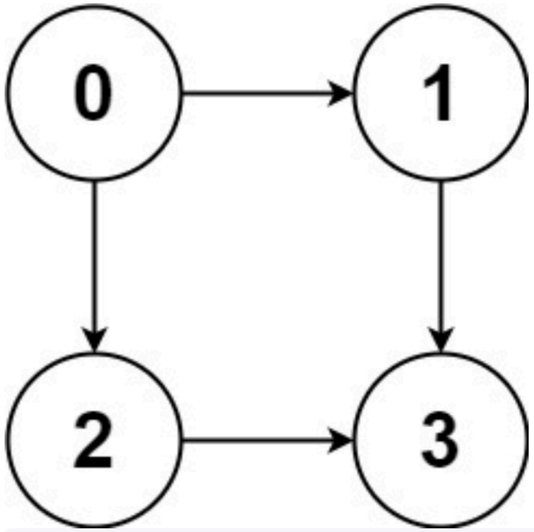
797. All Paths From Source to Target

Medium 4509 120 Add to List Share

Given a directed acyclic graph (**DAG**) of `n` nodes labeled from `0` to `n - 1`, find all possible paths from node `0` to node `n - 1` and return them in **any order**.

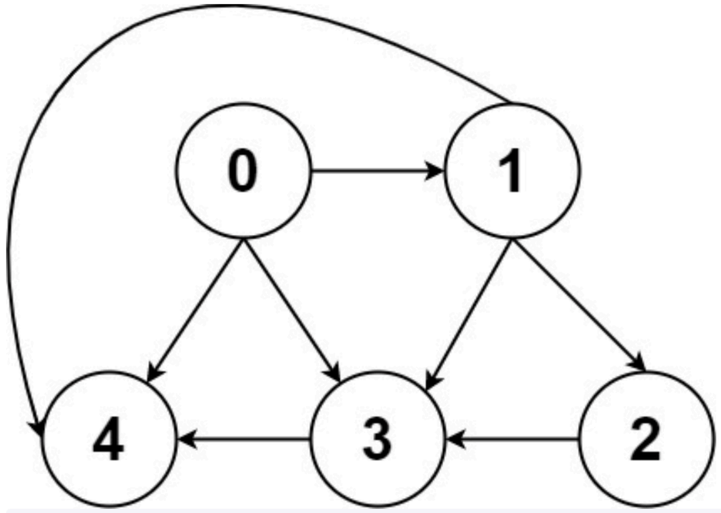
The graph is given as follows: `graph[i]` is a list of all nodes you can visit from node `i` (i.e., there is a directed edge from node `i` to node `graph[i][j]`).

Example 1:




```
Input: graph = [[1,2],[3],[3],[]]
Output: [[0,1,3],[0,2,3]]
Explanation: There are two paths: 0 -> 1 -> 3 and 0 -> 2 -> 3.
```

Example 2:

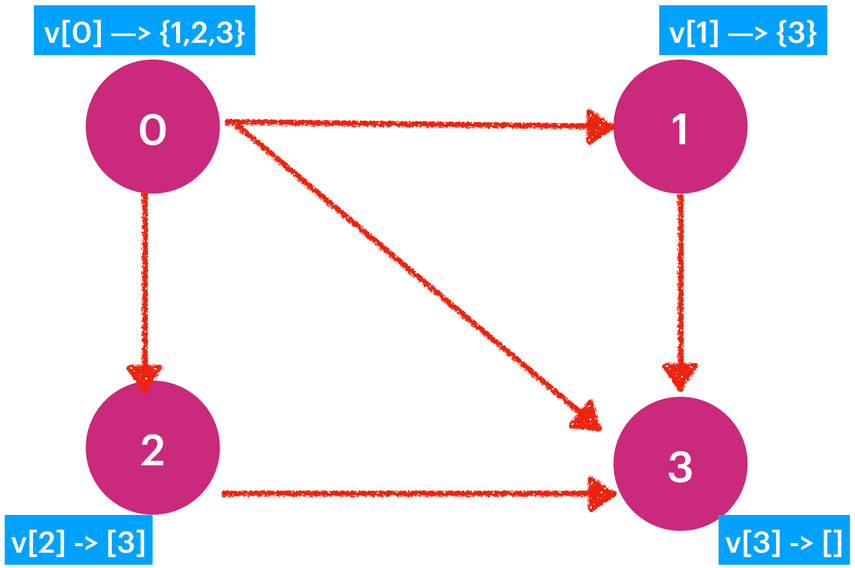



```
Input: graph = [[4,3,1],[3,2,4],[3],[4],[]]
Output: [[0,4],[0,3,4],[0,1,3,4],[0,1,2,3,4],[0,1,4]]
```

Constraints:

- `n == graph.length`
- `2 <= n <= 15`
- `0 <= graph[i][j] < n`
- `graph[i][j] != i` (i.e., there will be no self-loops).
- All the elements of `graph[i]` are **unique**.
- The input graph is **guaranteed** to be a **DAG**.

Direct Acyclic Graph [DAG]

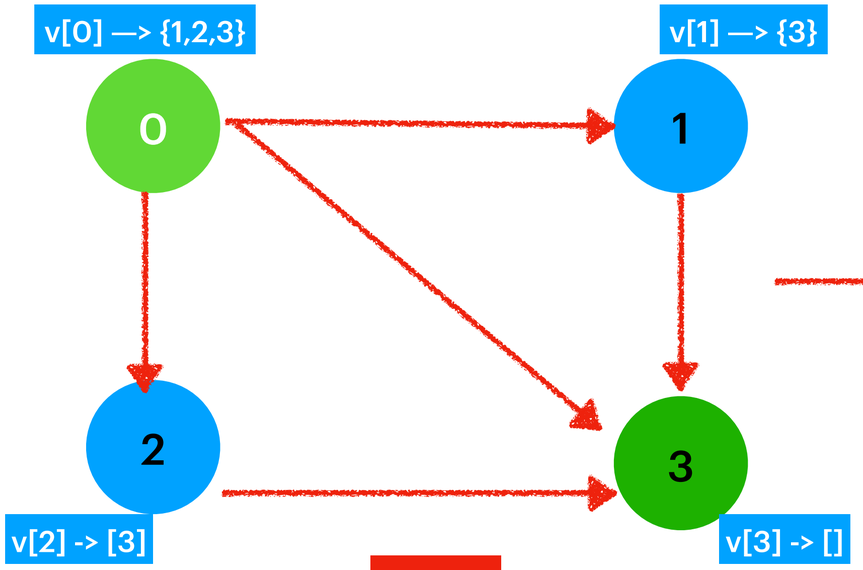


Find Out All the possible Paths from V[0] to V[n-1]

0 —> 3
0 —> 1 —> 3
0 —> 2 —> 3

Expected Output : result -> List< List<Integer> >

Bredth First Search [BFS]



Input

queue : [0]
pathQueue: [[0]]

As my queue is not Empty:

currentVertex = queue.poll(); 0
currentPath = pathQueue.poll(); [0]

For the currentVertex:0 we Found connections {1,2,3}

Iteration 1 :
Visited vertex 1 which is not target .
queue.add(1)
List<Integer> list = new ArrayList<>(currentPath);
list.add(1);
pathQueue.add(list);
queue : [1] ,
pathQueue: [0,1]

Iteration 2:
Visited vertex 2 which is not target .
queue.add(2)
List<Integer> list = new ArrayList<>(currentPath);
list.add(2);
pathQueue.add(list);
queue : [1,2] ,
pathQueue: [[0,1] , [0,2]]

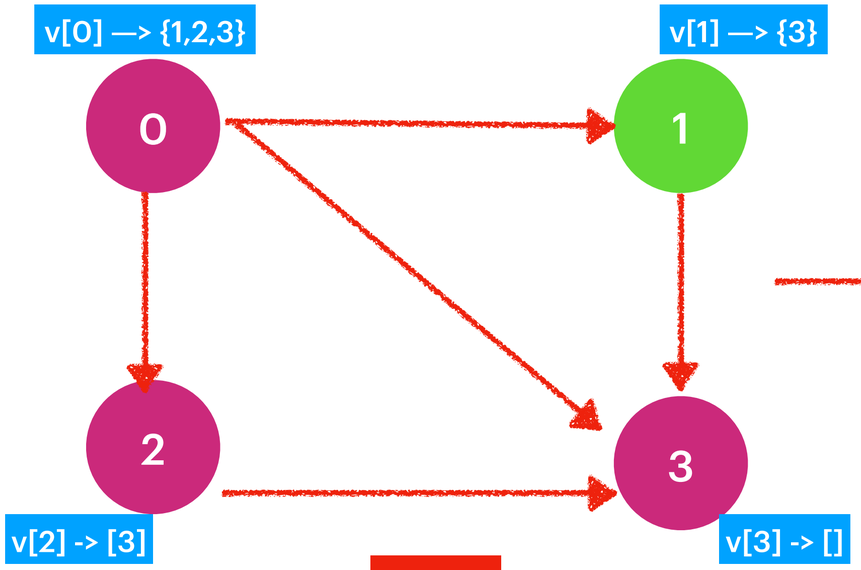
Iteration 3
Visited vertex 3 which our target .
List<Integer> list = new ArrayList<>(currentPath);
list.add(3);
result.add(list);

Output

queue: [1, 2]
pathQueue: [[0,1] , [0,2]]
result = [[0,3]]

Expected Output : result -> List< List<Integer> >

Bredth First Search [BFS]



Input

queue: [1, 2]
pathQueue: [[0,1] , [0,2]]
result = [[0,3]]

As my queue is not Empty:

currentVertex = queue.poll(); 1
currentPath = pathQueue.poll(); [0,1]

For the currentVertex:1 we Found connections {3}

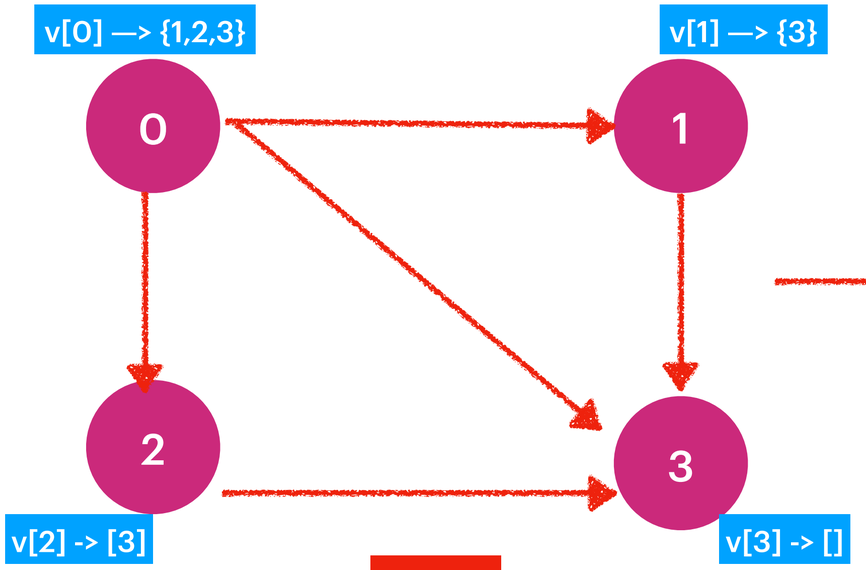
Iteration 1
Visited vertex 3 which our target .
List<Integer> list = new ArrayList<>(currentPath);
list.add(3);
result.add(list);

Output

queue: [2]
pathQueue: [[0,2]]
result = [[0,3], [0,1,3]]

Expected Output : result -> List< List<Integer> >

Bredth First Search [BFS]



Input

queue: [2]
pathQueue: [[0,2]]
result = [[0,3], [0,1,3]]

As my queue is not Empty:

currentVertex = queue.poll(); 2
currentPath = pathQueue.poll(); [0,2]

For the currentVertex:2 we Found connections {3}

Iteration 1
Visited vertex 3 which our target .
List<Integer> list = new ArrayList<>(currentPath);
list.add(3);
result.add(list);

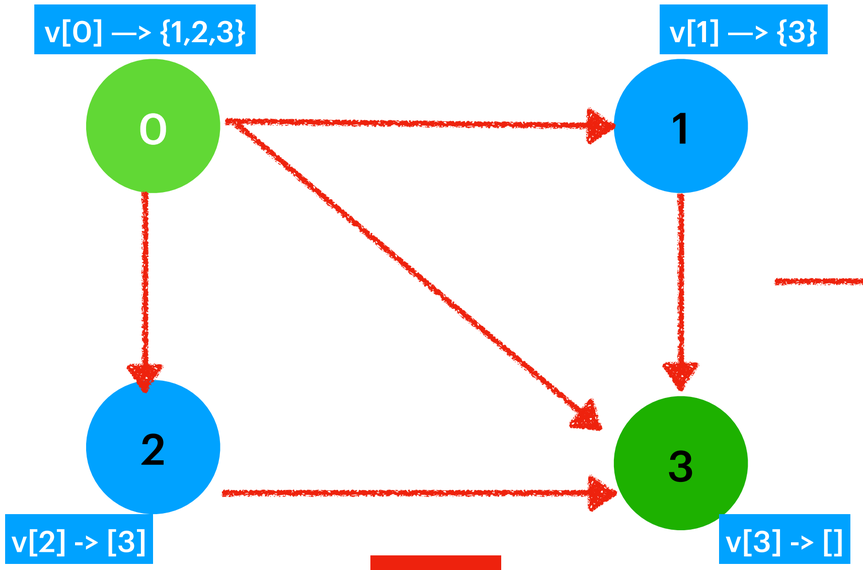
Output

queue: []
pathQueue: [ ]
result = [[0,3], [0,1,3], [0,2,3]]

Queue is Empty : so return the result : [[0,3] [0,1,3] [0,2,3]]

Expected Output : result -> List< List<Integer> >

Depth First Search [DFS]



Input

stack : [0]
pathStack : [[0]]

As my stack is not Empty:

currentVertex = stack.pop(); 0
currentPath = stack.pop(); [0]

For the currentVertex:0 we Found connections {1,2,3}

Iteration 1 :
Visited vertex 1 which is not target .
stack.push(1)
List<Integer> list = new ArrayList<>(currentPath);
list.add(1);
pathStack.add(list);
stack : [1] ,
pathStack: [[0,1]]

Iteration 2 :
Visited vertex 2 which is not target .
stack.push(2)
List<Integer> list = new ArrayList<>(currentPath);
list.add(2);
pathStack.add(list);
stack : [1 —> 2] ,
pathStack: [[0,1] —> [0,2]]

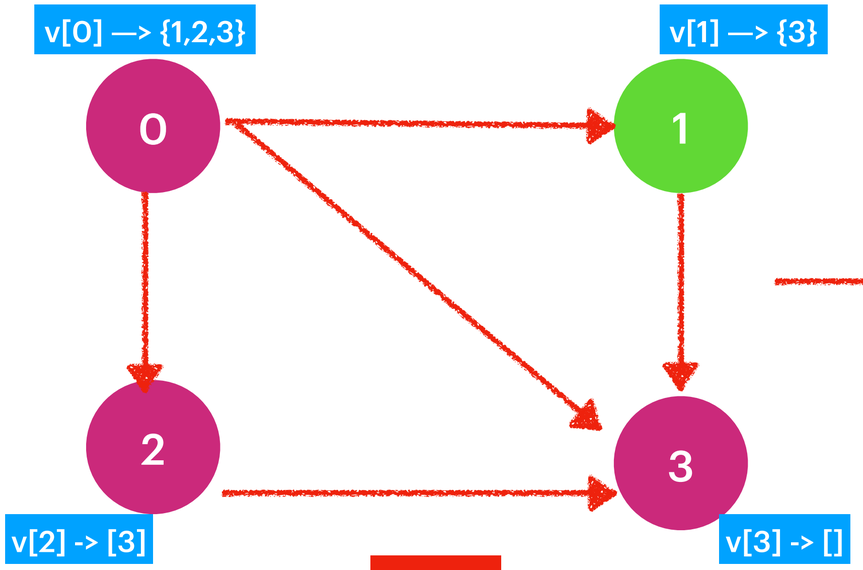
Iteration 3 :
Visited vertex 3 which is the target .
Add currentPath to result
List<Integer> list = new ArrayList<>(currentPath);
list.add(2);
result.add(list)

Output

stack: [1 —> 2]
pathStack: [[0,1] —-> [0,2]]
result = [[0,3]]

Expected Output : result -> List< List<Integer> >

Depth First Search [DFS]



Input

stack: [1 —> 2]
pathStack: [[0,1] —-> [0,2]]
result = [[0,3]]

As my stack is not Empty:

currentVertex = stack.pop(); 2
currentPath = pathStack.pop(); [0,2]

For the currentVertex:2 we Found connections {3}

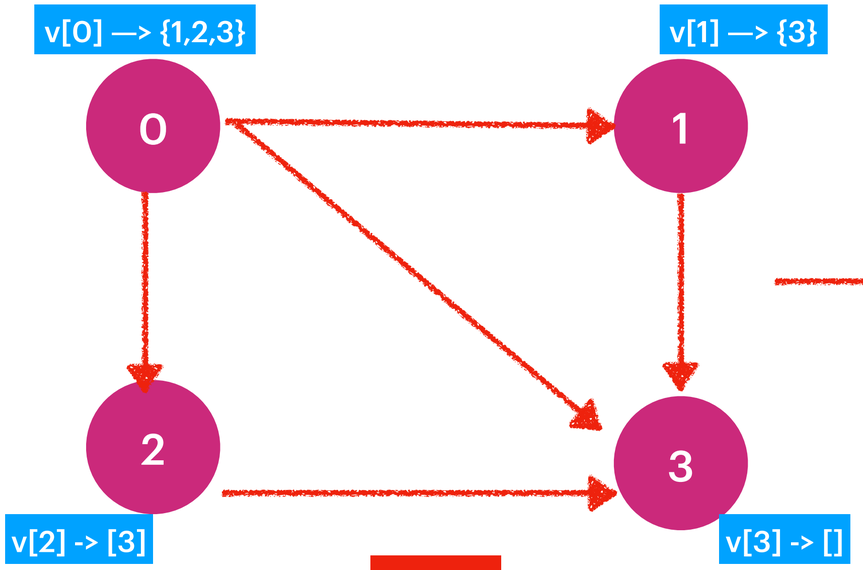
Iteration 1
Visited vertex 3 is our target .
List<Integer> list = new ArrayList<>(currentPath);
list.add(3);
result.add(list);

Output

stack: [1]
pathStack: [[0,1]]
result = [[0,3], [0,2,3]]

Expected Output : result -> List< List<Integer> >

Depth First Search [DFS]



Input

stack: [1]
pathStack: [[0,1]]
result = [[0,3], [0,2,3]]

As my stack is not Empty:

currentVertex = stack.pop(); 1
currentPath = pathStack.pop(); [0,1]

For the currentVertex:1 we Found connections {3}

Iteration 1
Visited vertex 3 is our target .
List<Integer> list = new ArrayList<>(currentPath);
list.add(3);
result.add(list);

Output

stack: []
pathStack: []
result = [[0,3], [0,2,3] , [0,1,3]]

Stack is Empty : so return the result : [[0,3] [0,1,3] [0,2,3]]