

BFS

```
def BFS(lab:Labyrinth):  
    print('Starting BFS')  
  
    trail = []  
    visited = []  
    pending = []  
  
    pending.append([lab.getStartCell()])  
    visited.append(lab.getStartCell())  
  
    while len(pending) != 0:  
  
        tmp = pending.pop(0)  
  
        if (tmp[-1] == lab.getEndCell()):  
            trail = tmp  
            break  
  
        for c in tmp[-1].getChildren():  
            if c not in visited:  
                pending.append(tmp + [c])  
                visited.append(c)  
  
    return trail
```

Big O: $O(b^d)$

Precondition: lab: Labyrinth; lab.getStartCell() != null and lab.getEndCell() != null
Existeix un laberint; Existeixen una cel·la d'entrada i sortida.

Postcondition: S'ha trobat la cel·la de sortida o s'han visitat totes les cel·les. Si existeix, el camí resultant de S a E és el més curt possible.

DFS

```
def keepSearching(c:Cell, lab:Labyrinth, trail, visited):  
  
    trail.append(c)  
  
    if c == lab.getEndCell():  
        return trail  
  
    for child in c.getChildren():  
        if child not in visited:  
            visited.append(child)  
            return keepSearching(child, lab, trail, visited)  
  
    return keepSearching(trail[-2], lab, trail[:-2], visited)
```

Big O: $O(b * m)$

Precondition: lab: Labyrinth; lab.getStartCell() != null and lab.getEndCell() != null

Visitar cada node possible d'un camí de nodes que no han estat visitats pel node v. V no es visitat

Postcondition: S'ha trobat la cel·la de sortida entre el camí S a E, si existeix