	0
1	T.	0	1
2	T	0	1
3	V.F.	2	2
4	T	2	2
5	M	0	1
6	F	77 I	1-3



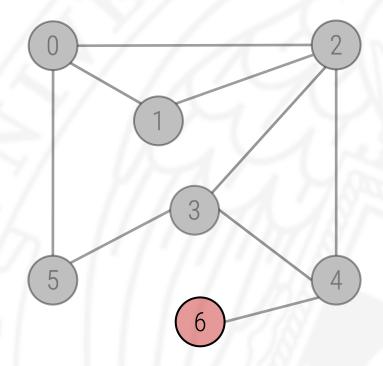
٧	visitado	anterior	distancia
0	T	7/7/	0
1	T.	0	1
2	T	0	1
3	V.F.	2	2
4	T	2	2
5	) I (	0	1
6	(F /	7A I	



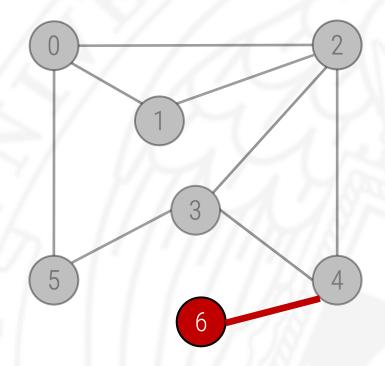
V	visitado	anterior	distancia
0	TV	7/7/	0
1	T.	0	1
2	T	0	1
3	7/5/	2	2
4	T	2	2
5	M	0	1
6	F /	7A I	



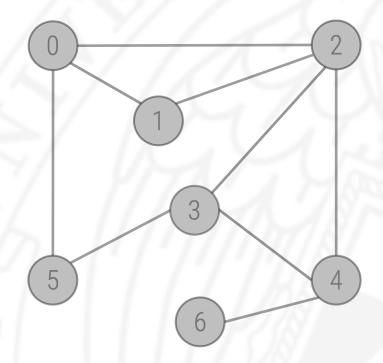
V	visitado	anterior	distancia
0	TV	7//(	0
1	T	0	1
2	T	0	1
3	V.F.	2	2
4	T	2	2
5	M	0	1
6	<b></b>	4	3



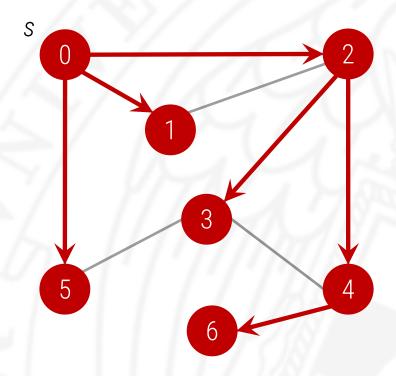
V	visitado	anterior	distancia
0	TV	7//	0
1	T	0	1
2	T	0	1
3	VF/	2	2
4	T	2	2
5	M	0	1
6	T	4	3



V	visitado	anterior	distancia
0	TV	7/7/	0
1	T.	0	1
2	T	0	1
3	7/5/	2	2
4	T	2	2
5	M	0	1
6	Ot I	4	3



٧	visitado	anterior	distancia
0	TV	7/7/	0
1	$\mathbf{t}$	0	1
2	T	0	1
3	V.F.	2	2
4	T	2	2
5	M	0	1
6	(t /	4	3



٧	visitado	anterior	distancia
0	TV	7/7/	0
1	T.	0	1
2	T	0	1
3	7/5/	2	2
4	T	2	2
5	M	0	1
6		4	3

```
class CaminoMasCorto
public:
  CaminoMasCorto(Grafo const& g, int s) : visit(g.V(), false),
                                           ant(g.V()), dist(g.V()), s(s) {
     bfs(g);
   // ¿hay camino del origen a v?
   bool hayCamino(int v) const {
     return visit[v];
```

```
// número de aristas entre s y v
int distancia(int v) const {
   return dist[v];
// devuelve el camino más corto desde el origen a v (si existe)
Camino camino(int v) const {
    if (!hayCamino(v)) throw std::domain_error("No existe camino");
    Camino cam;
    for (int x = v; x != s; x = ant[x])
        cam.push_front(x);
    cam.push_front(s);
    return cam;
```

```
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
    std::vector<int> dist; // dist[v] = aristas en el camino s-v más corto
    int s;
```

```
void bfs(Grafo const& g) {
   std::queue<int> q;
   dist[s] = 0; visit[s] = true;
   q.push(s);
   while (!q.empty()) {
      int v = q.front(); q.pop();
      for (int w : g.ady(v)) {
         if (!visit[w]) {
            ant[w] = v; dist[w] = dist[v] + 1; visit[w] = true;
            q.push(w);
```

#### Recorrido en anchura: corrección y coste

El recorrido en anchura encuentra caminos más cortos a todos los vértices conectados con el origen s, en un tiempo en O(V + A) (en el caso peor).

