```
def DFS(g, u, discovered):
    """
    Perform DFS on undiscovered portion of graph G starting at vertex u.
    discovered is a dict mapping each vertex to the edge used to discover it during DFS
    Newly discovered vertices will be added to the dict as a result
    """
    for e in g.incident_edges(u):          # for every outgoing edge from u
      v = e.opposite(u)
      if v not in discovered:
        discovered[v] = e                  # e is tree edge which discovered v
        DFS(g, v, discovered)              # recursively traverse from v
```

Reconstruction of a path from u → v

→ we can use the basic DFS function as a tool to identify the directed path leading from vertex u to v, if v is reachable from u.

- → begin at end of path examining discovery dict. to determine the edge used to reach vertex v and the other endpoint of that edge
- → add that vertex to a list and repeat process to determine edge used to discover it
- → once we have traced the path all the way to start vertex u, we can reverse the list so it is properly oriented from u to v, and return it to caller

```
def construct_path(u, v, discovered):
    path = []
    if v in discovered:
      path.append(v)
      walk = v
      while walk is not u:
        e = discovered[walk]               # find edge leading to walk
        parent = e.opposite(walk)
        path.append(parent)
        walk = parent
      path.reverse()                       # reorient path from u→v
    return path
```