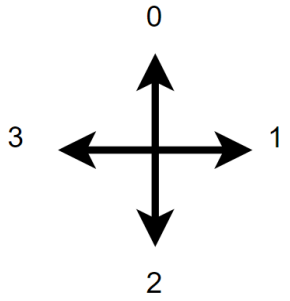
PASOS PARA LA REALIZACIÓN DE LA TAREA RECORRIDO DEL LABERINTO USANDO BFS

1 Crear el **MAZE** como un numpy array de 10x10 relleno de ceros.
2 Ponerle los bordes a 1
3 Crea la matriz **MARK** copia de MAZE.

Puedes visualizarla llamando a la siguiente función (importando las librerías):

```
def visualize_example(x):
      plt.figure()
      plt.imshow(x)
      plt.colorbar()
      plt.grid(False)
      plt.show()
```

4 Crear la matriz de moviviento MOVE (puede ser necesario trasponer la matriz para representar correctamente los movimientos en la pantalla).



0	-1
1	0
0	1
-1	0

5 Crea una clase Agente con los atributos **x** e **y** que almacenan su posición actual y el método **mover(2)** al que se le pasa un valor de 0 a 3 y modifica sus coordenadas de la manera indicada en las flechas.

Ejemplo: Partiendo de la posición x,y saltaría a G,H con el movimiento 2 (bajar).
G=x+MOVE[2,0]
H=y+MOVE[2,1]

6 Repasar Pilas y Colas en Python.

https://www.youtube.com/watch?v=6iGcnlwNwKE&t=505s

Para la búsqueda en anchura se usa una cola y para el recorrido en profundidad la Pila.

En la cola se almacenan las posiciones visitadas, para ello es interesante crear una clase llamada **Place** que contendrá los atributos la **fila** y **columna**.

```
CLASS PLACE:
        DEF __INIT__(SELF, FILA, COLUMNA):
        SELF.FILA = FILA
        SELF.COLUMNA = COLUMNA
        DEF MOSTRAR(SELF):
                PRINT("( " + SELF.FILA+","+SELF.COLUMNA+")")
```

PASOS PARA LA REALIZACIÓN DE LA TAREA RECORRIDO DEL LABERINTO USANDO BFS

Nota: Puede prescindirse de la matriz MARK añadiendo el campo **mov** a la clase Place, que contiene el último movimiento realizado desde ese sitio.

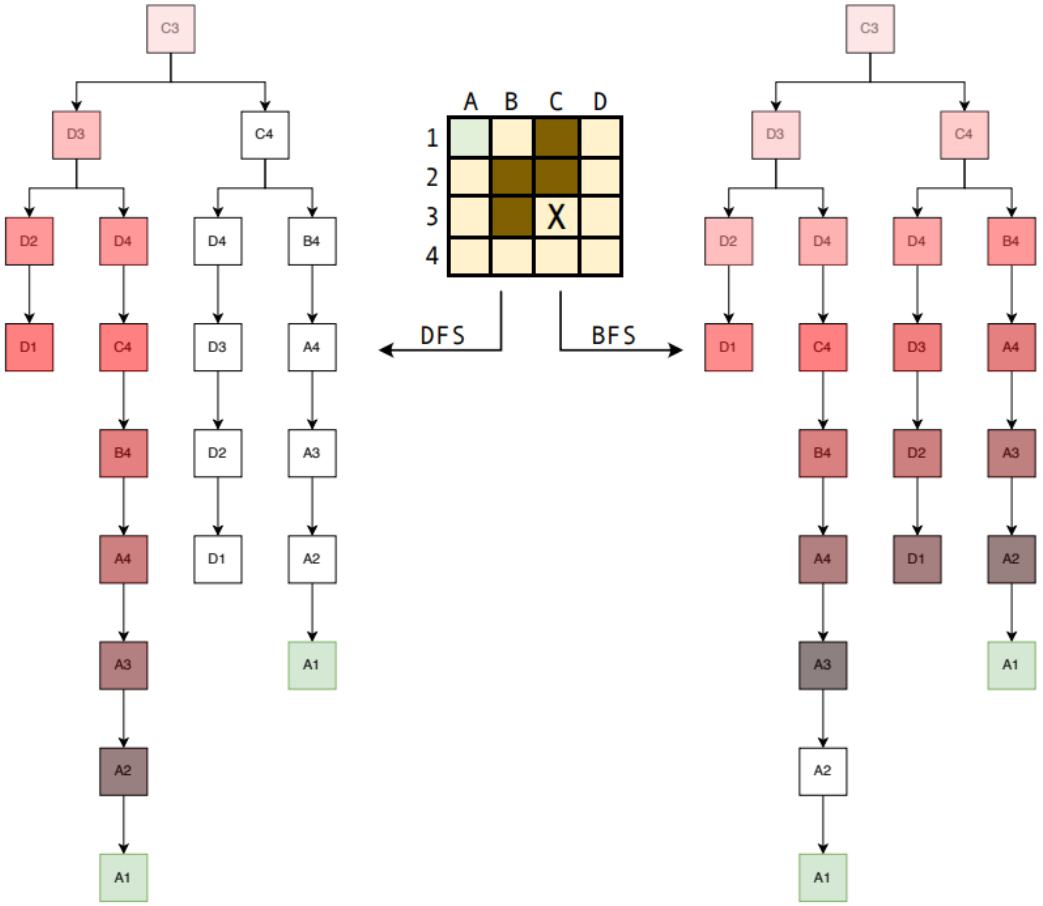


Figura 2: Búsqueda en profundidad (izquierda) y búsqueda en anchura (derecha) aplicadas a la resolución de un laberinto.