Lecture 4: Reachability in undirected graphs

What parts of a graph are reachable from a given vertex?

With an adjacency list representation, this is like navigating a maze...

<table>
<thead>
<tr>
<th>Potential difficulty</th>
<th>Don’t go round in circles</th>
<th>Don’t miss anything</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classical solution</td>
<td>Piece of chalk to mark visited junctions</td>
<td>Ball of string – leads back to starting point</td>
</tr>
<tr>
<td>Cyber-analog</td>
<td>Boolean variable for each vertex: visited or not</td>
<td>STACK</td>
</tr>
</tbody>
</table>

An exploration procedure

procedure explore(G,v)

input: graph G = (V,E); node v in V
output: visited[u] is set to true for all u reachable from v

visited[v] = true
for each edge (v,u) in E:
    if not visited[u]:
        explore(G,u)

explore(G,a):
**Does “explore” work?**

```plaintext
procedure explore(G,v)
    visited[v] = true
    for each edge (v,u) in E:
        if not visited[u]:
            explore(G,u)
```

Does it actually halt?

For any node u, explore(G,u) is called at most once; thereafter visited[u] is set.

Does it visit everything reachable from v?

Suppose it misses node u reachable from v; we'll derive a contradiction.

Pick any path from v to u, and let z be the last node on the path that was visited.

```
  V ---- Z ---- W ---- U
```

But w would not have been overlooked during explore(G,z); this is a contradiction.

**Alternative proof**

```plaintext
procedure explore(G,v)
    visited[v] = true
    for each edge (v,u) in E:
        if not visited[u]:
            explore(G,u)
```

Does explore(G,v) visit everything reachable from v?

Do a proof by induction.