Algorithm 1: Breadth-First-Search (BFS)

Input: graph $G = (V, E)$, $V = \{1, \ldots, n\}$ given as adjacency list, root node $r \in V$

Output: level function $L : V \to \mathbb{Z}_{\geq 0} \cup \{\infty\}$, predecessor function $\text{pred} : V \to V \cup \{0\}$

```
1  foreach v ∈ V \ {r} do
2  |   L(v) ← ∞
3  |   pred(v) ← 0
4  |   seen(v) ← 0
5  L(r) ← 0
6  pred(r) ← 0
7  seen(r) ← 1
8  Q ← ∅
9  push_back(Q, r)
10 while Q ≠ ∅ do
11 |   u ← pop_front(Q)
12 |   foreach v ∈ Adj(u) do
13 |   |   if seen(v) = 0 then
14 |   |   |   seen(v) ← 1
15 |   |   |   L(v) ← L(u) + 1
16 |   |   |   pred(v) ← u
17 |   |   |   push_back(Q, v)
```

Algorithm 2: Depth-First-Search (DFS)

Input: graph $G = (V, E)$, $V = \{1, \ldots, n\}$ given as adjacency list

Output: predecessor function pred : $V \to V \cup \{0\}$

```
1 foreach v ∈ V do
2 |   pred(v) ← 0
3 |   seen(v) ← 0
4 foreach v ∈ V do
5 |   if seen(v) = 0 then
6 |   |   DFSvisit(G, v)
```

Function DFSvisit(G,r)

Input: graph $G = (V, E)$ given as adjacency list, root node $r \in V$

```
1 seen(r) ← 1
2 foreach v ∈ Adj(r) do
3 |   if seen(v) = 0 then
4 |   |   pred(v) = r
5 |   |   DFSvisit(G, v)
```