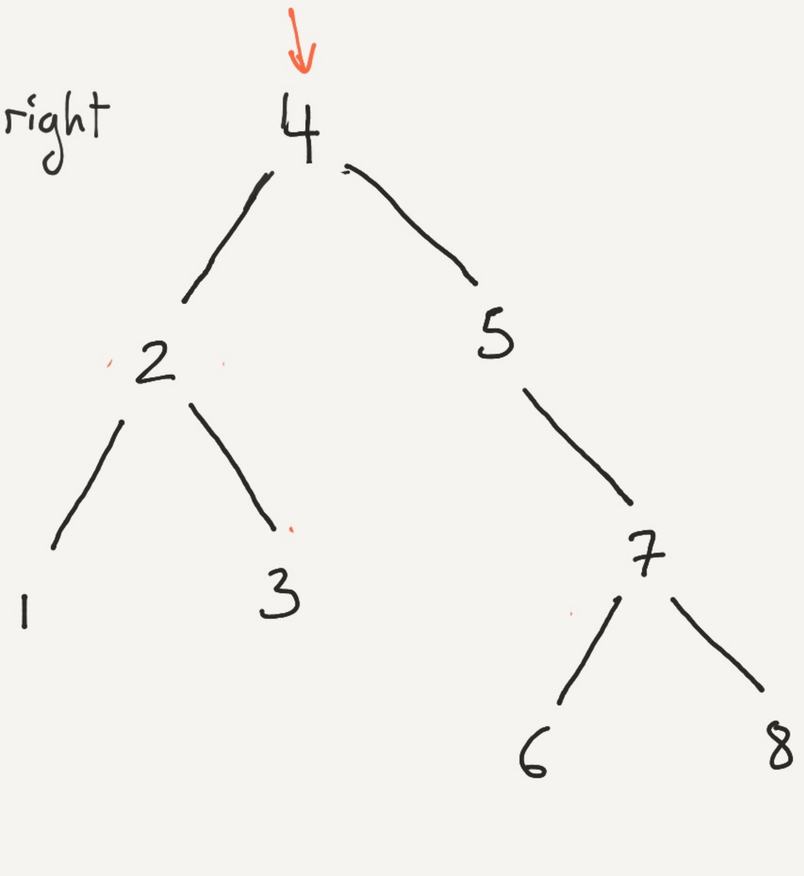
Binary Trees

```
TreeNode {
  int val
  TreeNode left, right
}
```



DFS – Depth first search

traverse left operate on node traverse right

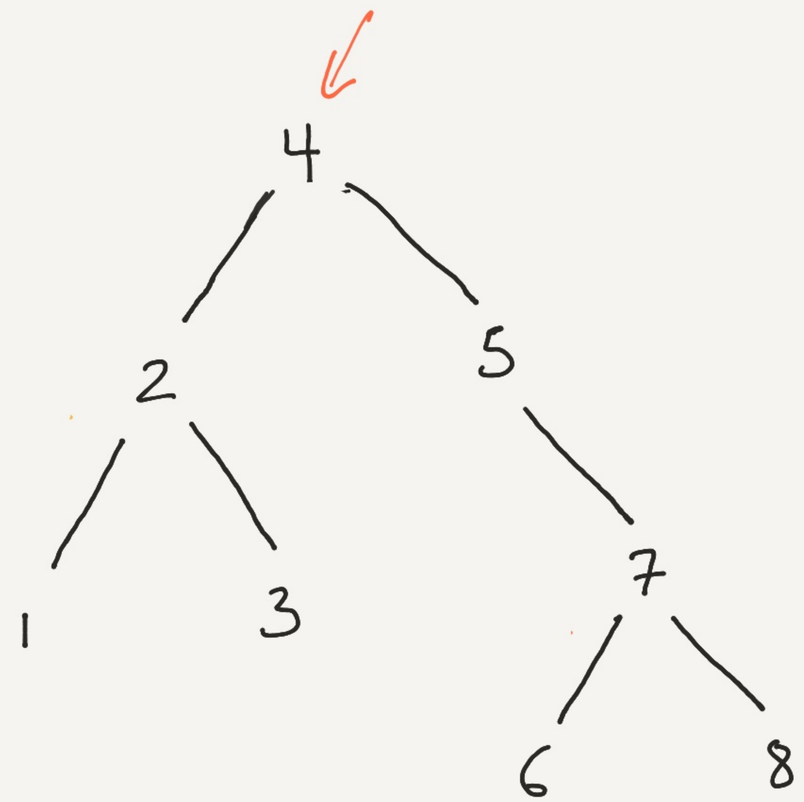
BFS – Breadth first search / Level-order traversal

4, 2, 5, 1, 3, 7, 6, 8

Binary Trees

BFS

```
TreeNode {
  int data
  TreeNode left, right
}
```



add root to q

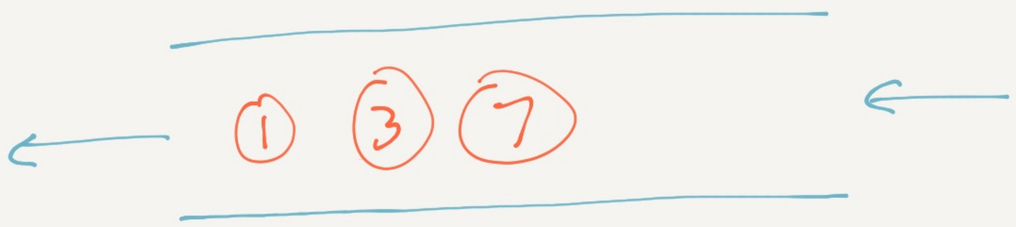
while q is not empty

  curr = pop off front of q

  add curr value to result list

  add left & right child nodes to q

return result list



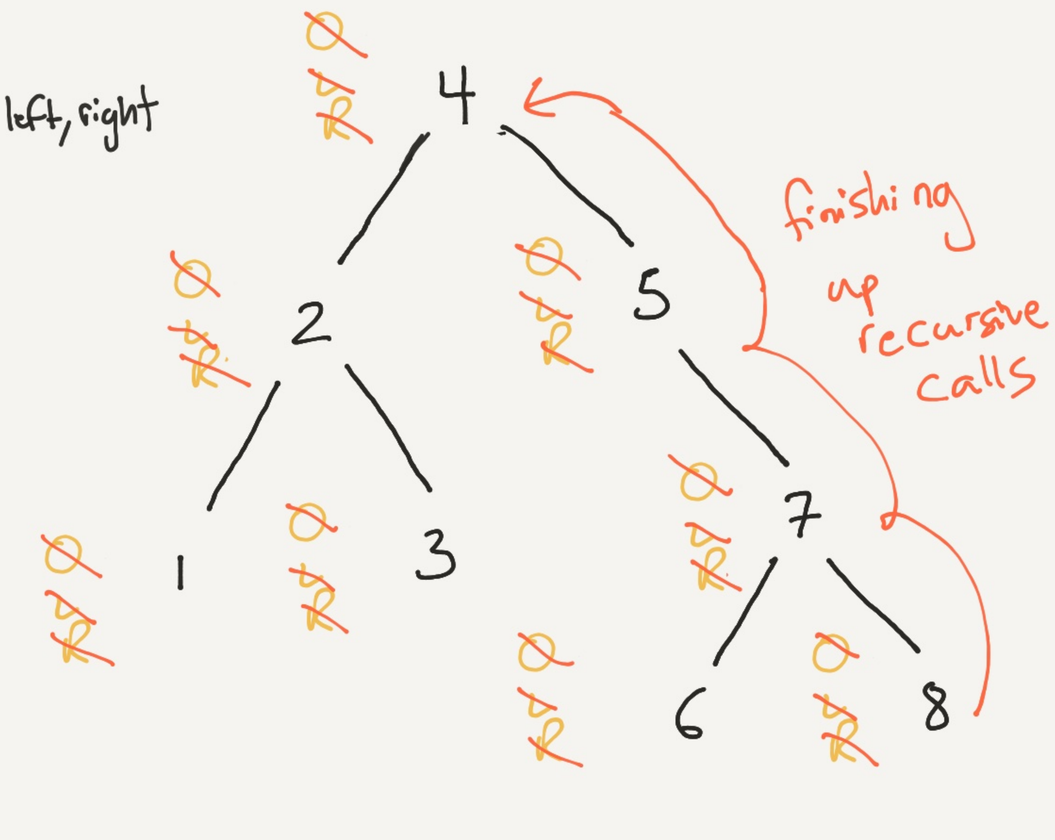
curr =

[4, 2, 5,

Binary Trees

Pre-Order DFS

```
TreeNode {
  int data
  TreeNode left, right
}
```



operate

left

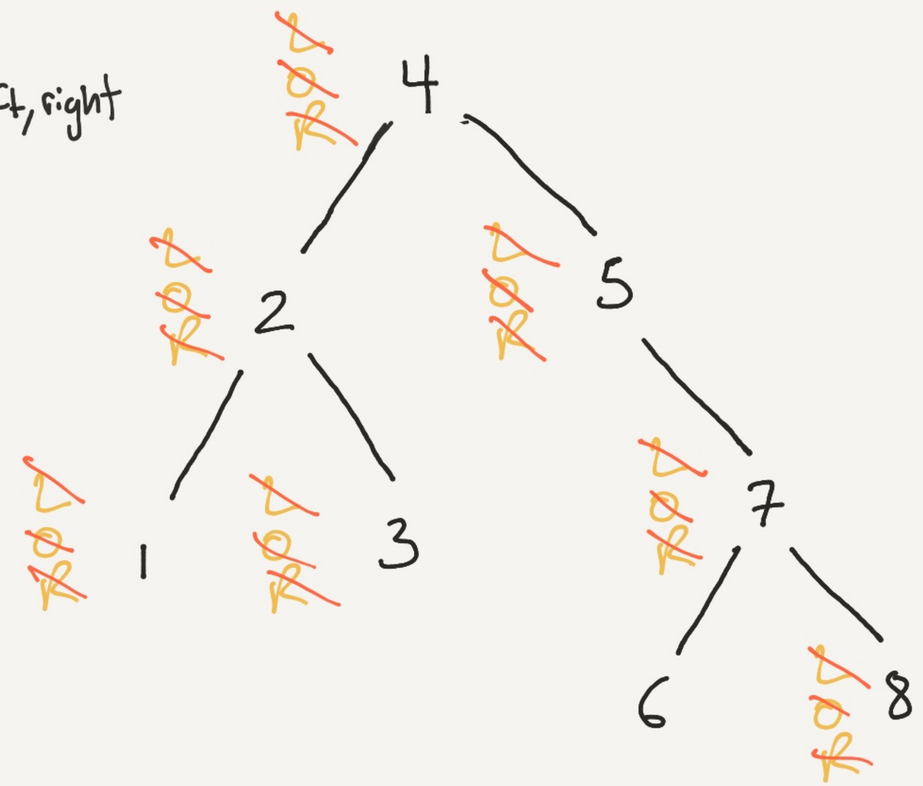
right

4, 2, 1, 3, 5, 7, 6, 8

Binary Trees

In-Order DFS

```
TreeNode {
  int data
  TreeNode left, right
}
```



Left Traverse

Operate

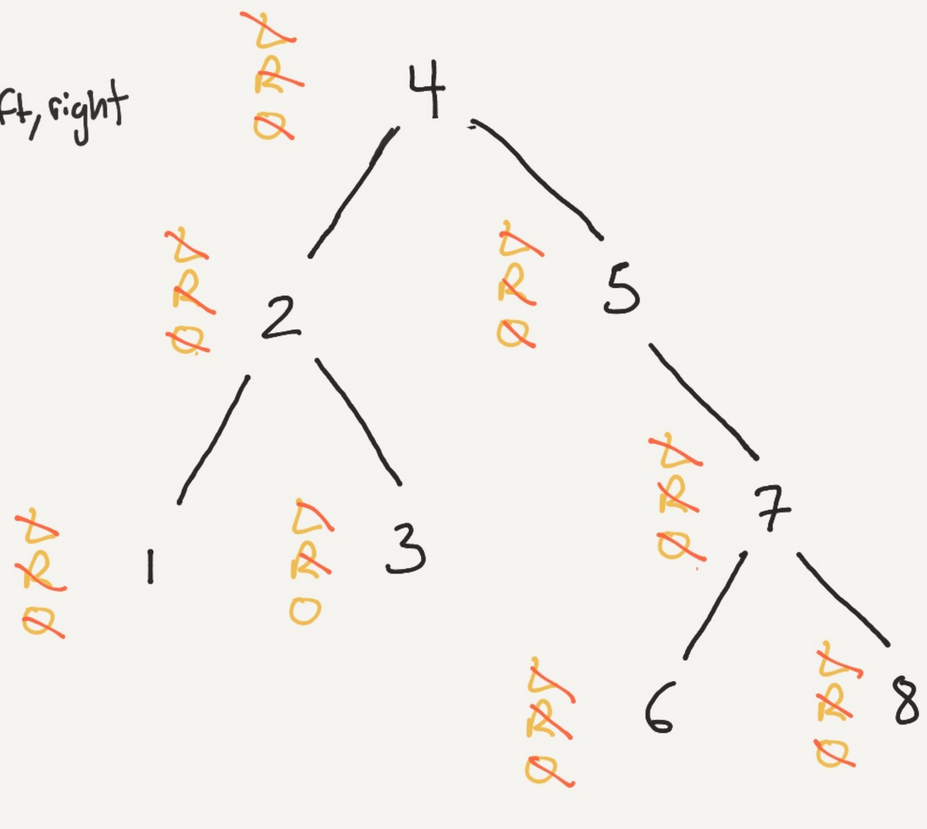
Right Traverse

1, 2, 3, 4, 5, 6, 7, 8

Binary Trees

Post Order DFS

```
TreeNode {
  int data
  TreeNode left, right
}
```



Left Traverse

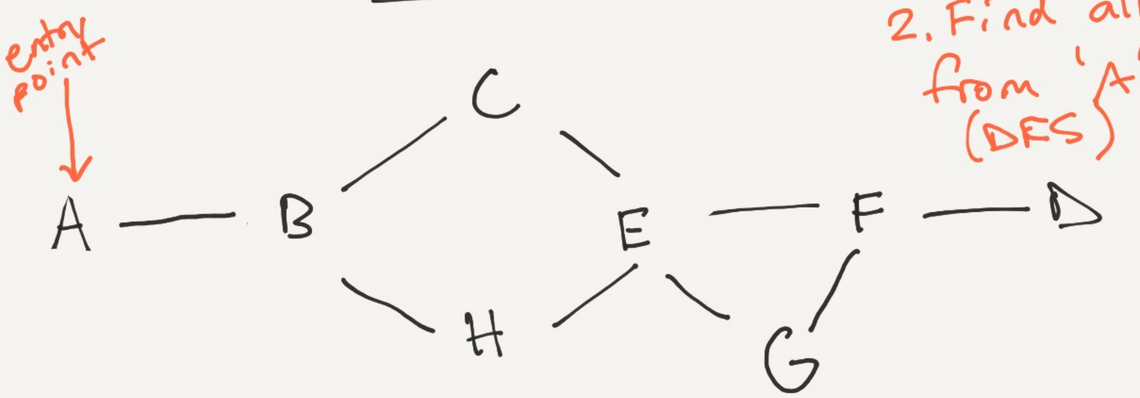
Right Traverse

Operate

1, 3, 2, 6, 8, 7, 5, 4

Graph Traversals

Problems
1. BFS
2. Find all paths from 'A' to 'D' (DFS)




```
TreeNode
    int value
    TreeNode left, right
```

```
Vertex {
    char id
    List<Vertex> edges
}
```

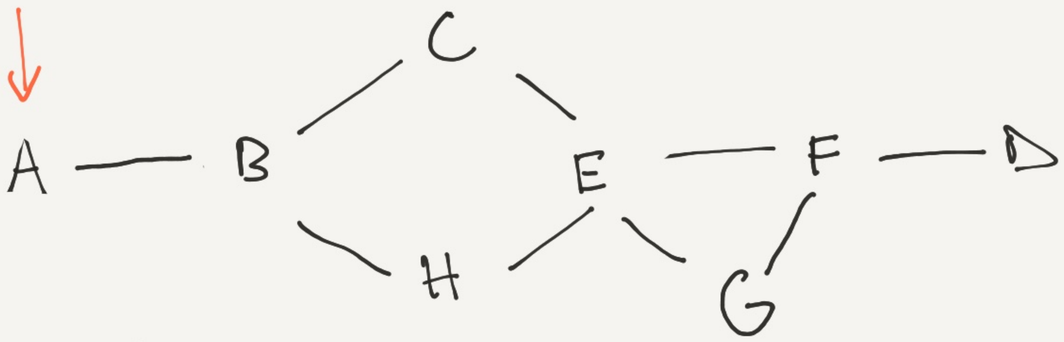
Undirected graph

ABCHEGFD
ABHCEGFD
} 2 valid BFS Traversals

A: B

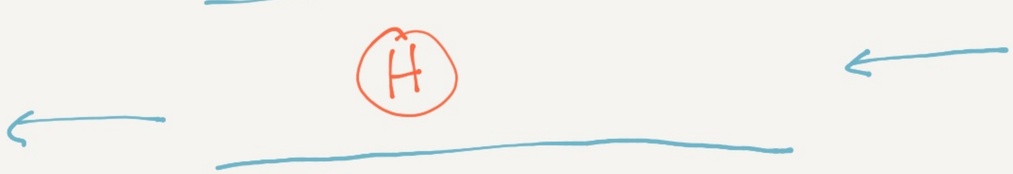
B: A, C, H

Graph BFS




```
Vertex {
   char id
   List<Vertex> neighbors
}
```

seen { A, B

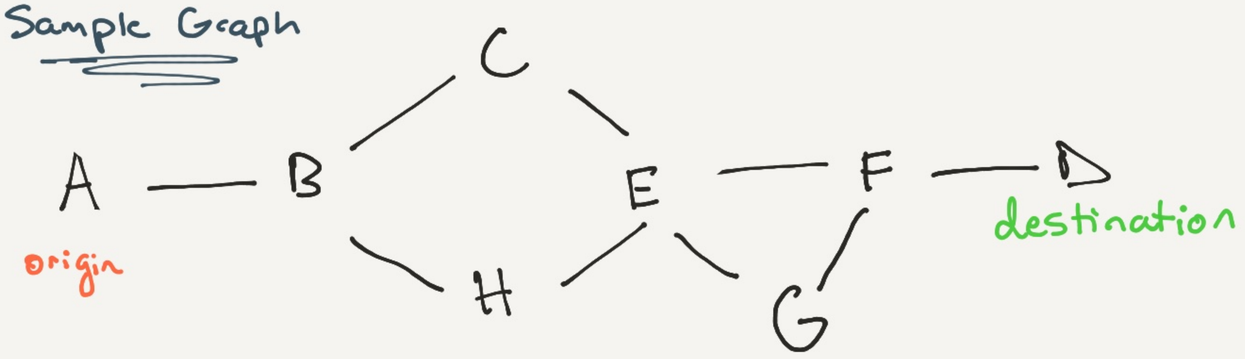


A B

Curr = C

Find all Paths in Undirected, Unweighted Graph From Origin to Destination

Sample Graph



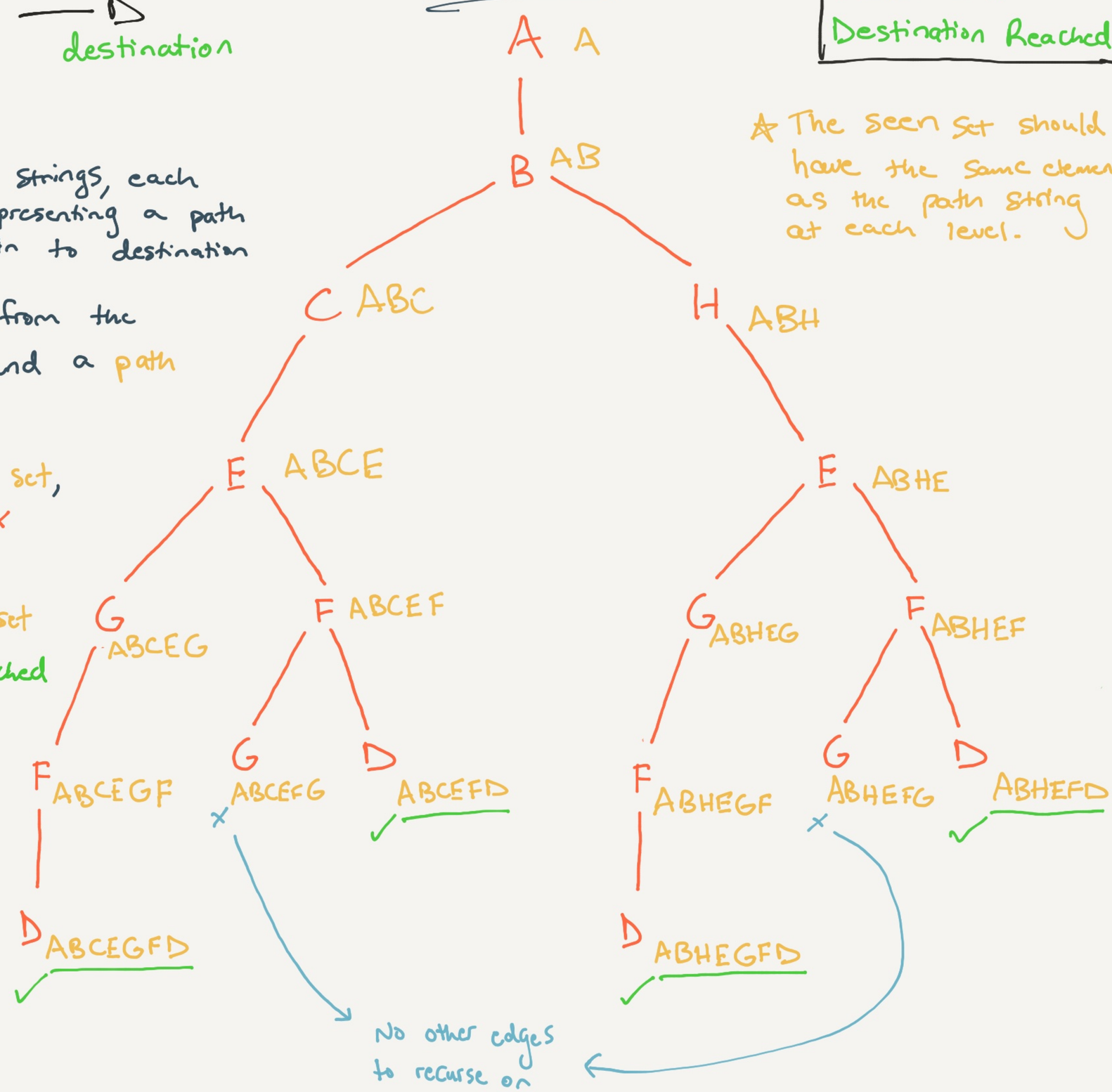
Input: origin vertex

Output: list of strings, each string representing a path from origin to destination

Recursive Approach: Perform a DFS from the origin vertex using a seen set and a path string

1) Make 1st recursive call with empty seen set, empty path string, and origin vertex
2) At start of recursive function add vertex id to path string and seen set
3.1) Check base case for destination reached
   Add current path string to result list
3.2) Else, loop over all edges of current vertex
   if edge id not in seen set, recurse on edge
4) Remove current vertex id from seen set and exit function

Recursive DFS Traversal Diagram



Path/Seen to this point
Dead End
Destination Reached

* The seen set should have the same elements as the path string at each level.