RECORRIDO EN ANCHURA

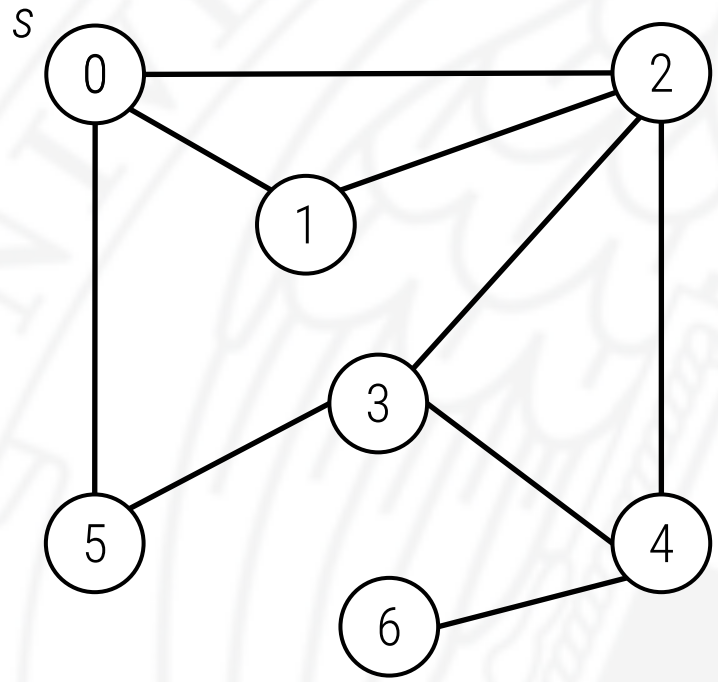
UNIVERSIDAD COMPLUTENSE MADRID

ALBERTO VERDEJO

Recorrido en anchura

- 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.

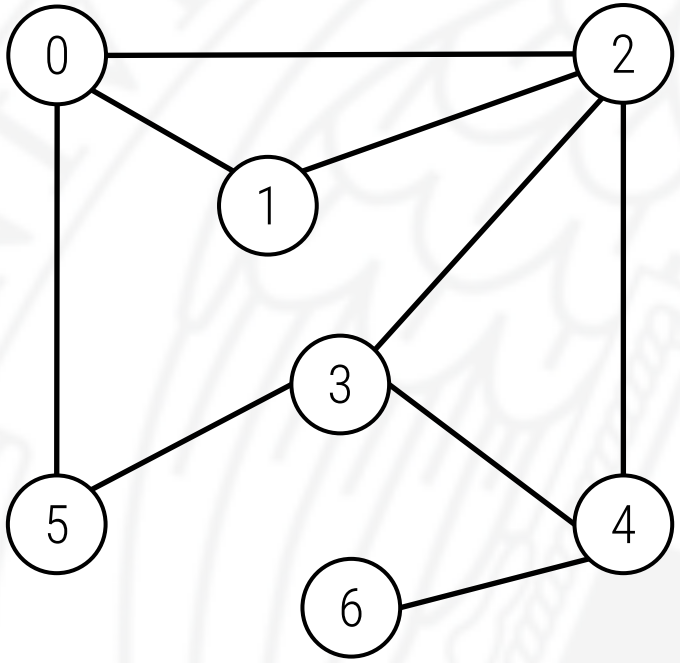
Recorrido en anchura



v	visitado	anterior	distancia
0	F	–	–
1	F	–	–
2	F	–	–
3	F	–	–
4	F	–	–
5	F	–	–
6	F	–	–

Cola:

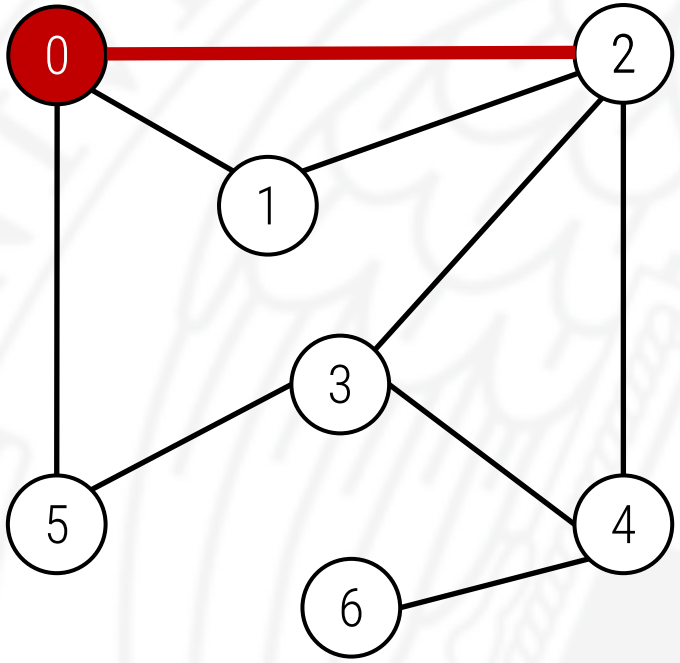
Recorrido en anchura



v	visitado	anterior	distancia
0	T	–	0
1	F	–	–
2	F	–	–
3	F	–	–
4	F	–	–
5	F	–	–
6	F	–	–

Cola: 0

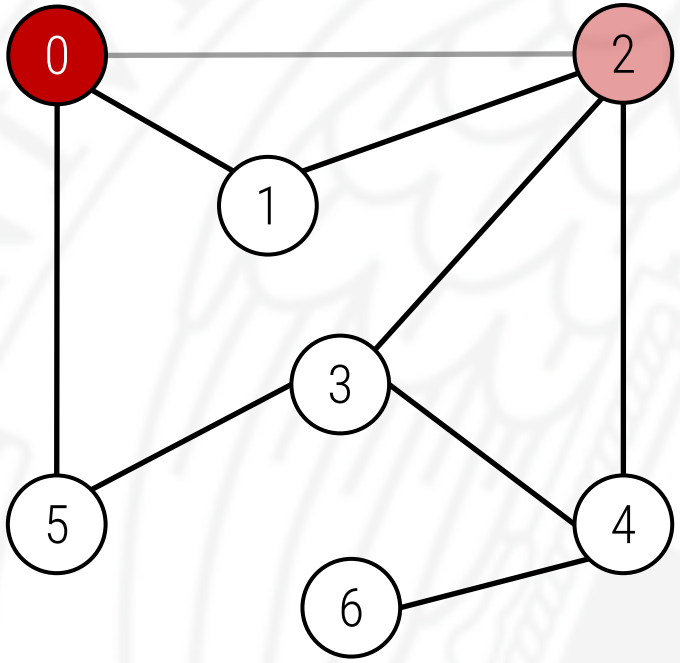
Recorrido en anchura



v	visitado	anterior	distancia
0	T	–	0
1	F	–	–
2	F	–	–
3	F	–	–
4	F	–	–
5	F	–	–
6	F	–	–

Cola: 0

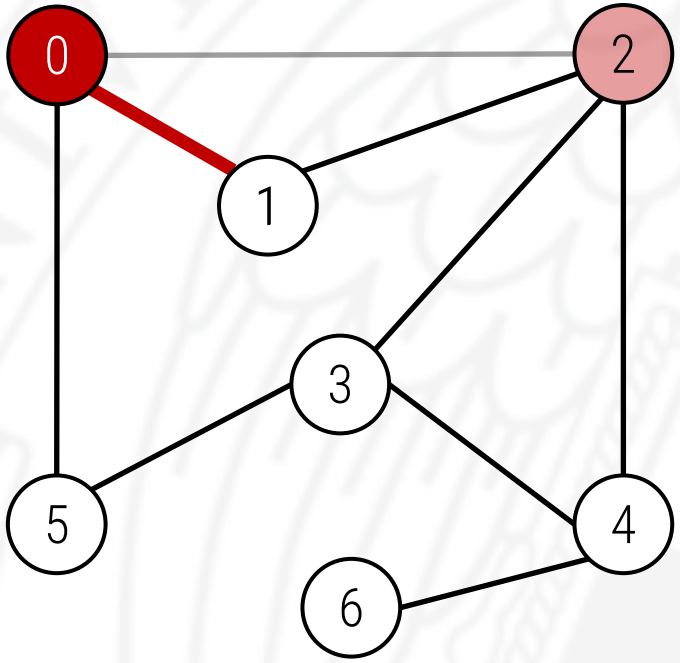
Recorrido en anchura



v	visitado	anterior	distancia
0	T	–	0
1	F	–	–
2	T	0	1
3	F	–	–
4	F	–	–
5	F	–	–
6	F	–	–

Cola: 2

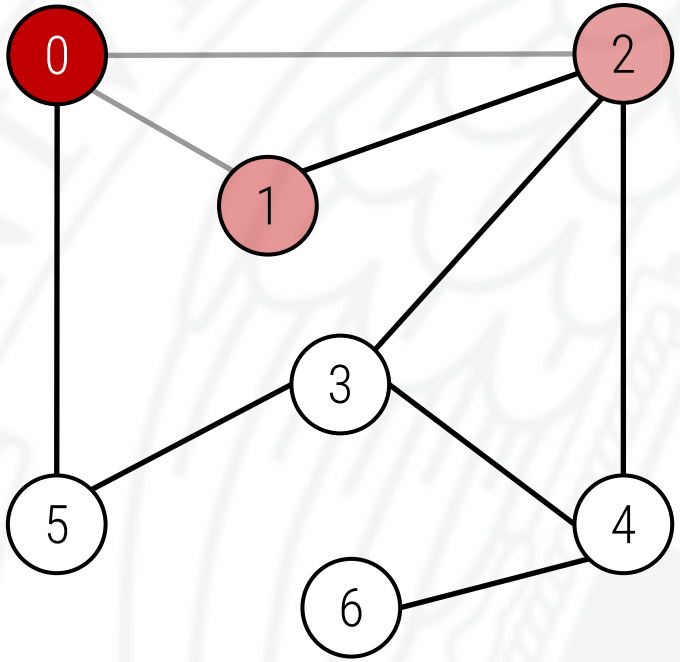
Recorrido en anchura



v	visitado	anterior	distancia
0	T	–	0
1	F	–	–
2	T	0	1
3	F	–	–
4	F	–	–
5	F	–	–
6	F	–	–

Cola: 2

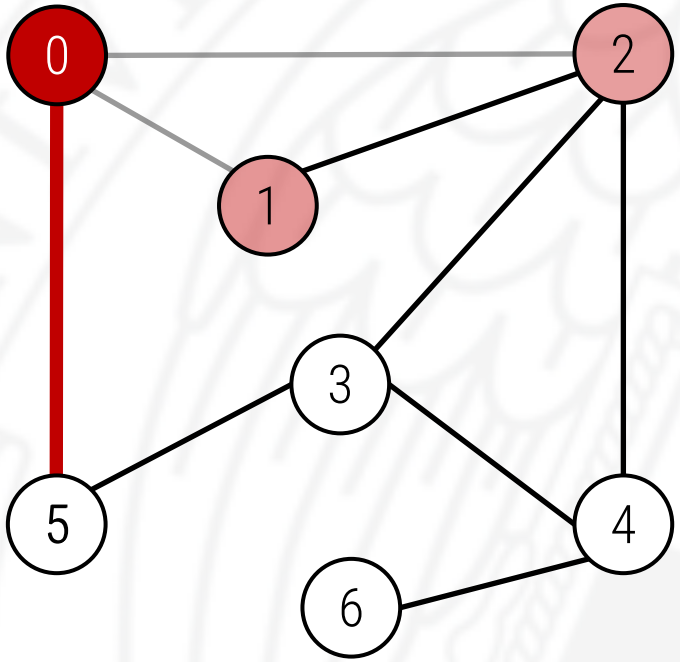
Recorrido en anchura



v	visitado	anterior	distancia
0	T	–	0
1	T	0	1
2	T	0	1
3	F	–	–
4	F	–	–
5	F	–	–
6	F	–	–

Cola: 2 1

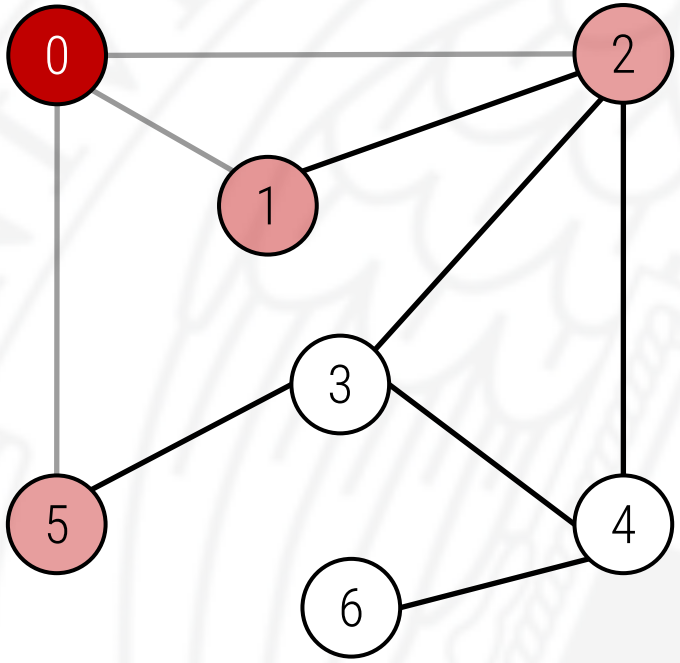
Recorrido en anchura



v	visitado	anterior	distancia
0	T	–	0
1	T	0	1
2	T	0	1
3	F	–	–
4	F	–	–
5	F	–	–
6	F	–	–

Cola: 2 1

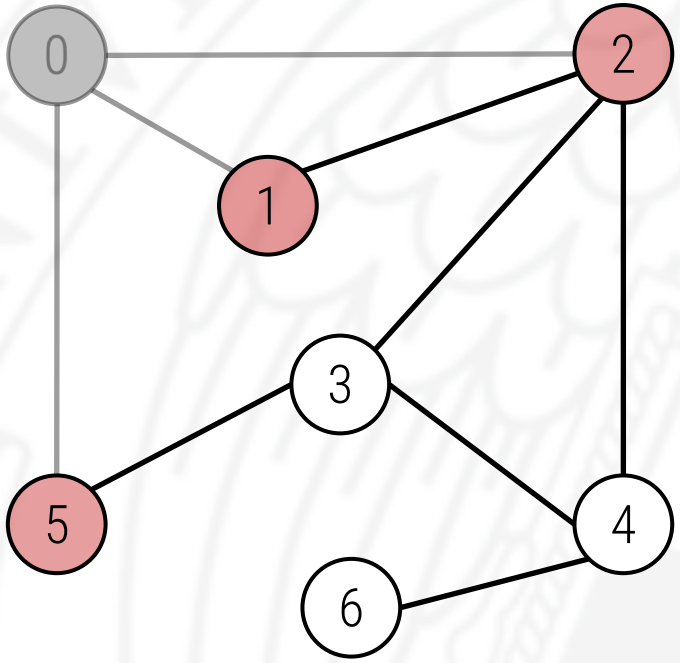
Recorrido en anchura



v	visitado	anterior	distancia
0	T	–	0
1	T	0	1
2	T	0	1
3	F	–	–
4	F	–	–
5	T	0	1
6	F	–	–

Cola: 2 1 5

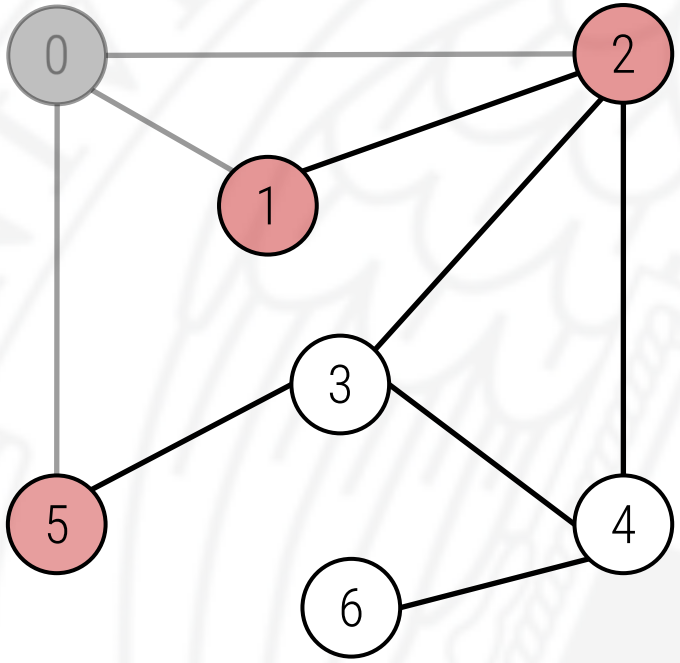
Recorrido en anchura



v	visitado	anterior	distancia
0	T	–	0
1	T	0	1
2	T	0	1
3	F	–	–
4	F	–	–
5	T	0	1
6	F	–	–

Cola: 2 1 5

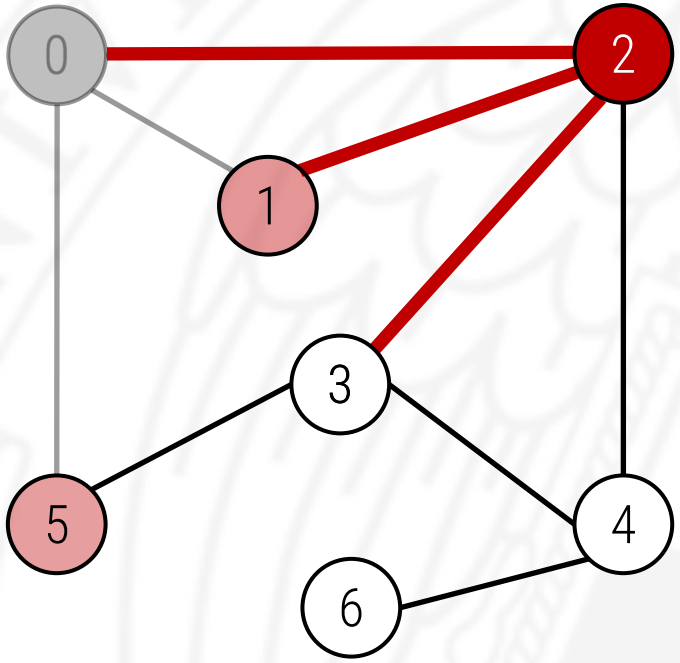
Recorrido en anchura



v	visitado	anterior	distancia
0	T	–	0
1	T	0	1
2	T	0	1
3	F	–	–
4	F	–	–
5	T	0	1
6	F	–	–

Cola: 2 1 5

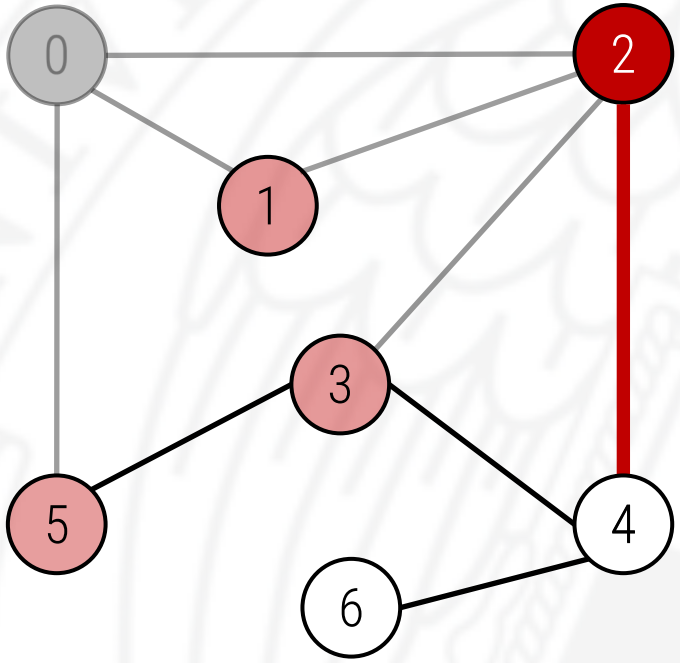
Recorrido en anchura



v	visitado	anterior	distancia
0	T	–	0
1	T	0	1
2	T	0	1
3	F	–	–
4	F	–	–
5	T	0	1
6	F	–	–

Cola: 1 5

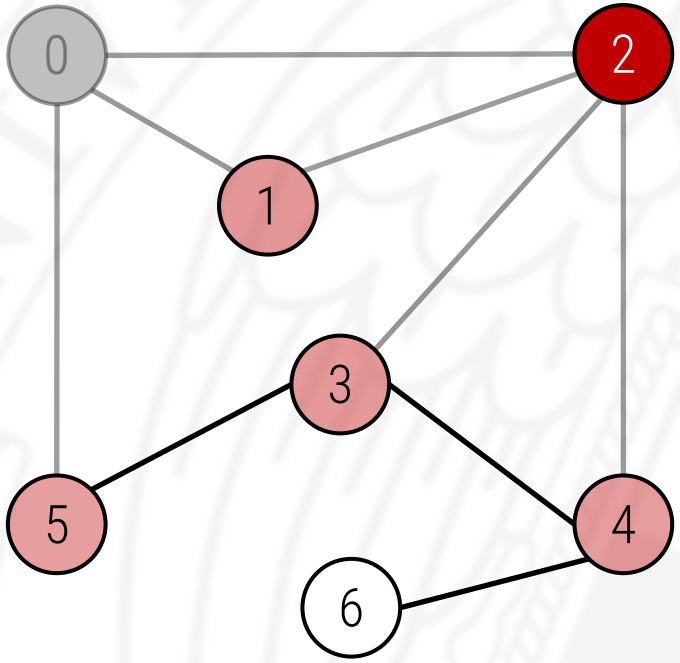
Recorrido en anchura



v	visitado	anterior	distancia
0	T	–	0
1	T	0	1
2	T	0	1
3	T	2	2
4	F	–	–
5	T	0	1
6	F	–	–

Cola: 1 5 3

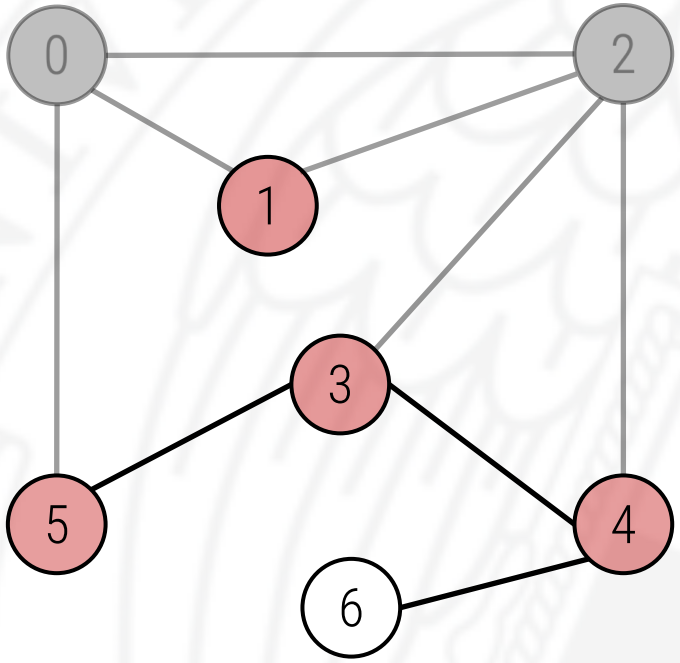
Recorrido en anchura



v	visitado	anterior	distancia
0	T	–	0
1	T	0	1
2	T	0	1
3	T	2	2
4	T	2	2
5	T	0	1
6	F	–	–

Cola: 1 5 3 4

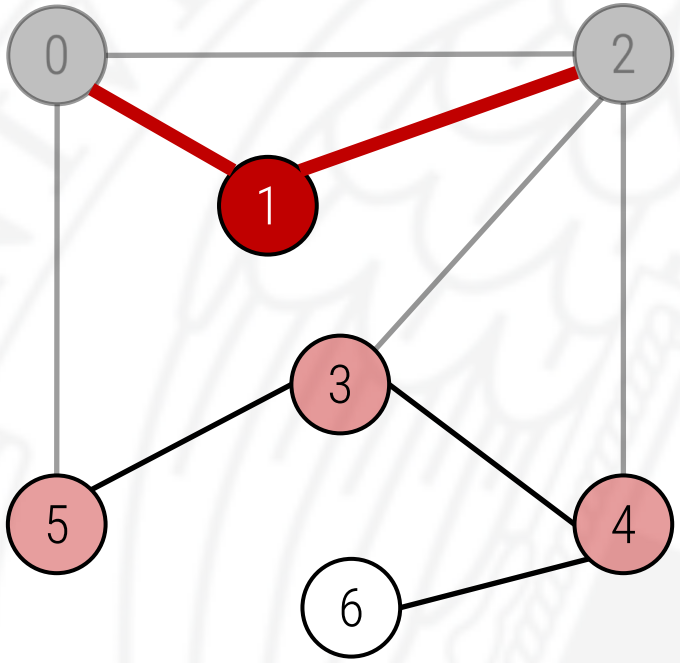
Recorrido en anchura



v	visitado	anterior	distancia
0	T	–	0
1	T	0	1
2	T	0	1
3	T	2	2
4	T	2	2
5	T	0	1
6	F	–	–

Cola: 1 5 3 4

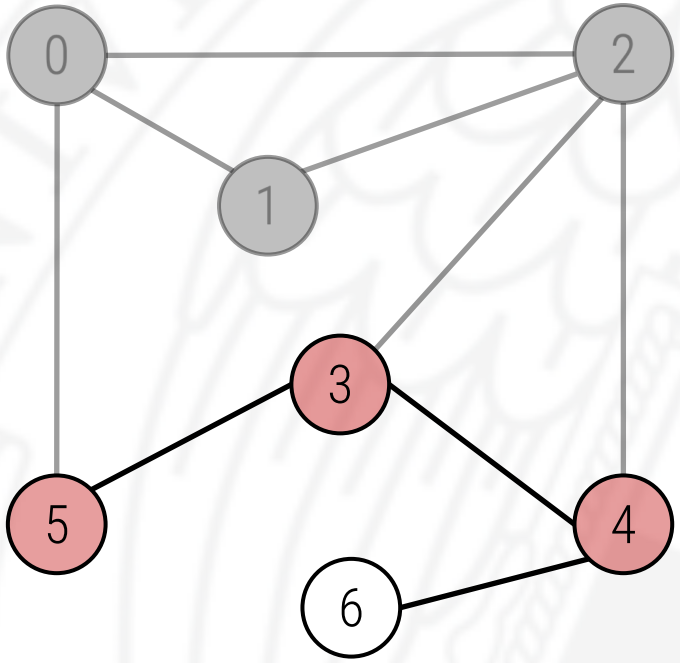
Recorrido en anchura



v	visitado	anterior	distancia
0	T	–	0
1	T	0	1
2	T	0	1
3	T	2	2
4	T	2	2
5	T	0	1
6	F	–	–

Cola: 5 3 4

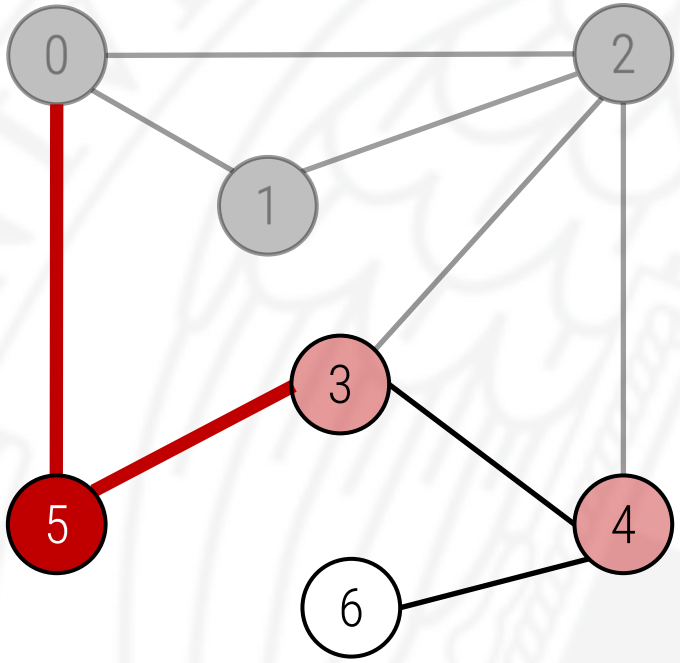
Recorrido en anchura



v	visitado	anterior	distancia
0	T	–	0
1	T	0	1
2	T	0	1
3	T	2	2
4	T	2	2
5	T	0	1
6	F	–	–

Cola: 5 3 4

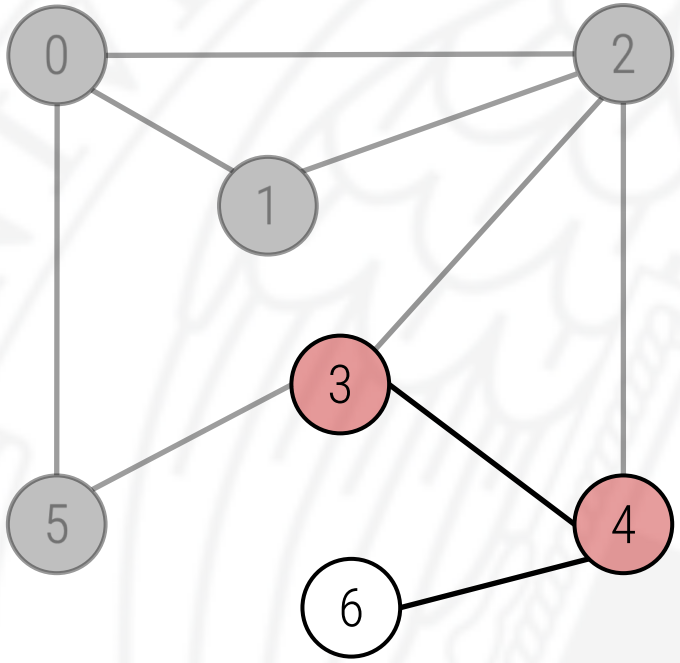
Recorrido en anchura



v	visitado	anterior	distancia
0	T	–	0
1	T	0	1
2	T	0	1
3	T	2	2
4	T	2	2
5	T	0	1
6	F	–	–

Cola: 3 4

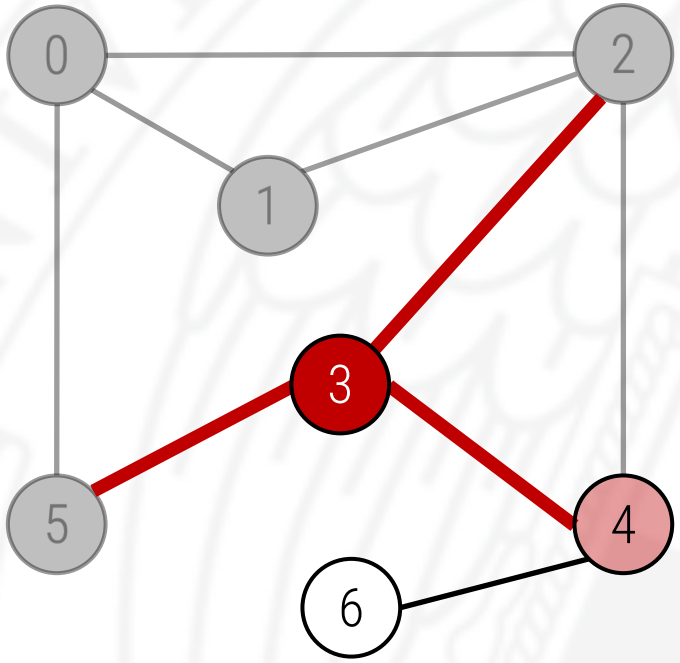
Recorrido en anchura



v	visitado	anterior	distancia
0	T	–	0
1	T	0	1
2	T	0	1
3	T	2	2
4	T	2	2
5	T	0	1
6	F	–	–

Cola: 3 4

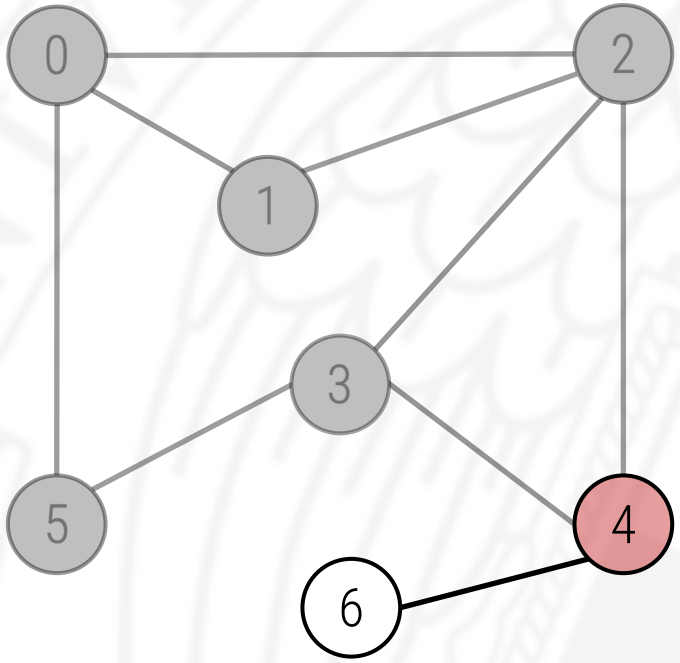
Recorrido en anchura



v	visitado	anterior	distancia
0	T	–	0
1	T	0	1
2	T	0	1
3	T	2	2
4	T	2	2
5	T	0	1
6	F	–	–

Cola: 4

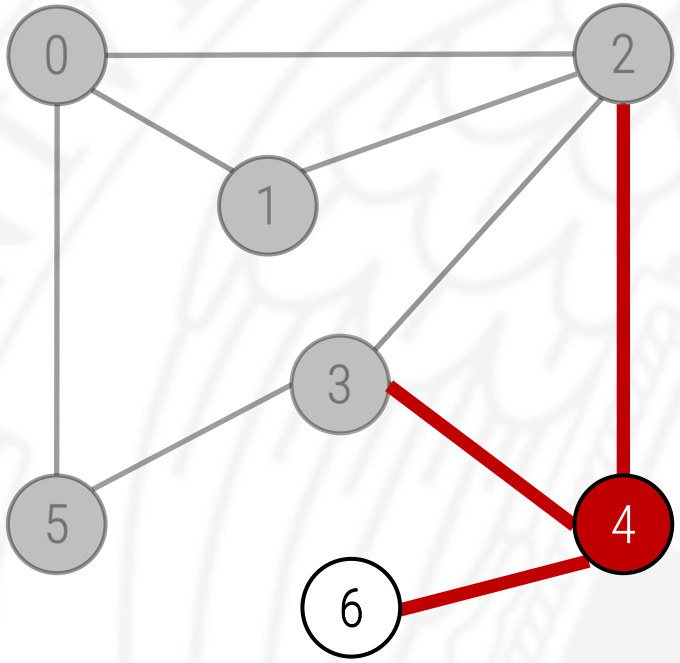
Recorrido en anchura



v	visitado	anterior	distancia
0	T	–	0
1	T	0	1
2	T	0	1
3	T	2	2
4	T	2	2
5	T	0	1
6	F	–	–

Cola: 4

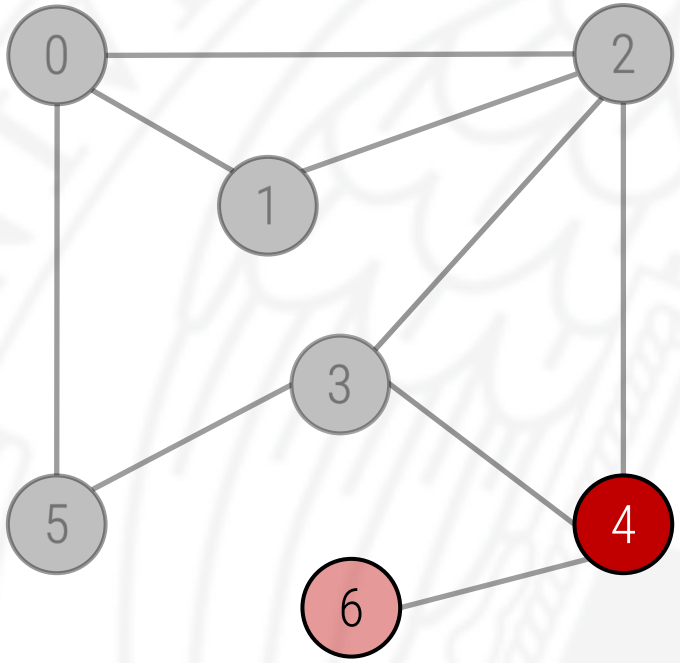
Recorrido en anchura



v	visitado	anterior	distancia
0	T	–	0
1	T	0	1
2	T	0	1
3	T	2	2
4	T	2	2
5	T	0	1
6	F	–	–

Cola:

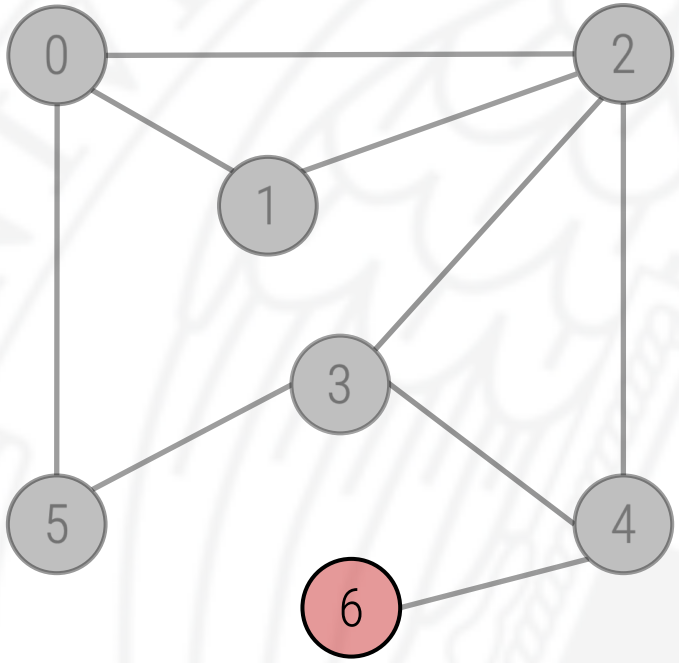
Recorrido en anchura



v	visitado	anterior	distancia
0	T	–	0
1	T	0	1
2	T	0	1
3	T	2	2
4	T	2	2
5	T	0	1
6	T	4	3

Cola: 6

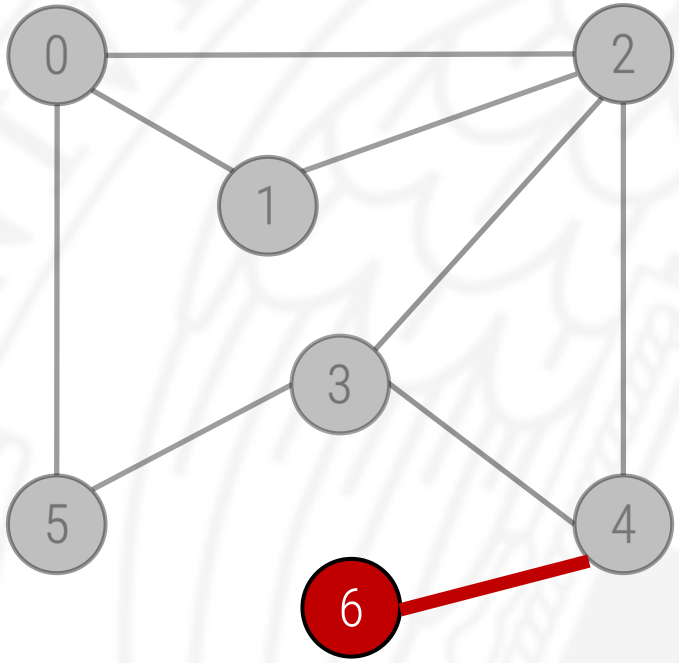
Recorrido en anchura



v	visitado	anterior	distancia
0	T	–	0
1	T	0	1
2	T	0	1
3	T	2	2
4	T	2	2
5	T	0	1
6	T	4	3

Cola: 6

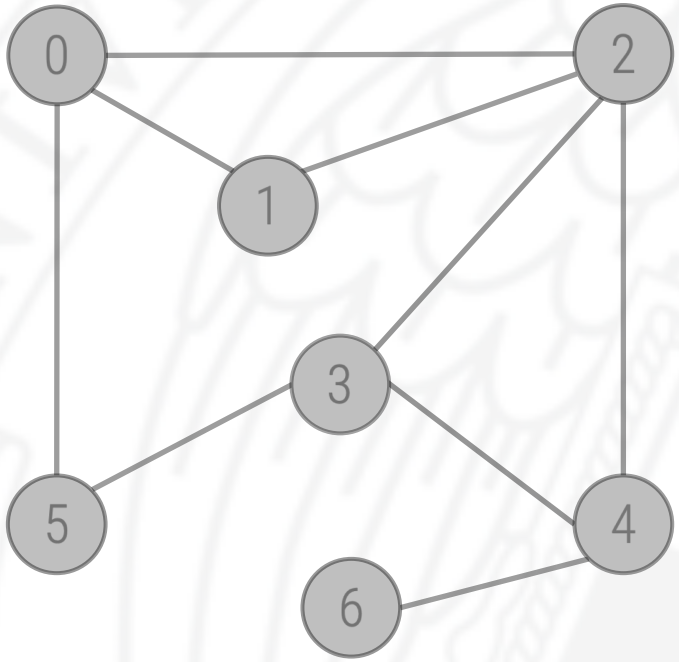
Recorrido en anchura



v	visitado	anterior	distancia
0	T	–	0
1	T	0	1
2	T	0	1
3	T	2	2
4	T	2	2
5	T	0	1
6	T	4	3

Cola: 6

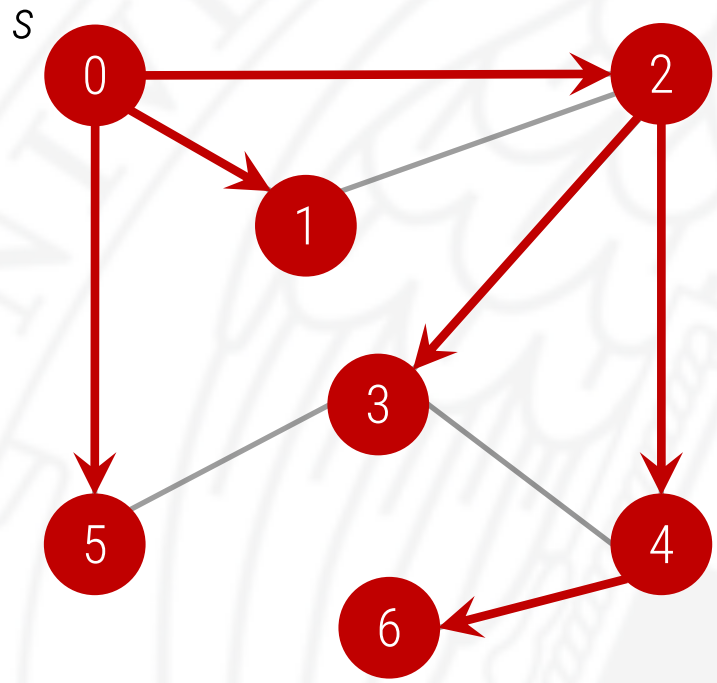
Recorrido en anchura



v	visitado	anterior	distancia
0	T	–	0
1	T	0	1
2	T	0	1
3	T	2	2
4	T	2	2
5	T	0	1
6	T	4	3

Cola:

Recorrido en anchura



v	visitado	anterior	distancia
0	T	–	0
1	T	0	1
2	T	0	1
3	T	2	2
4	T	2	2
5	T	0	1
6	T	4	3

Cola:

Caminos más cortos desde un origen

```
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];
   }
```

Caminos más cortos desde un origen

```
// 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;
}
```

Caminos más cortos desde un origen

```
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;
```

Caminos más cortos desde un origen

```
    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).