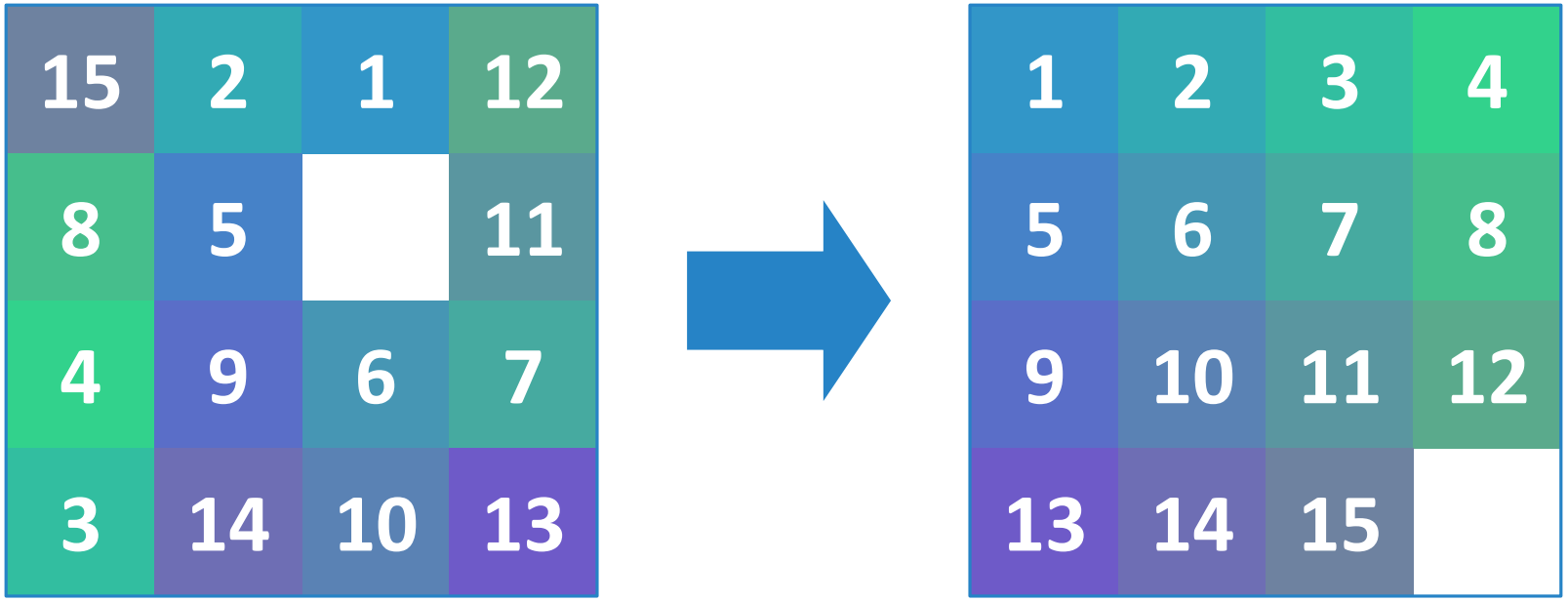
Puzzle para niños

Probablemente conozcan este tipo de puzzle:

Dada una instancia, ¿cuáles son los pasos que llevan a la solución?

Formalmente: El puzzle de 15

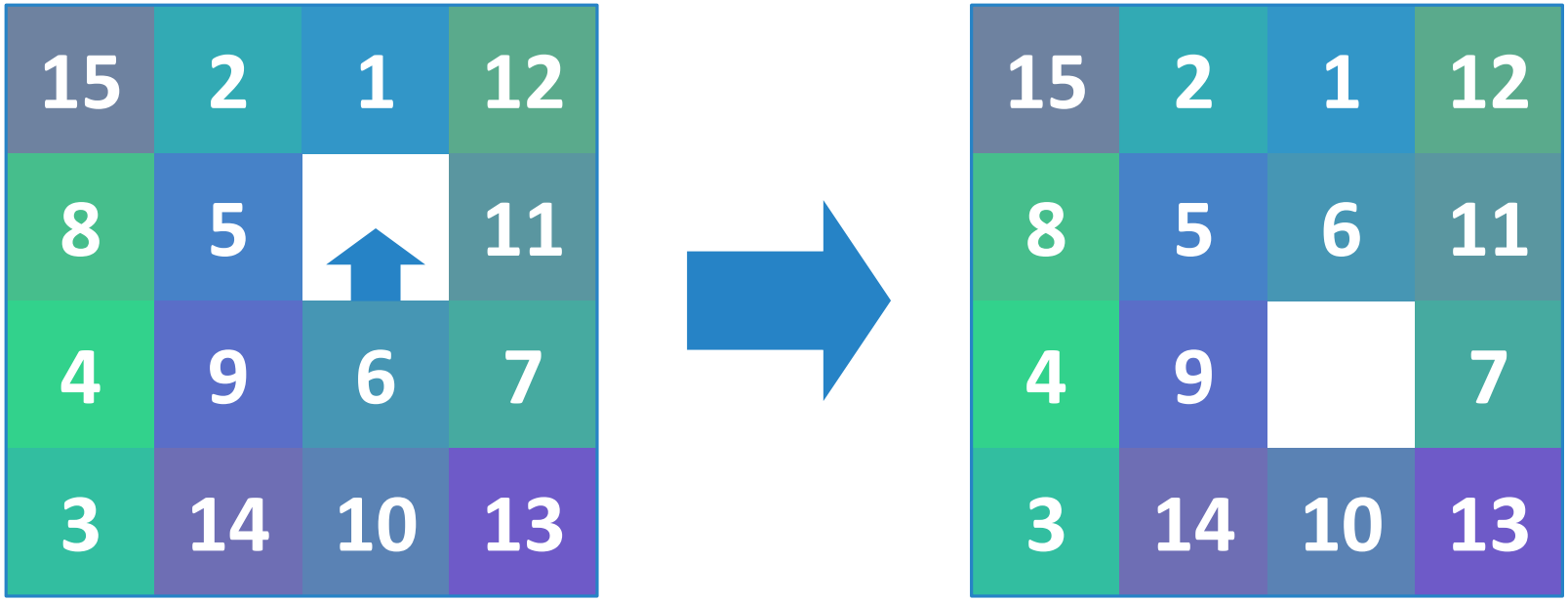
Este puzzle consta de 15 piezas con números en una grilla de 4×4



La idea es deslizar las piezas hasta dejar los números en orden

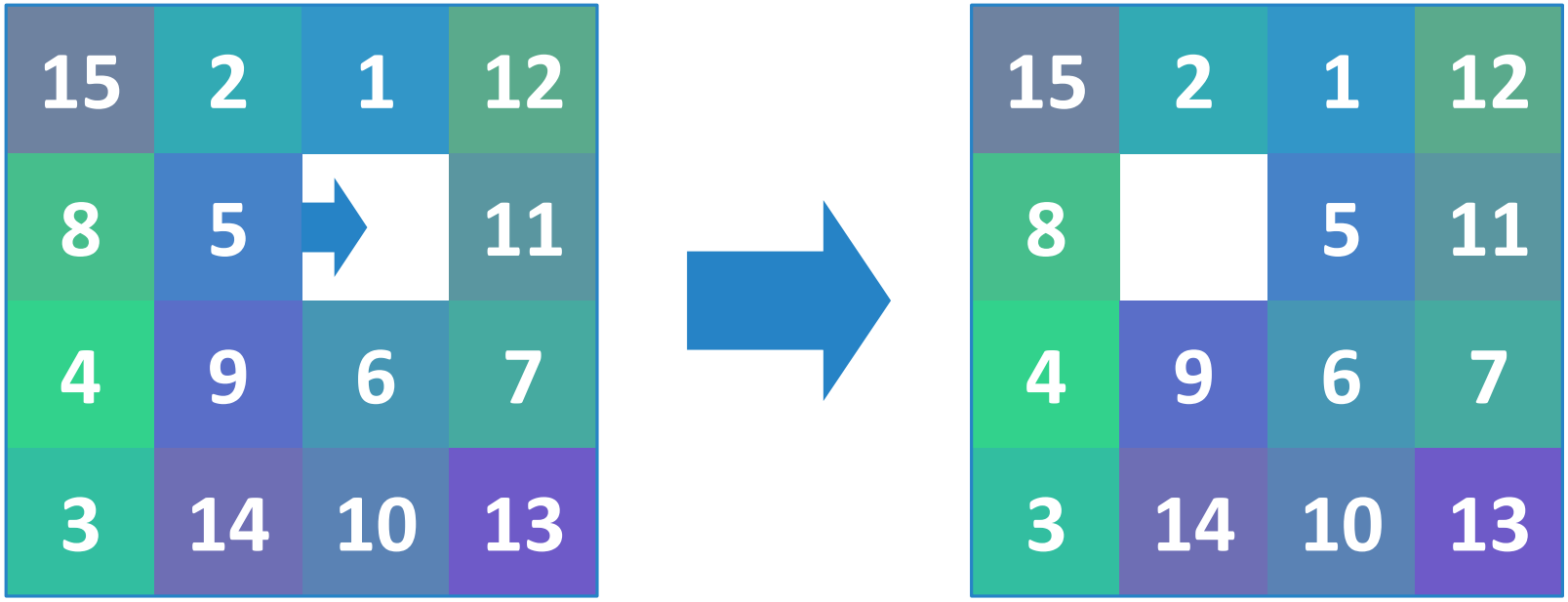
Operaciones del problema

1. Deslizar hacia arriba



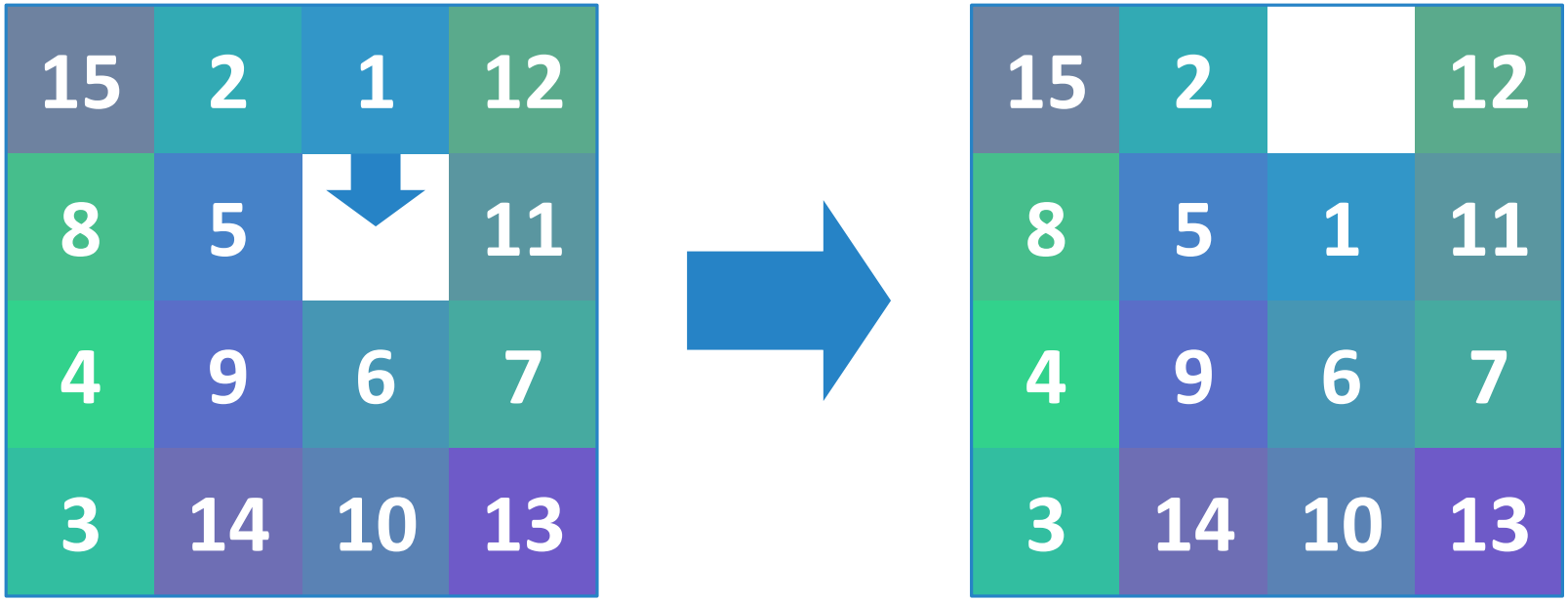
Operaciones del problema

2. Deslizar hacia la derecha



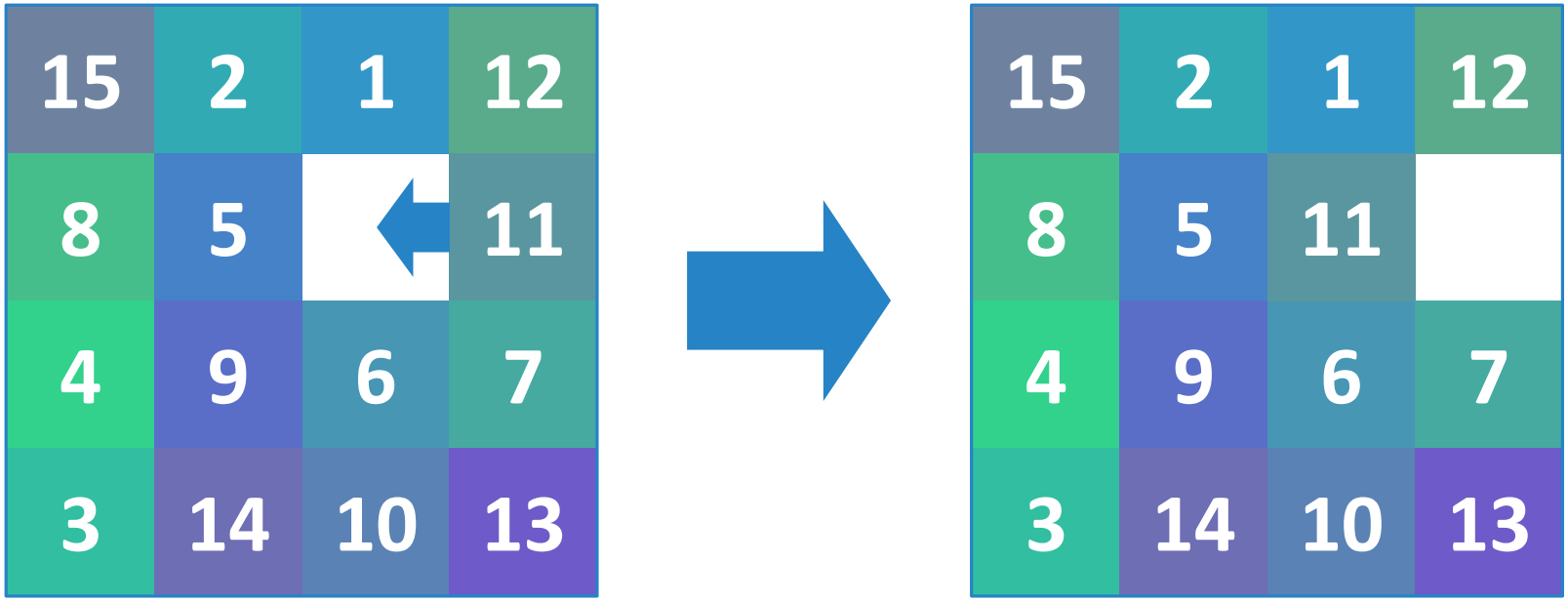
Operaciones del problema

3. Deslizar hacia abajo



Operaciones del problema

4. Deslizar hacia la izquierda



15 Puzzle

Entonces,

¿Cómo lo resolvemos?

Búsqueda

Podríamos hacer lo siguiente:

1. Construir un **grafo** que represente el problema
2. Utilizar **DFS** para **buscar** el camino a la solución

¿Cómo hacemos esto?

Búsqueda

Podríamos hacer lo siguiente:

1. Construir un **grafo** que represente el problema
2. Utilizar **DFS** para **buscar** el camino a la solución

¿Cómo hacemos esto?

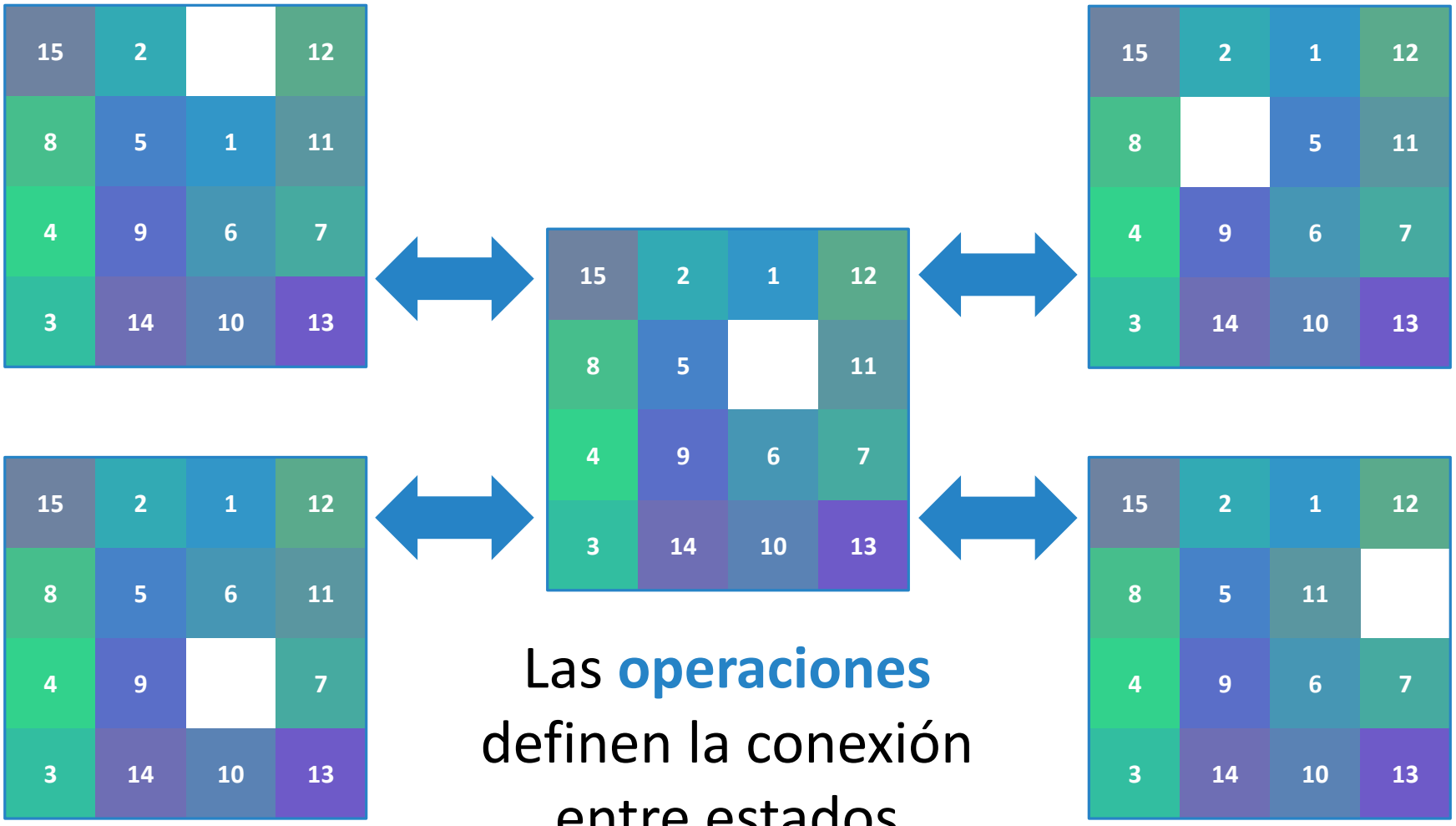
Grafo de estados

Un **grafo de estados** $G(V, E)$ se define de la siguiente manera:

Cada nodo en V es una **configuración** distinta del problema

Hay una arista de u a v si se puede pasar de u a v en un paso.

En el caso del Puzzle de 15



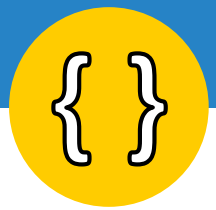
Las **operaciones** definen la conexión entre estados

Limitaciones

¿Qué tamaño tiene el grafo de estados del puzzle de 15?

¿Hay algún problema con eso?

Diccionarios al rescate



Los problemas de este tipo suelen tener **muchos** estados

Hay que generar el grafo a medida que se exploran los estados

Se necesita un **diccionario** para no generar estados repetidos

Búsqueda

Podríamos hacer lo siguiente:

1. Construir un **grafo** que represente el problema
2. Utilizar **DFS** para **buscar** el camino a la solución

¿Cómo hacemos esto?

```
buscar dfs(D, s, g):
    Insertar s en D
    if s = g, return true
    foreach operation op:
        t ← op(s)
        if t ∈ D, continue
        t.parent ← s,    t.operation ← op
        if buscar dfs(D, t, g):
            return true:
    return false
```

El Puzzle de 15++

Dada una configuración del puzzle de 15

¿Cuáles son los pasos necesarios para llegar a la solución…

… en la menor cantidad de pasos posible?

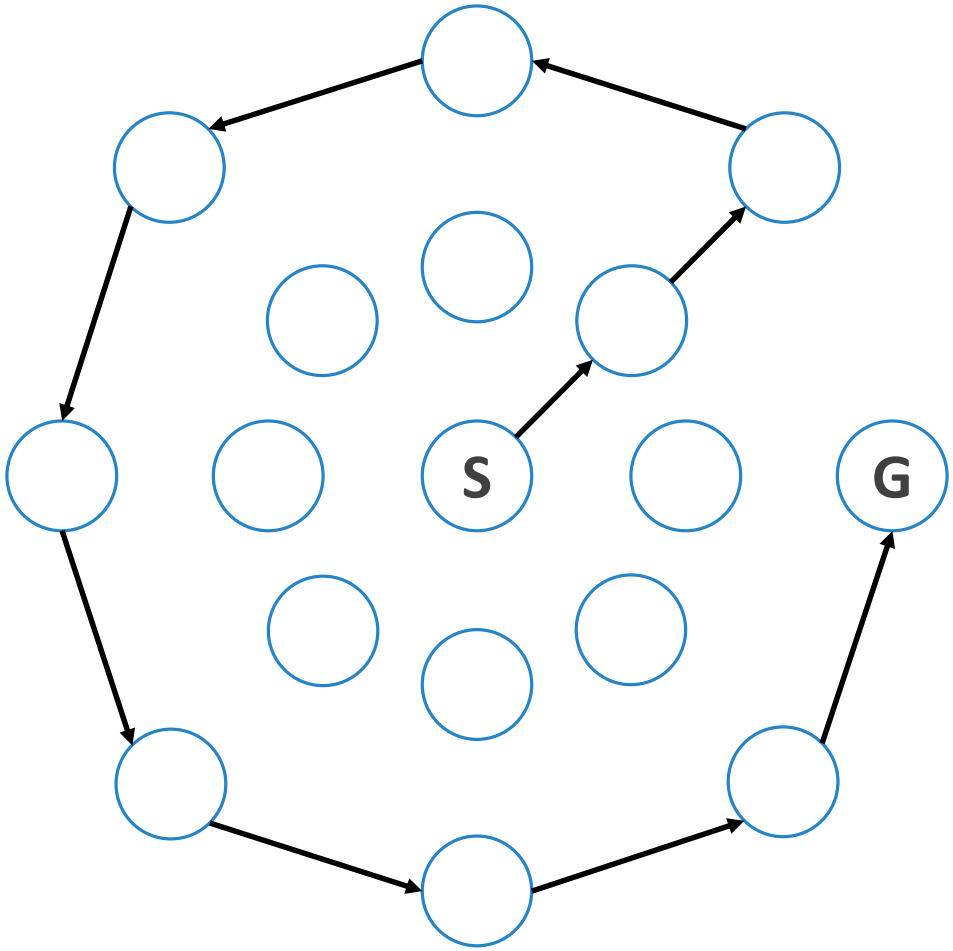
Ruta más corta

Si la **distancia** es el largo de la **ruta más corta** entre dos nodos,

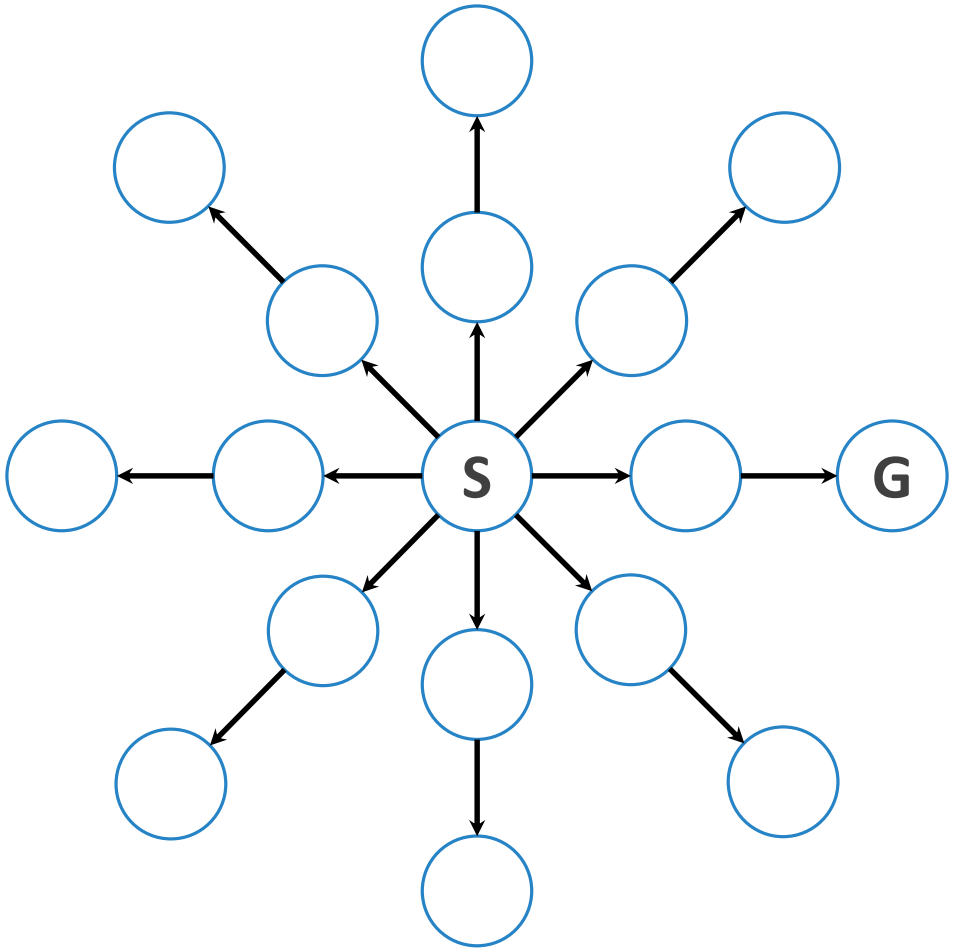
¿Está la solución a distancia 1 del origen? ¿Y a distancia 2?

¿Cómo podemos responder esa pregunta para una distancia n?

Tenemos: Depth First Search



Queremos: Breadth First Search



La idea del algoritmo de BFS

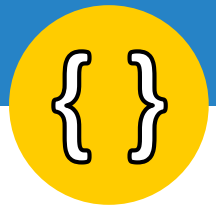
Partiendo de $i = 1$:

1. Generar los estados a distancia i del origen
2. Si alguno de esos es el destino, estamos listos
3. Si no, incrementar i en 1 y volver a 1.

¿Cómo hacemos esto de manera eficiente?

```
buscar bfs(D, s, g):
    Open ← una cola vacía.  D ← un diccionario vacío.
    Insertar s en Open y en D
    while Open ≠ ∅:
        s ← el siguiente elemento de Open
        foreach operation op:
            t ← op(s)
            if t ∈ D, continue
            t.parent ← s,    t.operation ← op
            if t = g, return true
            Insertar t en D y en Open
    return false
```

Relación entre DFS y BFS



Si reemplazamos esa **cola** en **BFS** por un **stack**, tenemos **DFS**.

Es una forma de implementar **DFS** de manera iterativa

La complejidad de ambos algoritmos es la misma

Control

Demuestra que cuando **BFS** genera **cualquier** nodo del grafo, lo ha encontrado por la **ruta más corta** posible desde el nodo de origen.