Department of
Computer Engineering

Data Structure & Algorithms

Graph Search

Searching a graph

- Given: a graph G = (V, E), directed or undirected
- Goal: *methodically* explore every vertex (and every edge)
- Different methods of exploration =>
  - Different order of vertex discovery
  - Different shape of the exploration trace subgraph (which can be a spanning tree of the graph)
- Methods of exploration:
  - Breadth-First Search
  - Depth-First Search

Breadth-First Search

Breadth first search

- Given
  - a graph G=(V,E) – set of vertices and edges
  - a distinguished source vertex s
- Breadth first search systematically explores the edges of G to discover every vertex that is reachable from s.
- It also produces a ‘breadth first tree’ with roots that contains all the vertices reachable from s.
- For any vertex v reachable from s, the path in the breadth first tree corresponds to the shortest path in graph G from s to v.
- It works on both directed and undirected graphs.

Breadth first search - concepts

- To keep track of progress, it colors each vertex - white, gray or black.
- All vertices start white.
- A vertex discovered first time during the search becomes nonwhite.
- All vertices adjacent to black ones are discovered. Whereas, gray ones may have some white adjacent vertices.
- Gray represent the frontier between discovered and undiscovered vertices.

BFS – How it produces a Breadth first tree

- The tree initially contains only root: s
- Whenever a vertex v is discovered in scanning adjacency list of vertex u
  - Vertex v and edge (u,v) are added to the tree.

Data Structure & Algorithms Fall 2021

BFS - algorithm

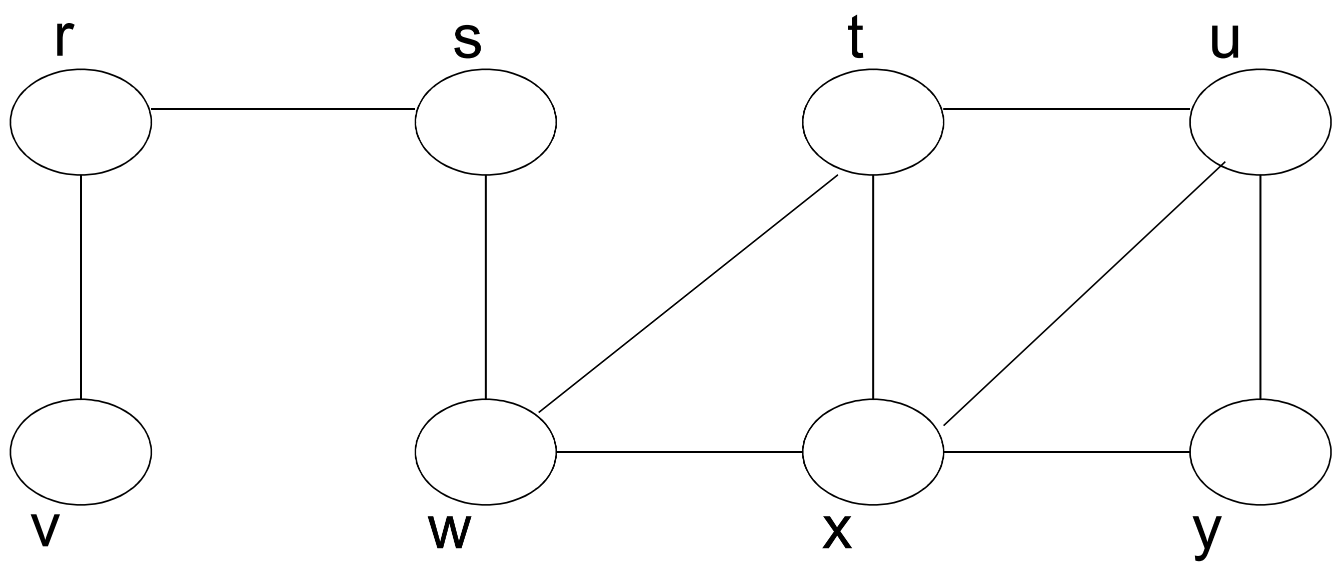
```
BFS(G, s)                    // G is the graph and s is the starting node
 1  for each vertex u ∈ V [G] - {s}
 2       do color[u] ← WHITE       // color of vertex u
 3          d[u] ← ∞               // distance from source s to vertex u
 4          π[u] ← NIL             // predecessor of u
 5  color[s] ← GRAY
 6  d[s] ← 0
 7  π[s] ← NIL
 8  Q ← Ø                          // Q is a FIFO - queue
 9  ENQUEUE(Q, s)
10  while Q ≠ Ø              // iterates as long as there are gray vertices. Lines 10-18
11      do u ← DEQUEUE(Q)
12         for each v ∈ Adj[u]
13             do if color[v] = WHITE       // discover the undiscovered adjacent
vertices
14                   then color[v] ← GRAY   // enqueued whenever painted gray
15                        d[v] ← d[u] + 1
16                        π[v] ← u
17                        ENQUEUE(Q, v)
18         color[u] ← BLACK         // painted black whenever dequeued
```

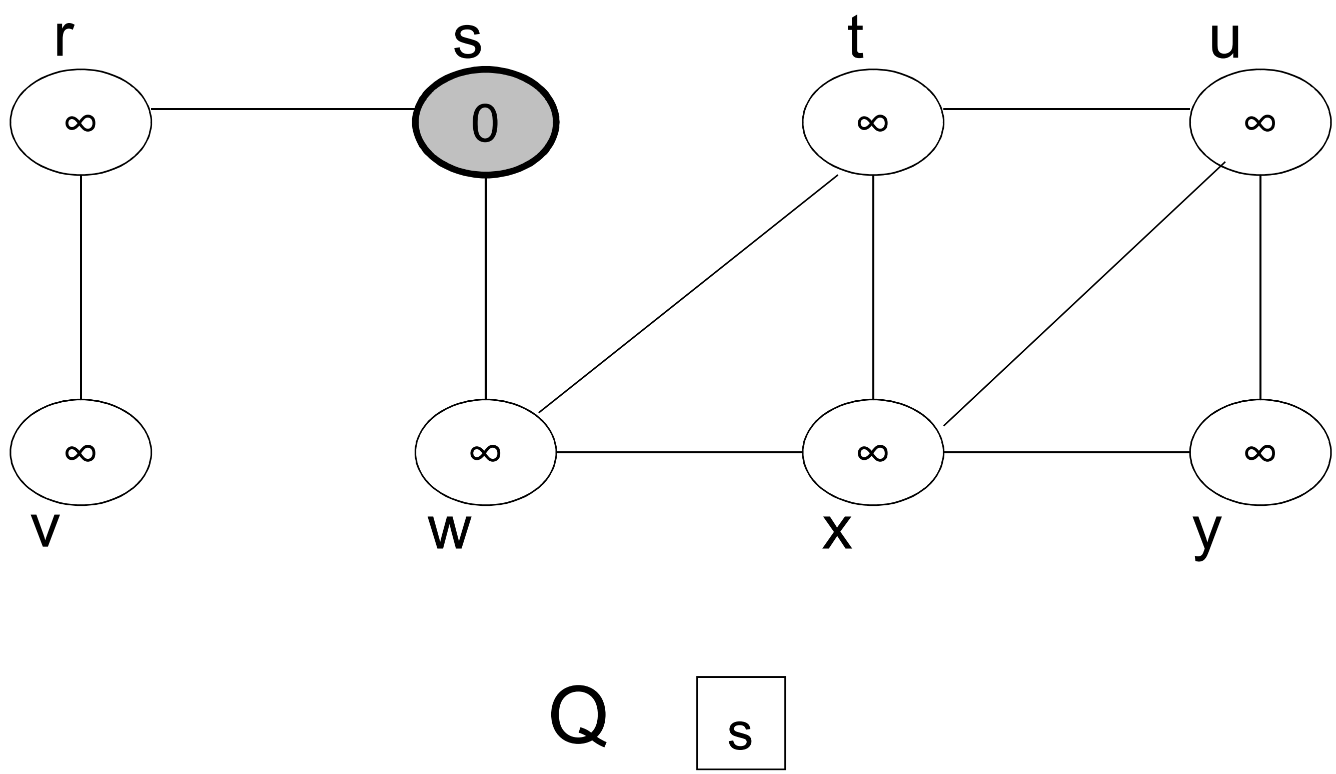
Breadth first search - analysis

- Enqueue and Dequeue happen only once for each node – O(V).
- Sum of the lengths of adjacency lists – θ(E) (for a directed graph)
- Initialization overhead O(V)

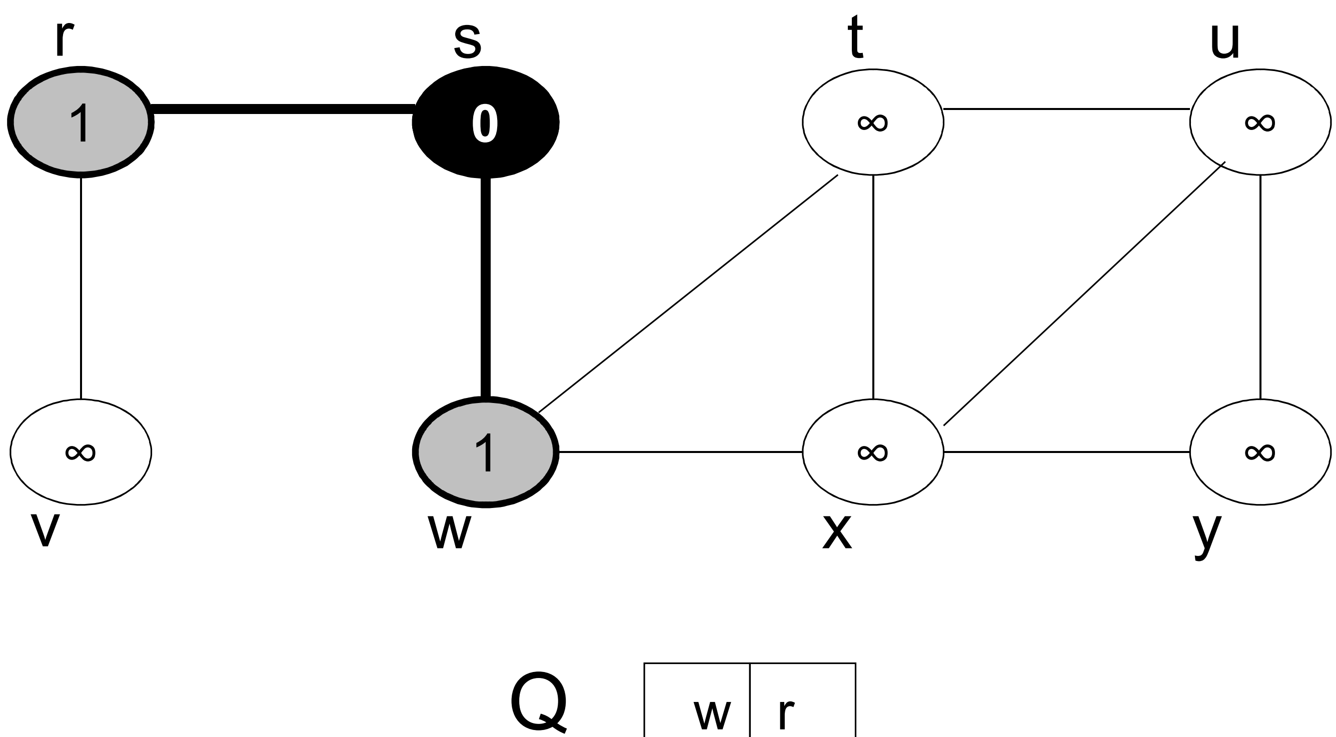
Total runtime O(V+E)

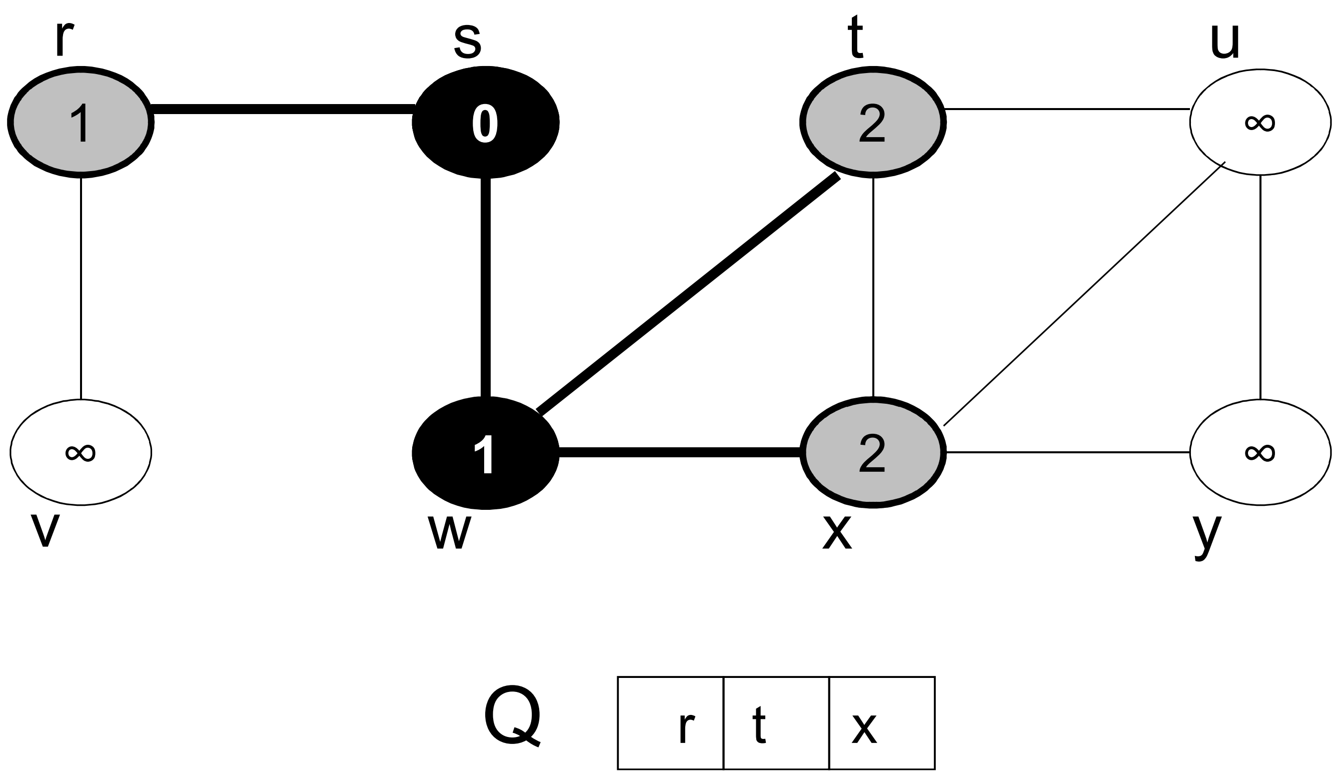
Example – Applying BFS

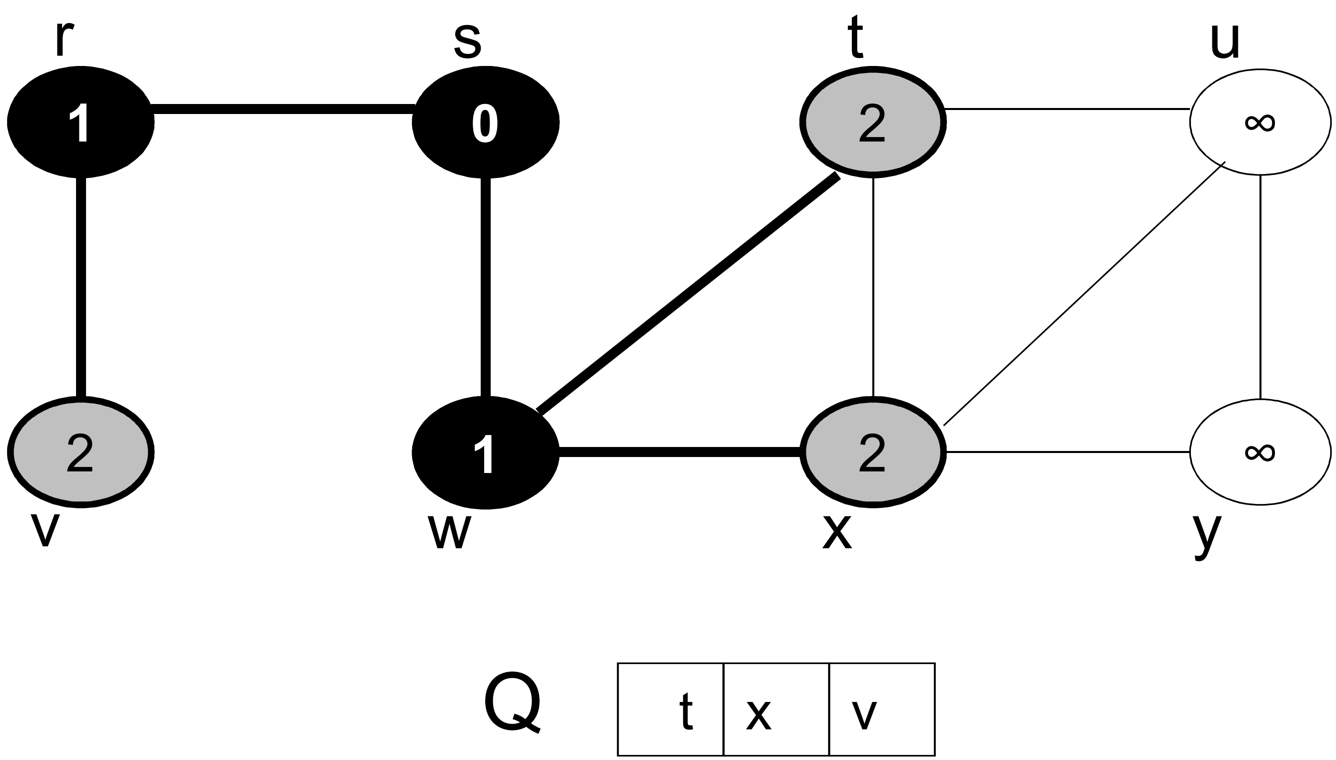


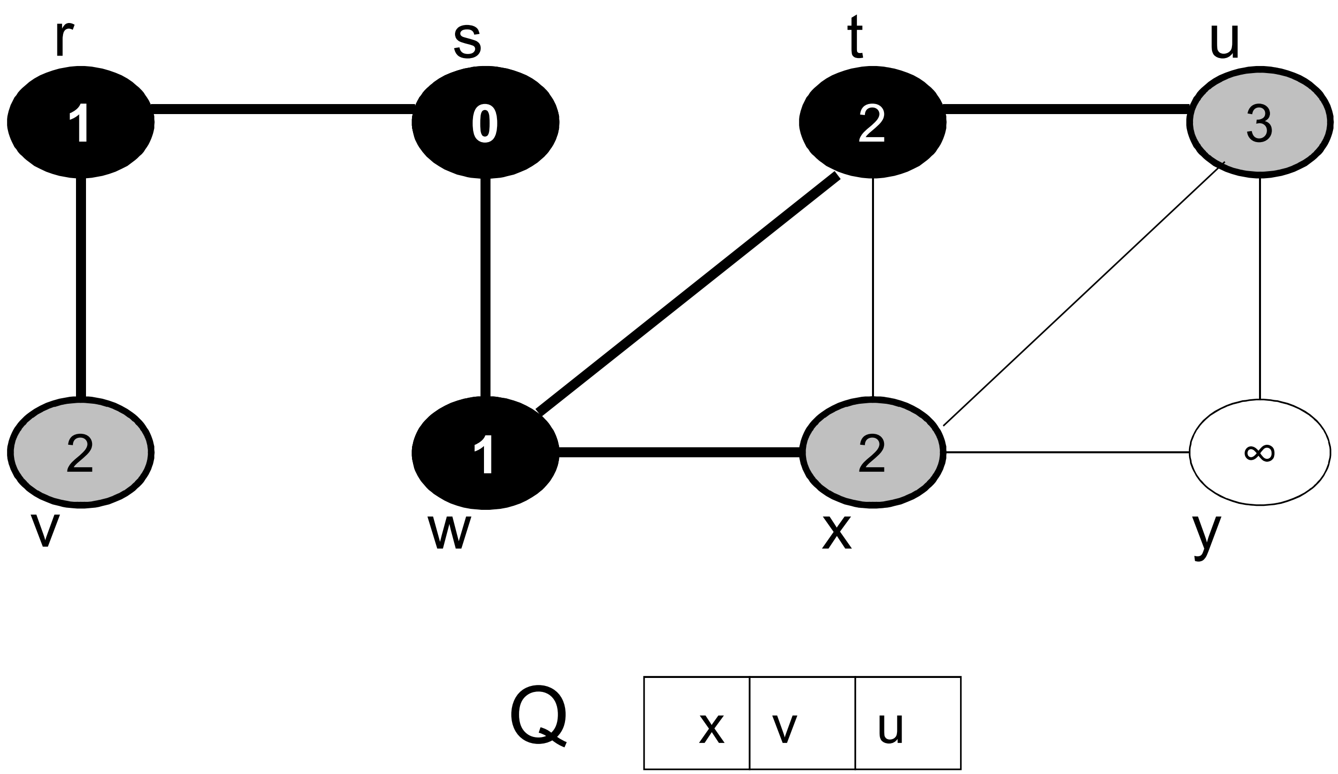


r
s
t
u
∞
0
∞
∞
∞
∞
∞
∞
v
w
x
y
Q
s

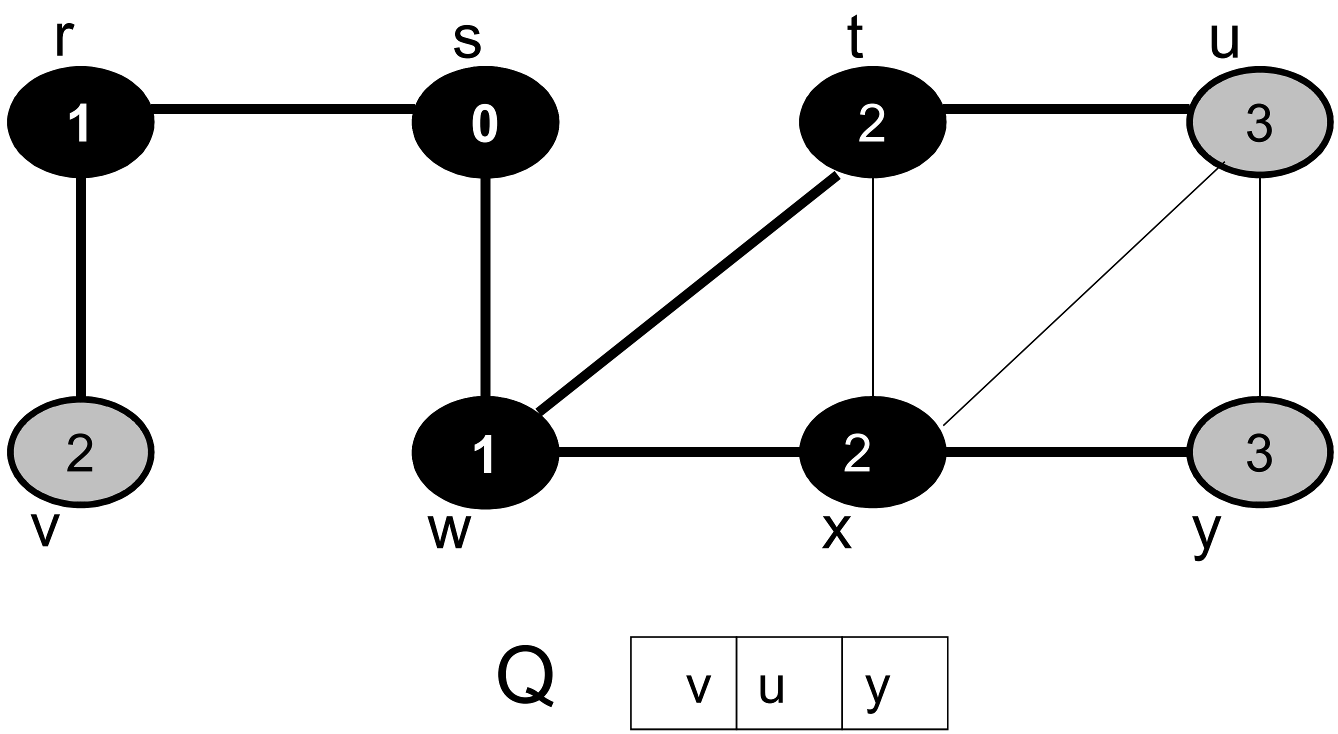

r
s
t
u
1
0
∞
∞
∞
1
∞
∞
v
w
x
y
Q
w
r

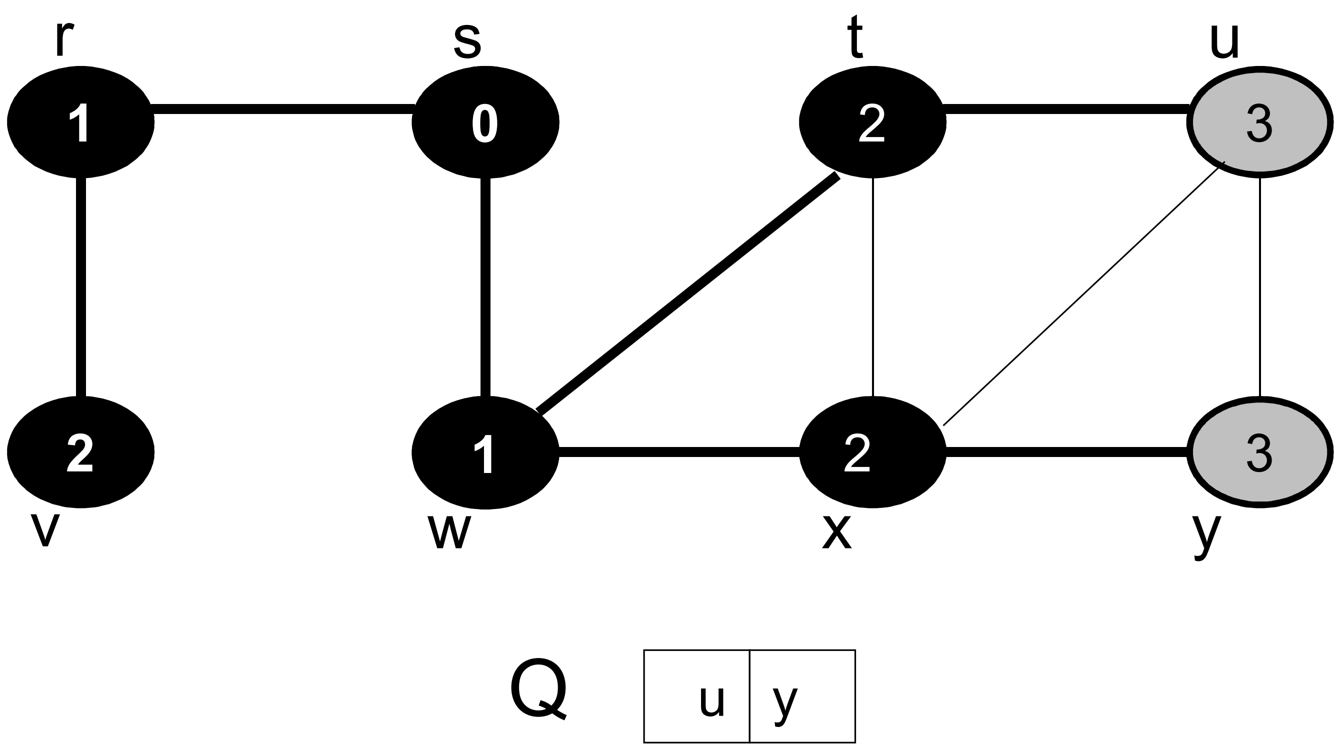

r
s
t
u
1
0
2
∞
∞
1
2
∞
v
w
x
y
Q
r
t
x

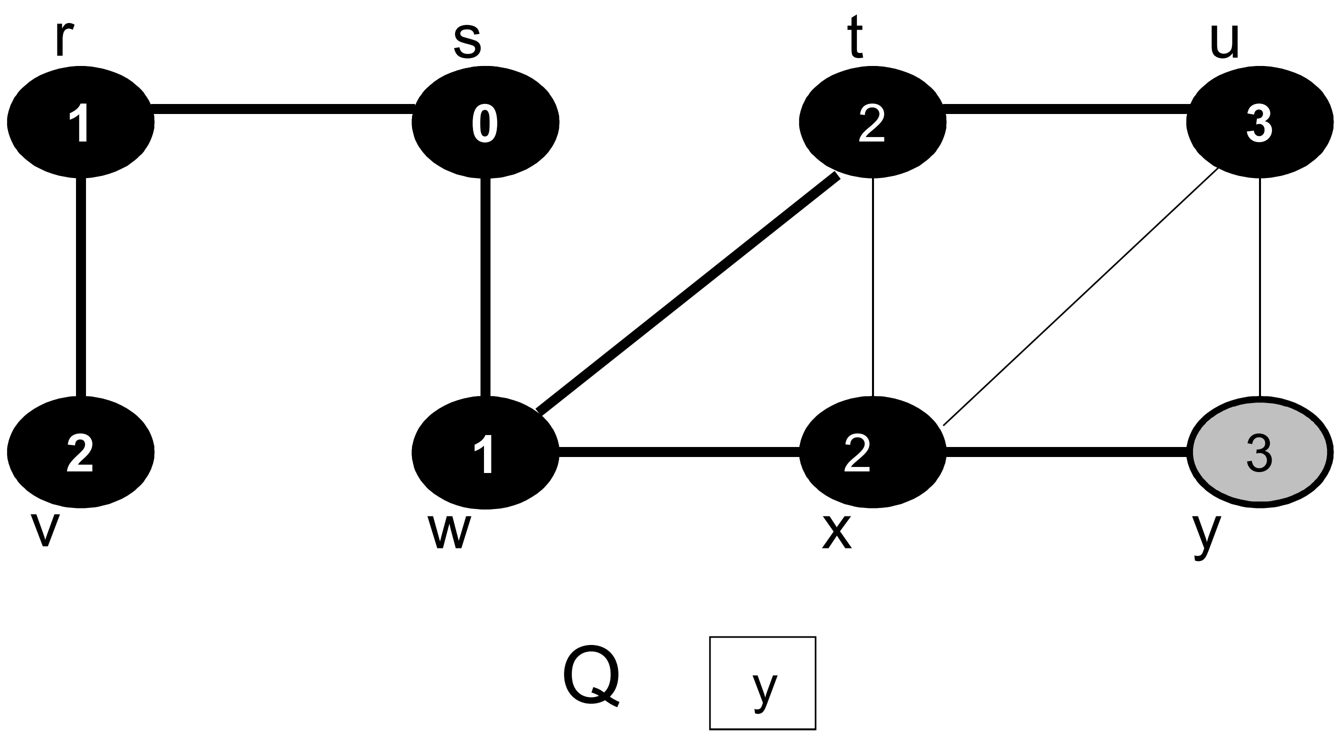

r
s
t
u
1
0
2
∞
2
1
2
∞
v
w
x
y
Q
t
x
v

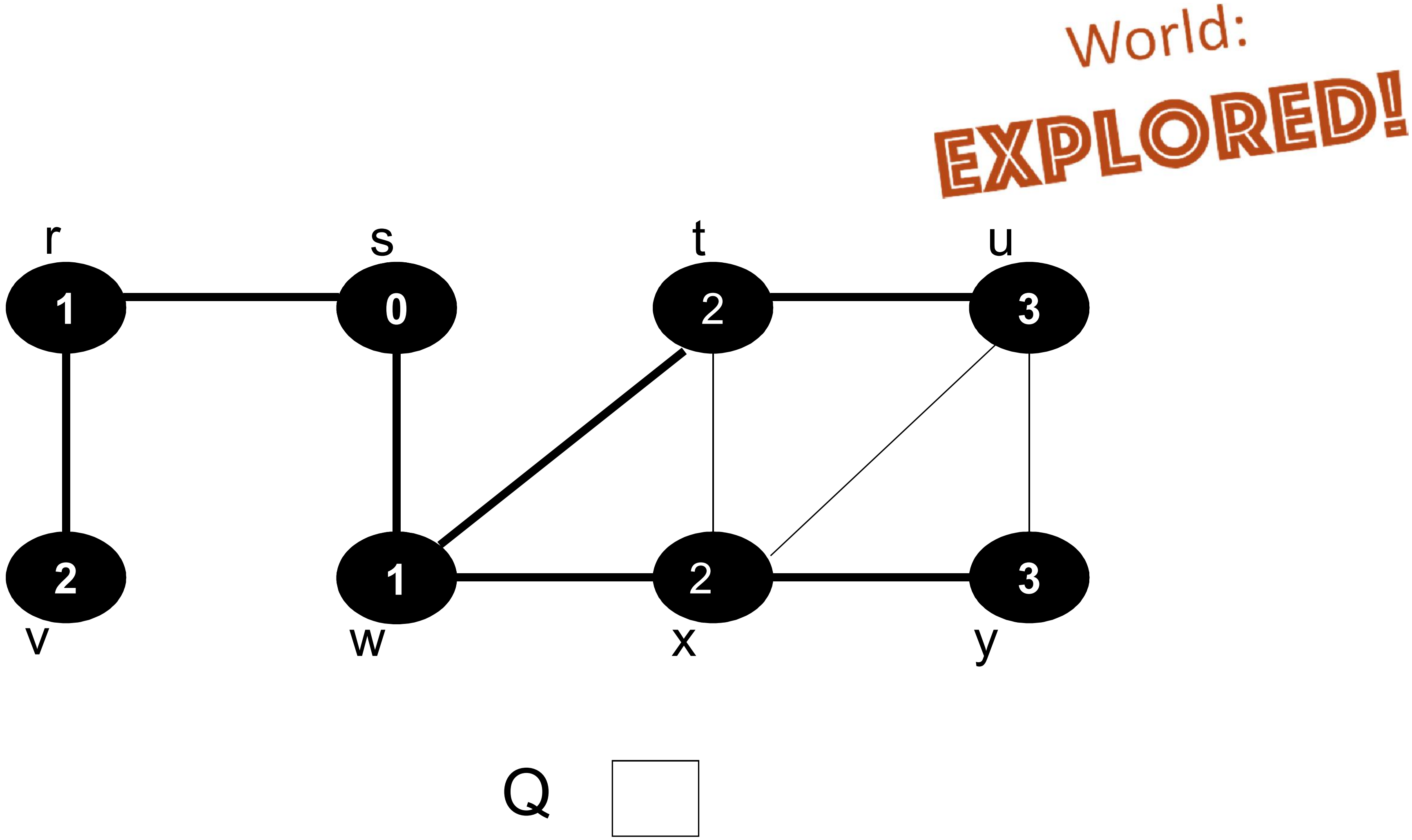


r
s
t
u
1
0
2
3
2
1
2
∞
v
w
x
y
Q
x
v
u


r
s
t
u
1
0
2
3
2
1
2
3
v
w
x
y
Q
v
u
y


r
s
t
u
1
0
2
3
2
1
2
3
v
w
x
y
Q
u
y


r
s
t
u
1
0
2
3
2
1
2
3
v
w
x
y
Q
y


World:
EXPLORED!
r
s
t
u
1
0
2
3
2
1
2
3
v
w
x
y
Q

Depth-First Search

Depth first search

- It searches ‘deeper’ the graph when possible.
- Starts at the selected node and explores as far as possible along each branch before backtracking.
- Vertices go through white, gray and black stages of color.
  - White – initially
  - Gray – when discovered first
  - Black – when finished i.e. the adjacency list of the vertex is completely examined.
- Also records timestamps for each vertex
  - d[v] when the vertex is first discovered
  - f[v] when the vertex is finished

Depth first search - algorithm

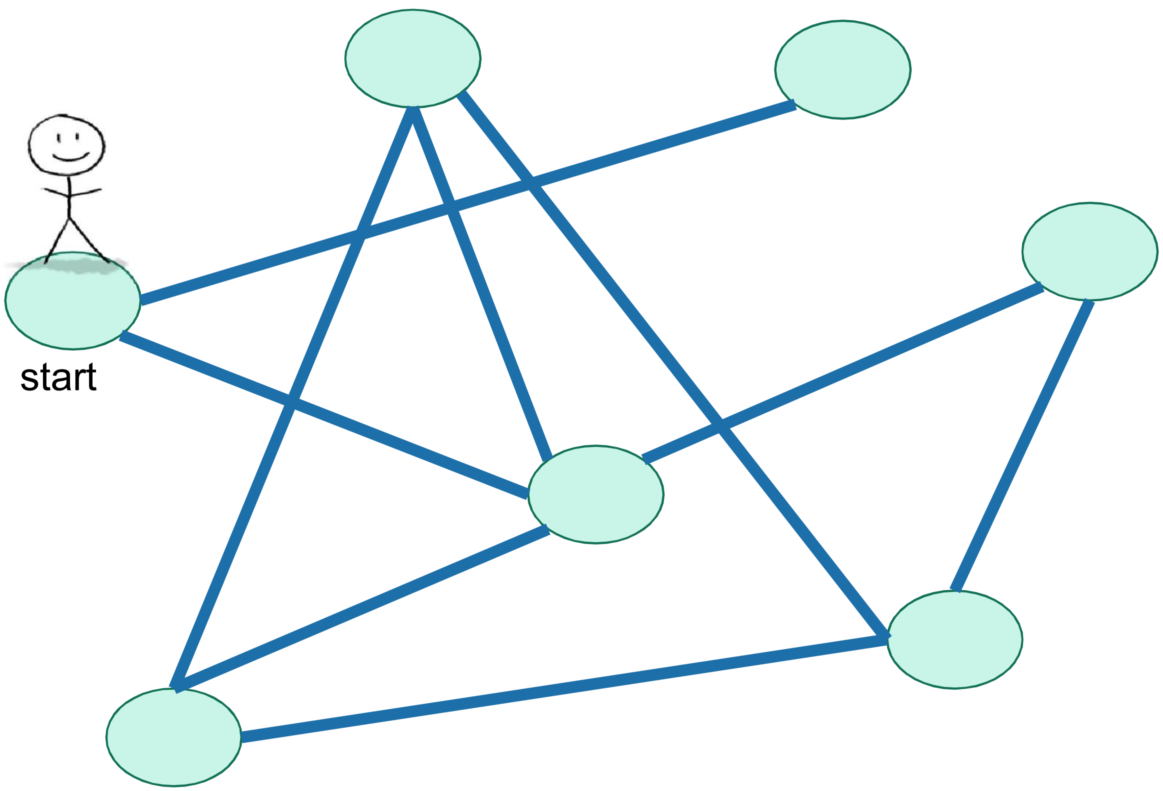
```
DFS(G)
for each vertex u ∈ V [G]
     do color[u] ← WHITE          // color all vertices white, set their parents NIL
        π[u] ← NIL
time ← 0                          // zero out time
for each vertex u ∈ V [G]                  // call only for unexplored vertices
     do if color[u] = WHITE                // this may result in multiple sources
           then DFS-VISIT(u)

DFS-VISIT(u)
color[u] ← GRAY      ▹White vertex u has just been discovered.
time ← time +1
d[u] time                 // record the discovery time
for each v ∈ Adj[u]       ▹Explore edge(u, v).
     do if color[v] = WHITE
           then π[v] ← u // set the parent value
                       DFS-VISIT(v)        // recursive call
color[u] BLACK       ▹ Blacken u; it is finished.
f [u] ▹ time ← time +1
```

Depth first search - analysis

- Lines 1-3, initialization take time $\Theta(V)$.
- Lines 5-7 take time $\Theta(V)$, excluding the time to call the DFS-VISIT.
- DFS-VISIT is called only once for each node (since it's called only for white nodes and the first step in it is to paint the node gray).
- Loop on line 4-7 is executed $|Adj(v)|$ times. Since, $\sum_{v \in V} |Adj(v)| = \Theta(E)$, the total cost of DFS-VISIT it $\theta(E)$

The total cost of DFS is $\theta(V+E)$

Depth First Search

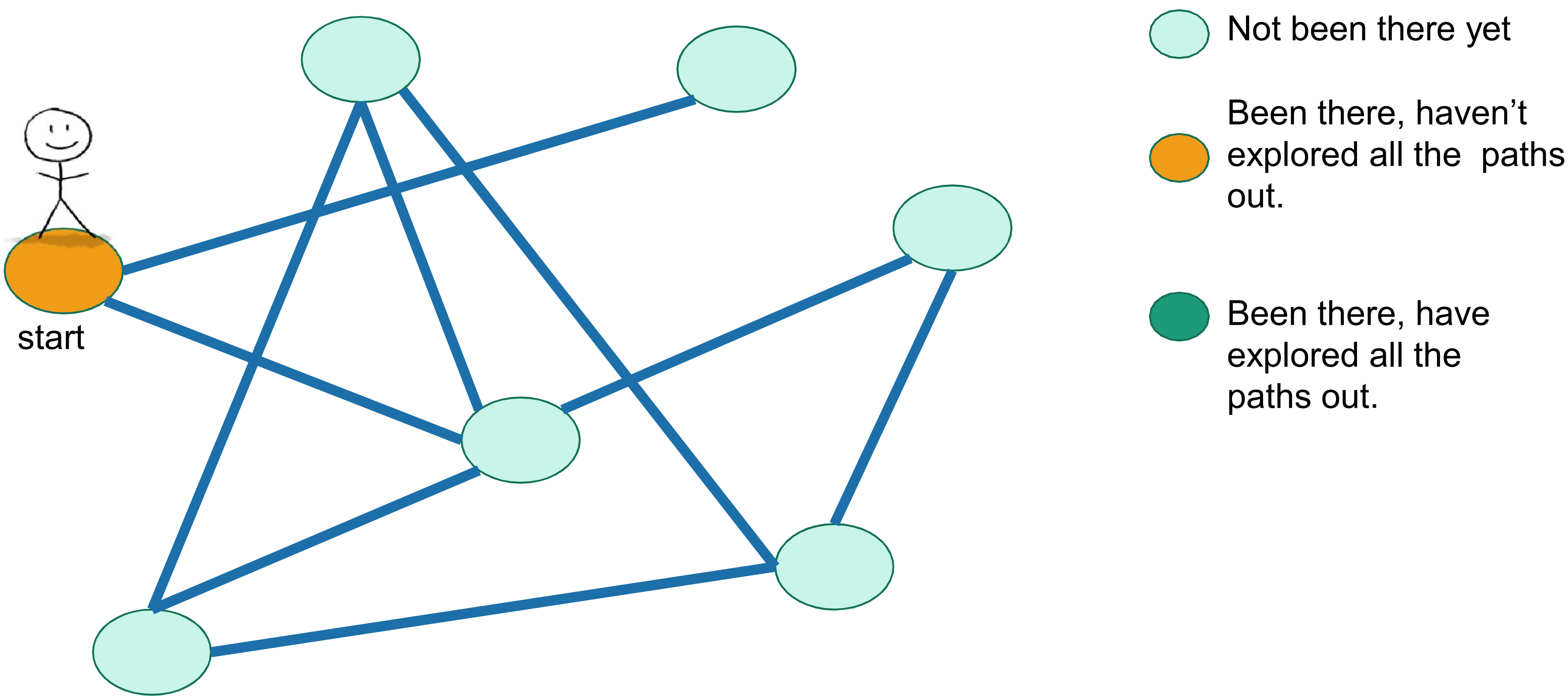


Not been there yet

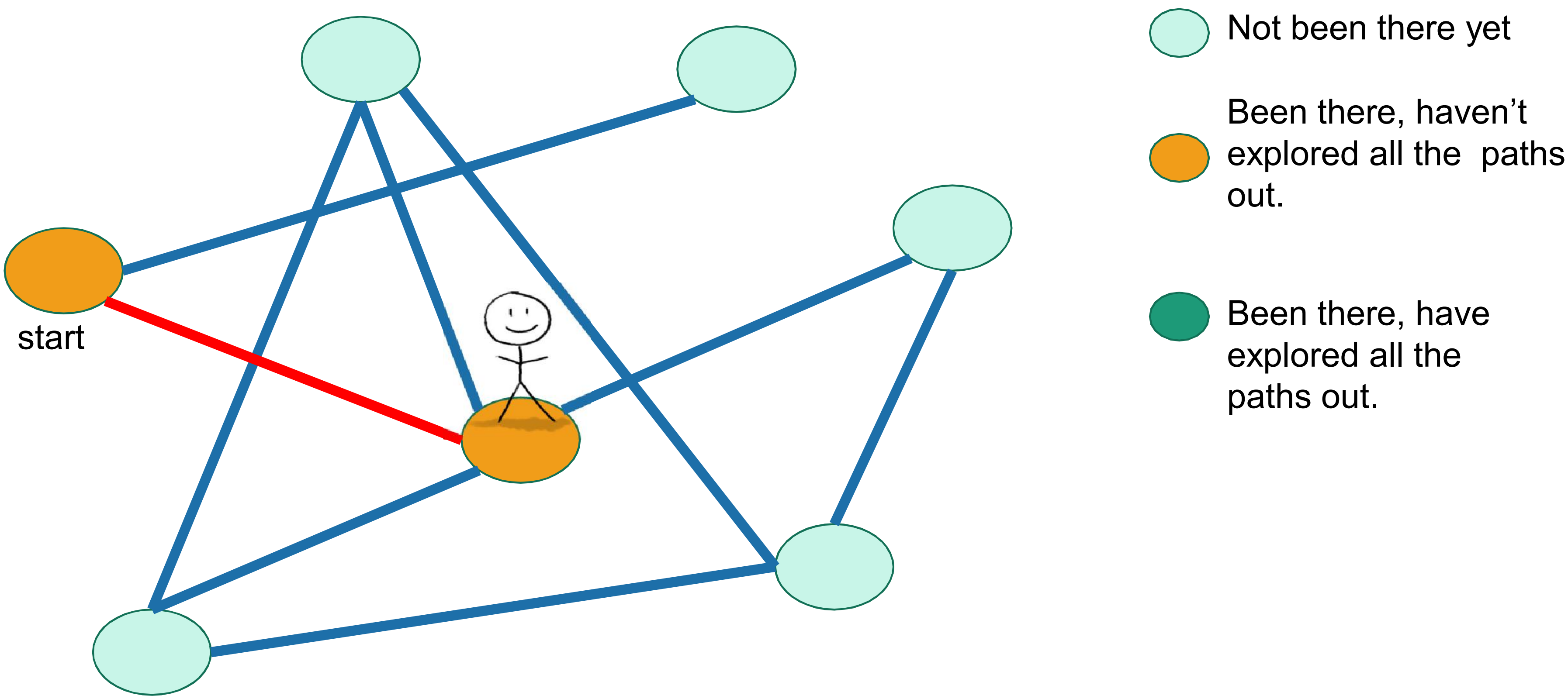
Been there, haven't explored all the paths out.

Been there, have explored all the paths out.

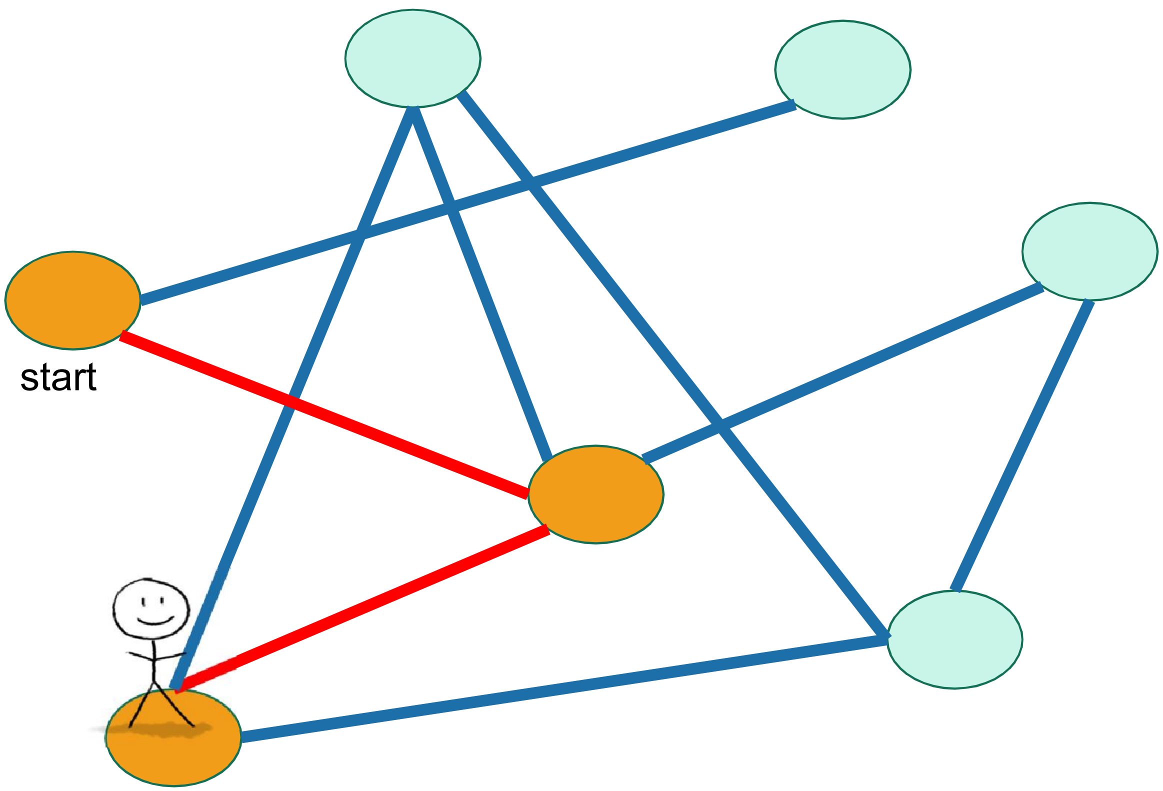
Depth First Search

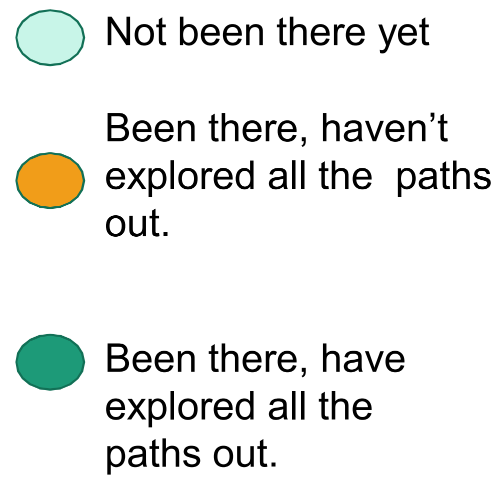


Depth First Search

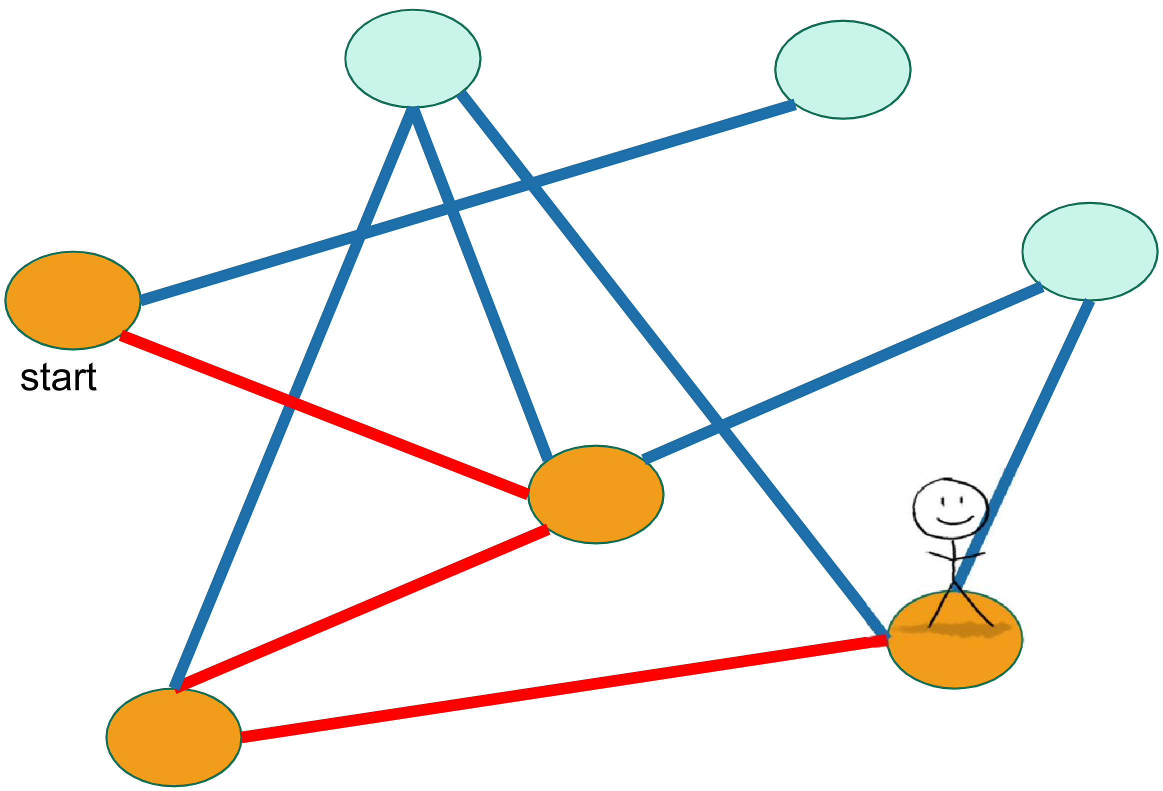


Depth First Search





Depth First Search

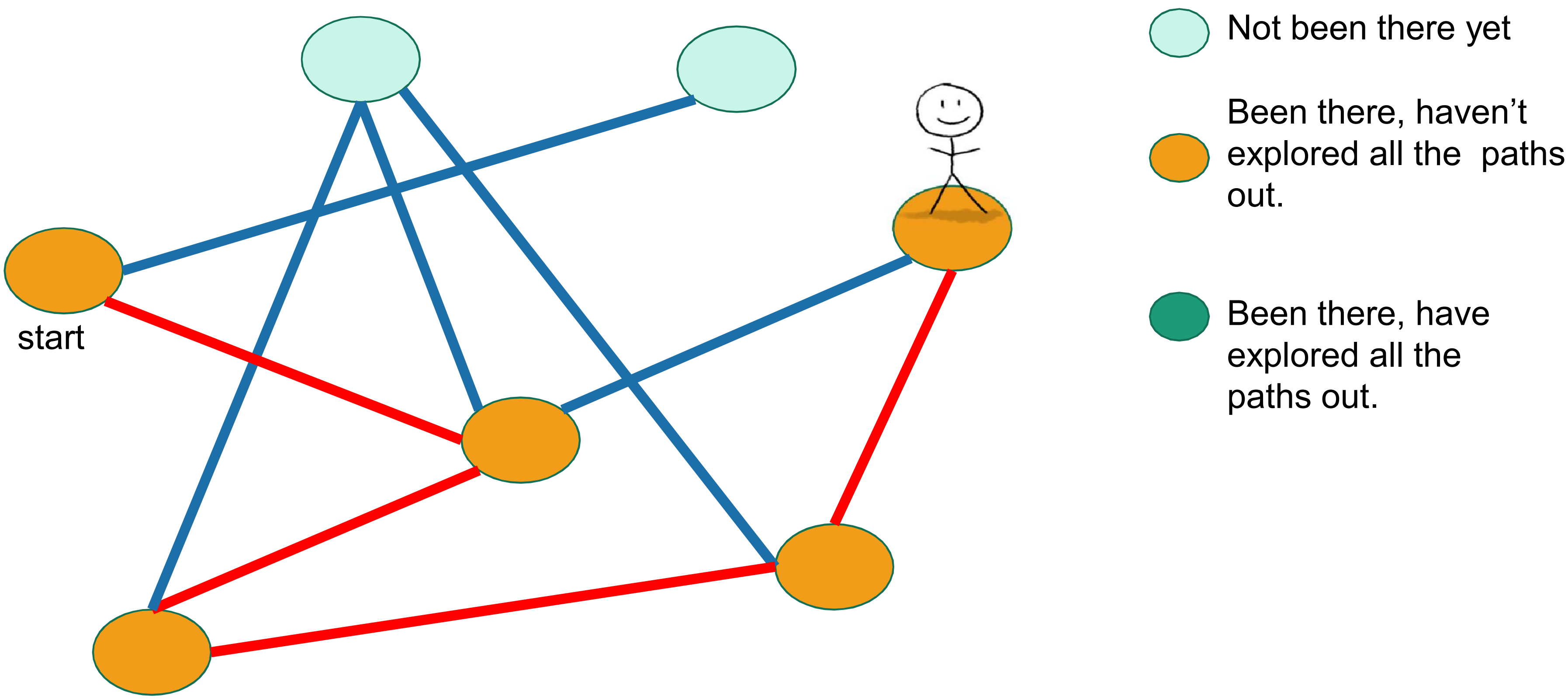


Not been there yet

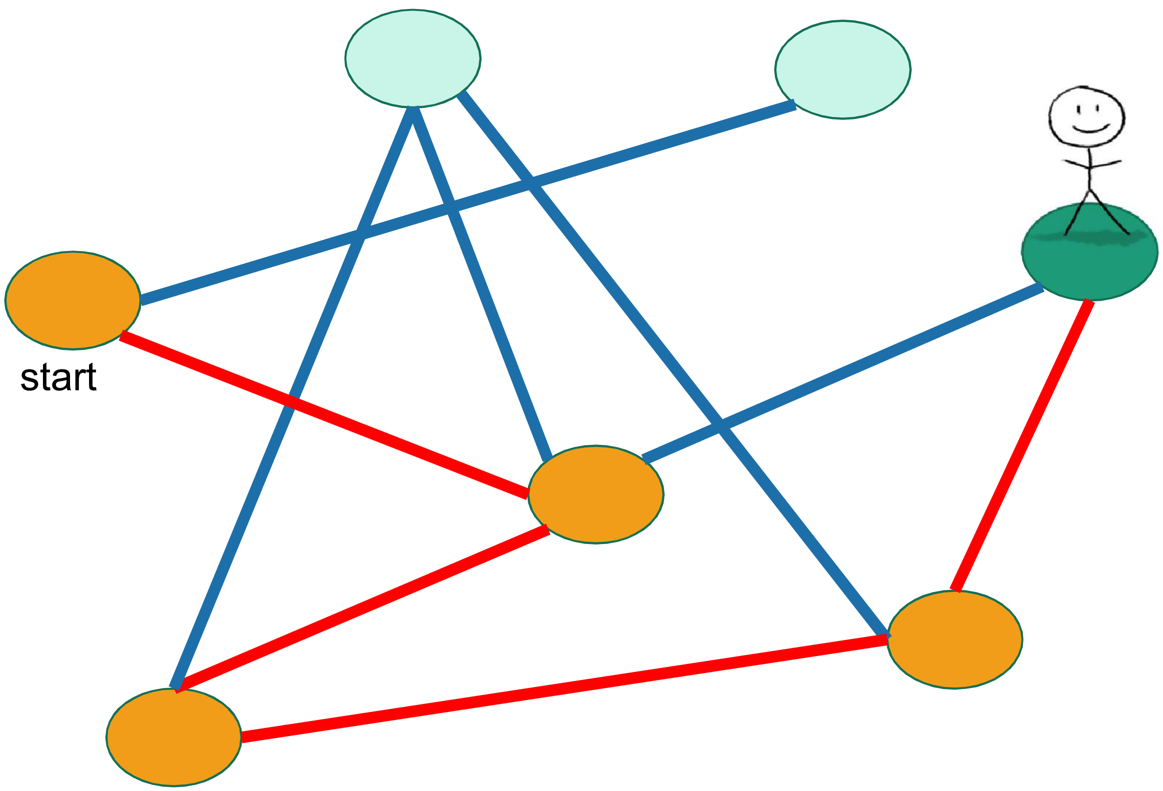
Been there, haven't explored all the paths out.

Been there, have explored all the paths out.

Depth First Search



Depth First Search

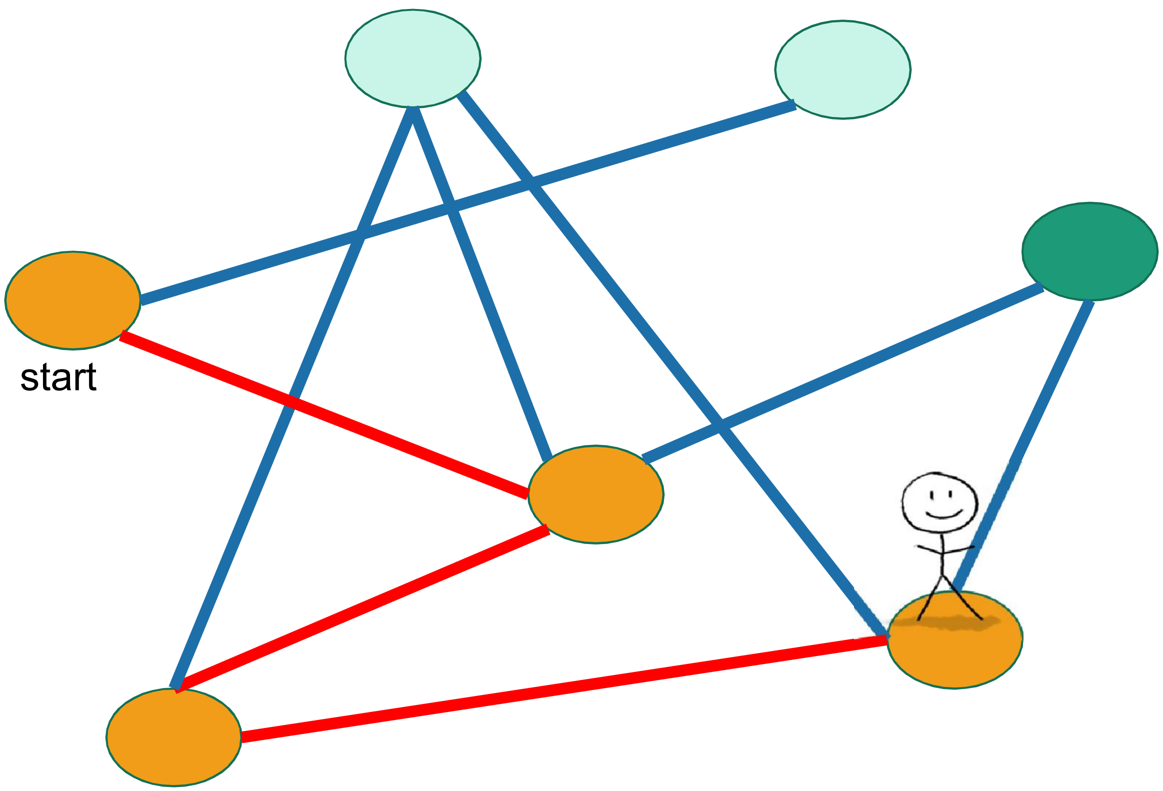


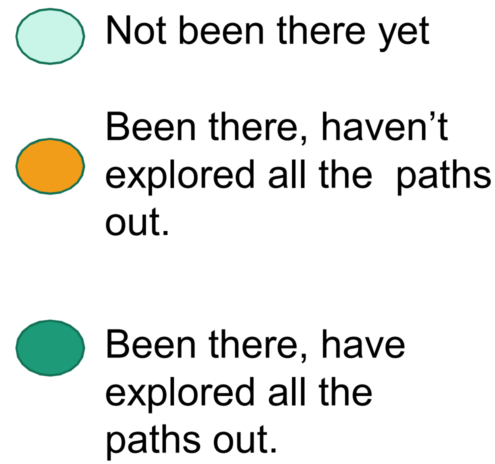
Not been there yet

Been there, haven't explored all the paths out.

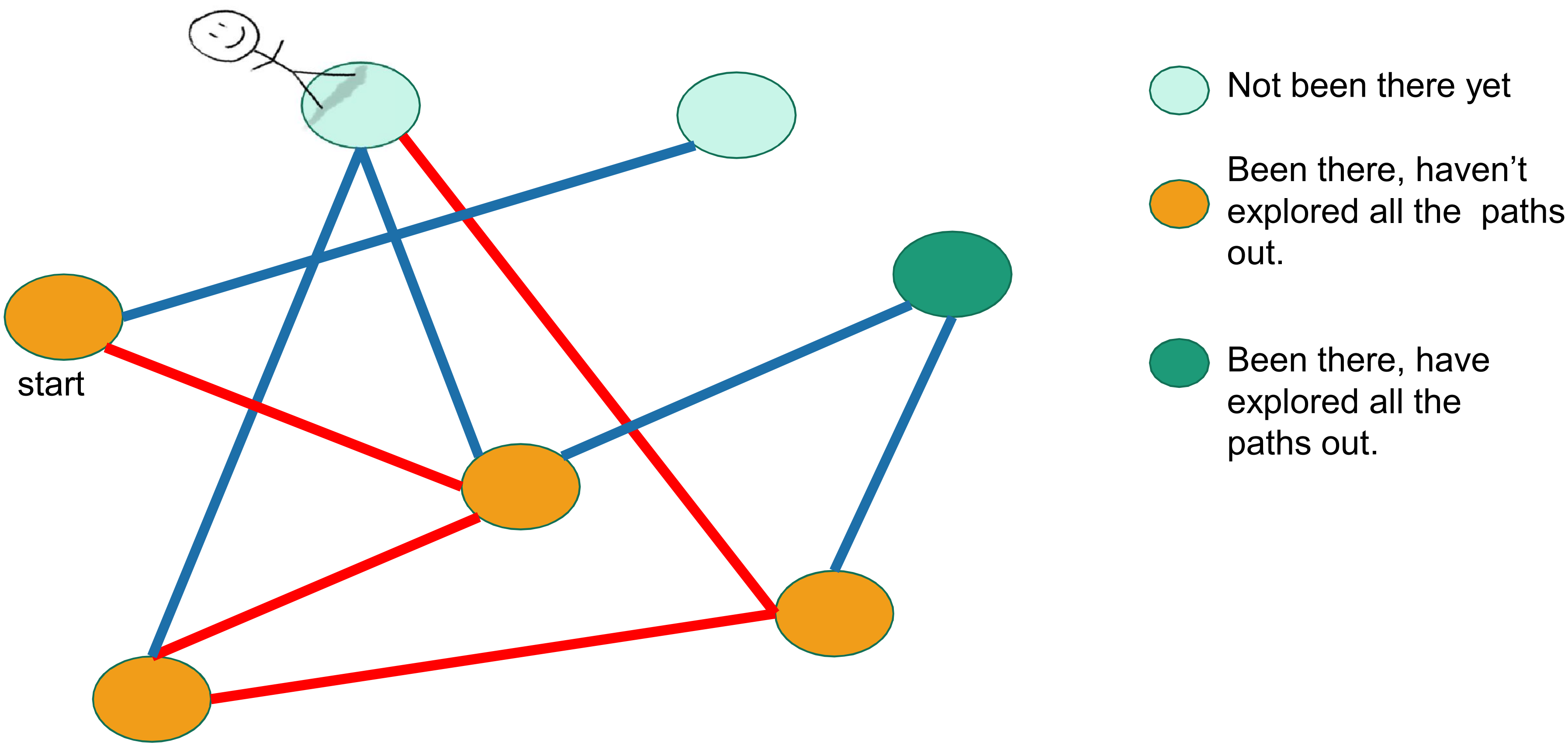
Been there, have explored all the paths out.

Depth First Search

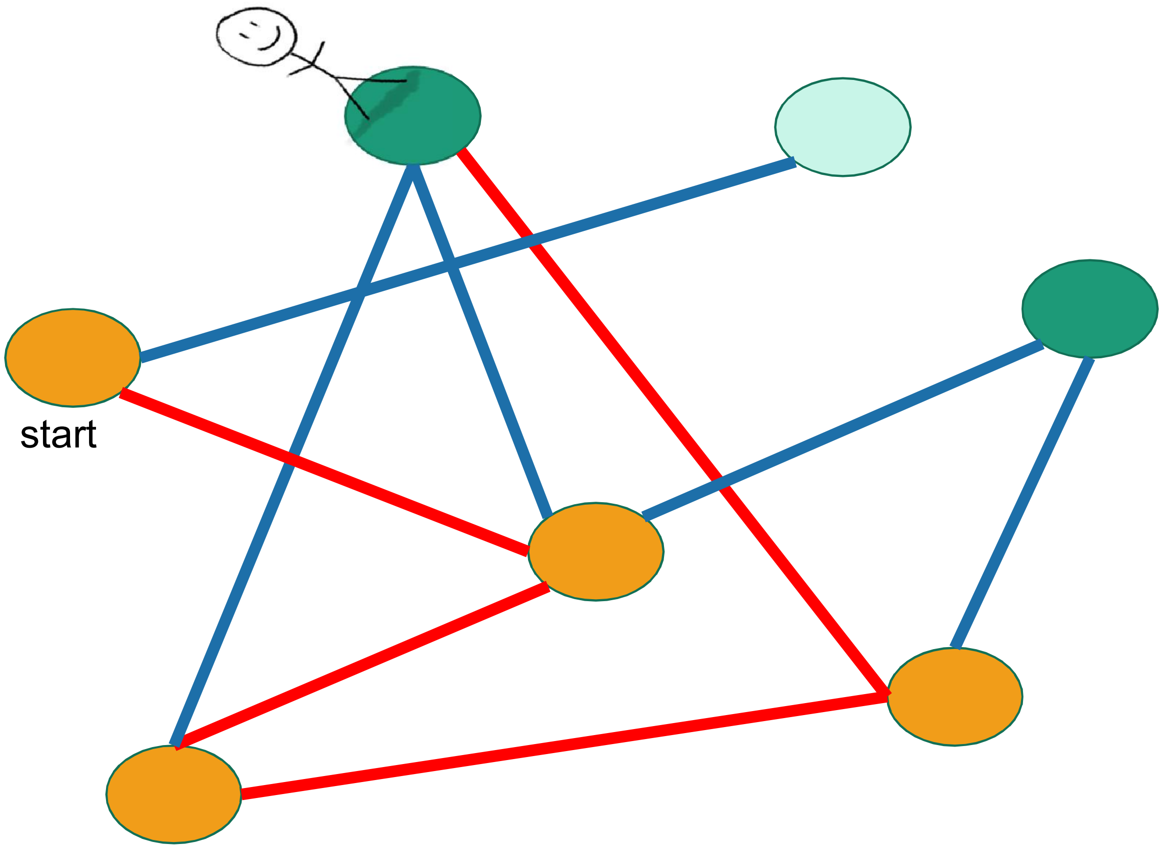




Depth First Search



Depth First Search

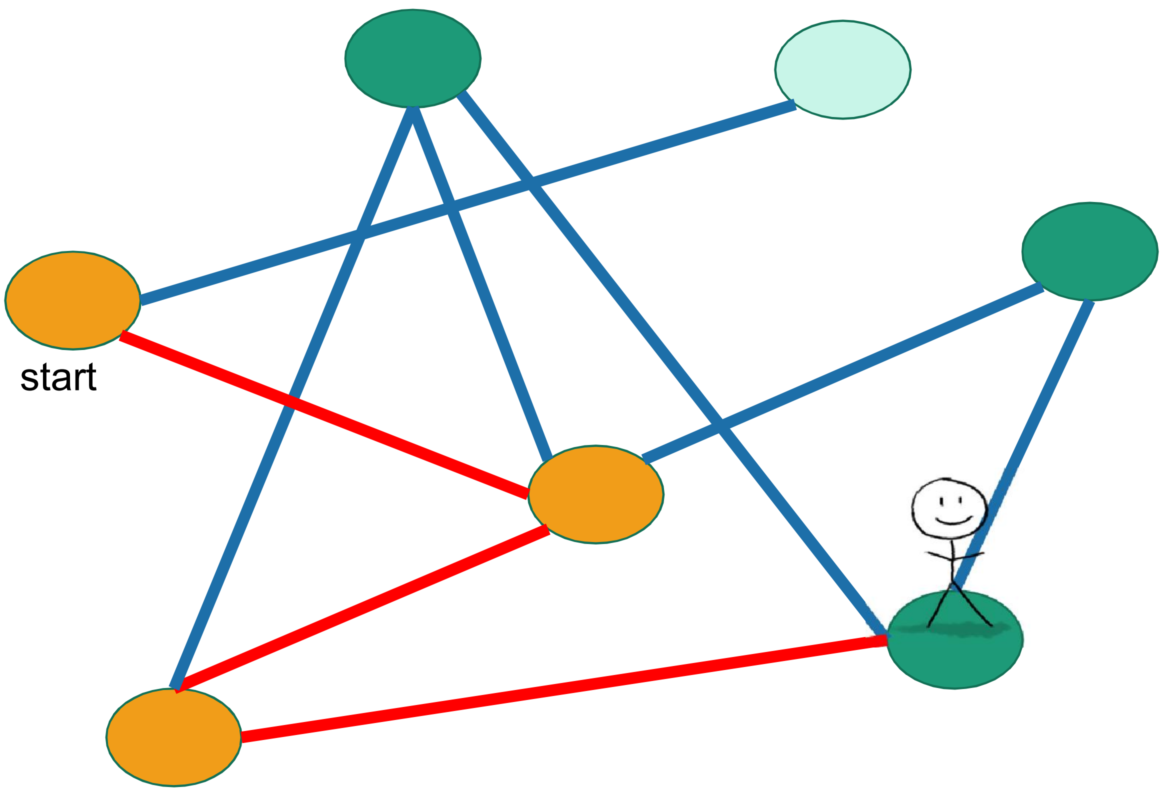


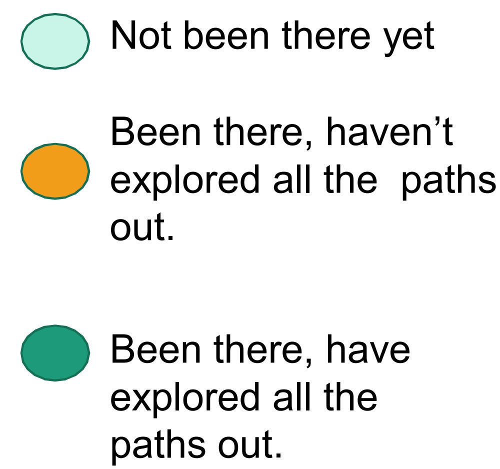
Not been there yet

Been there, haven't explored all the paths out.

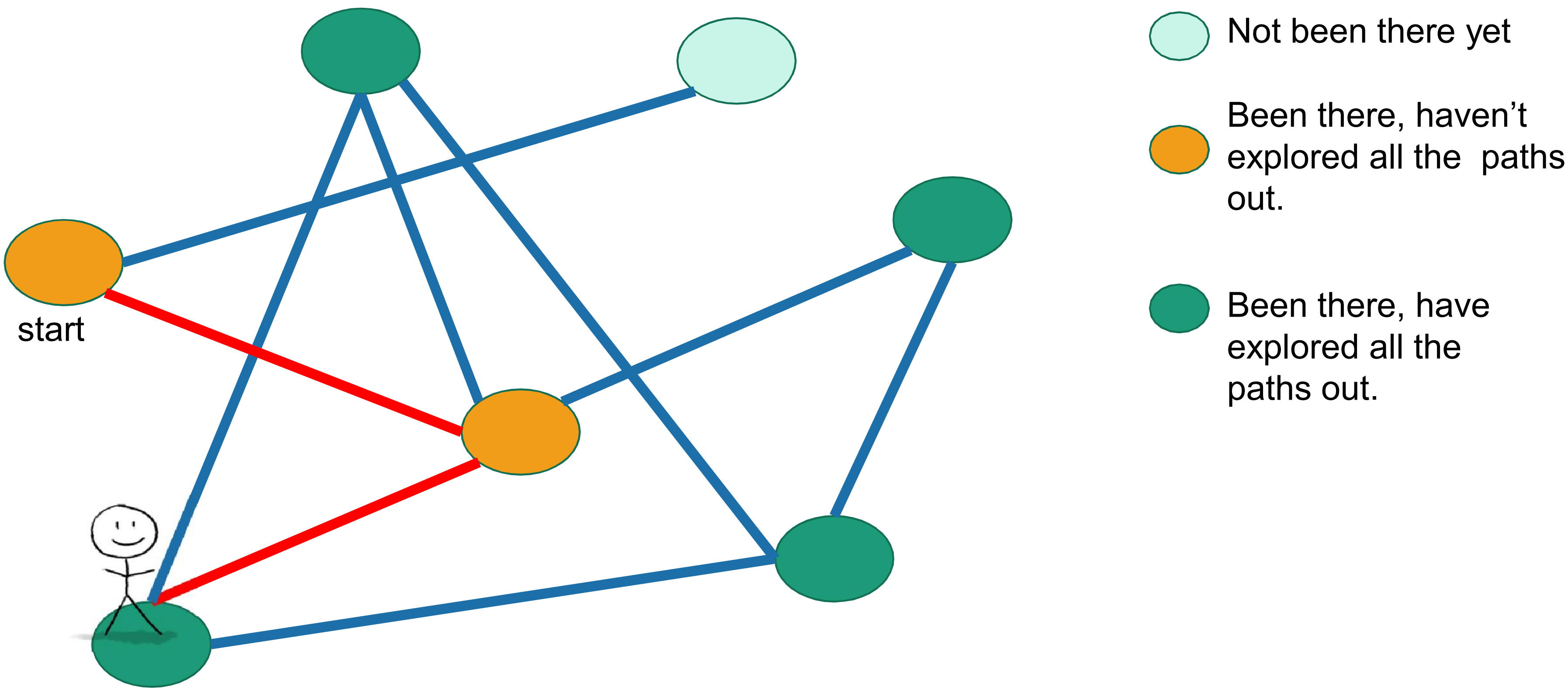
Been there, have explored all the paths out.

Depth First Search

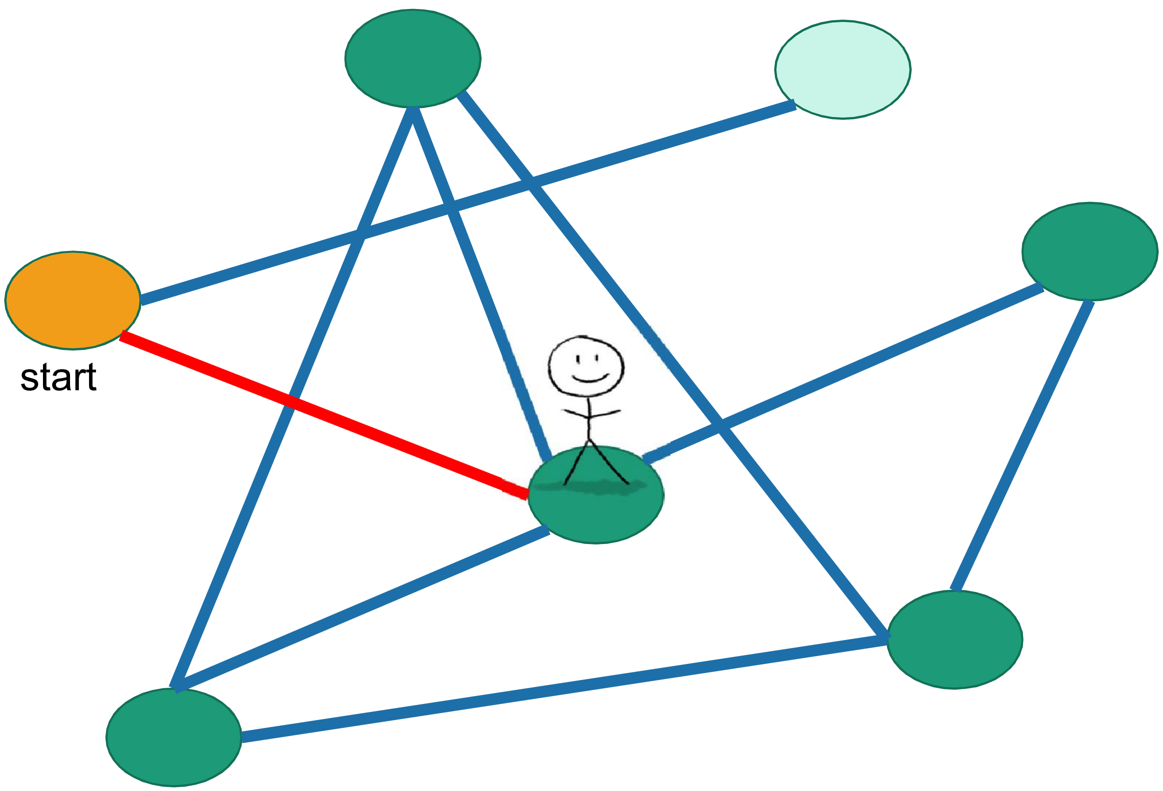




Depth First Search



Depth First Search

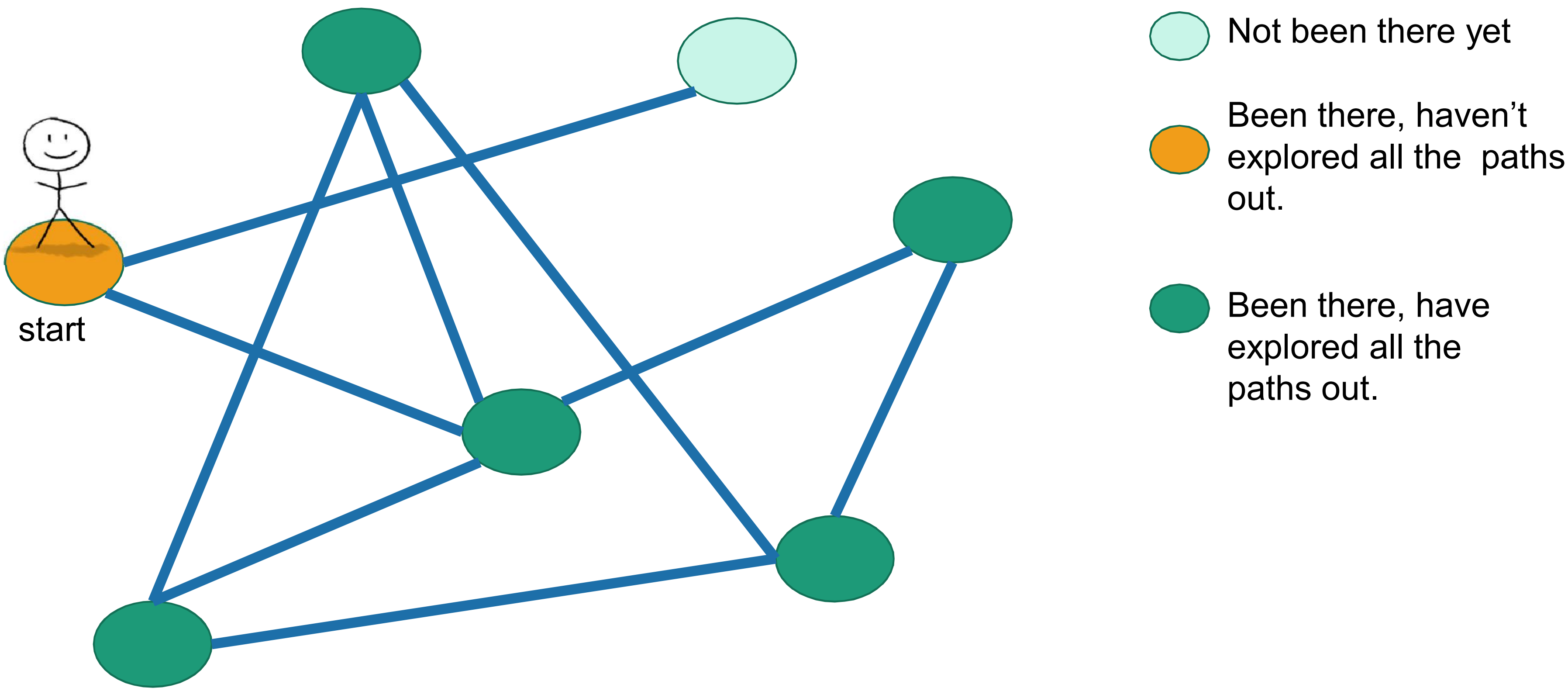


Not been there yet

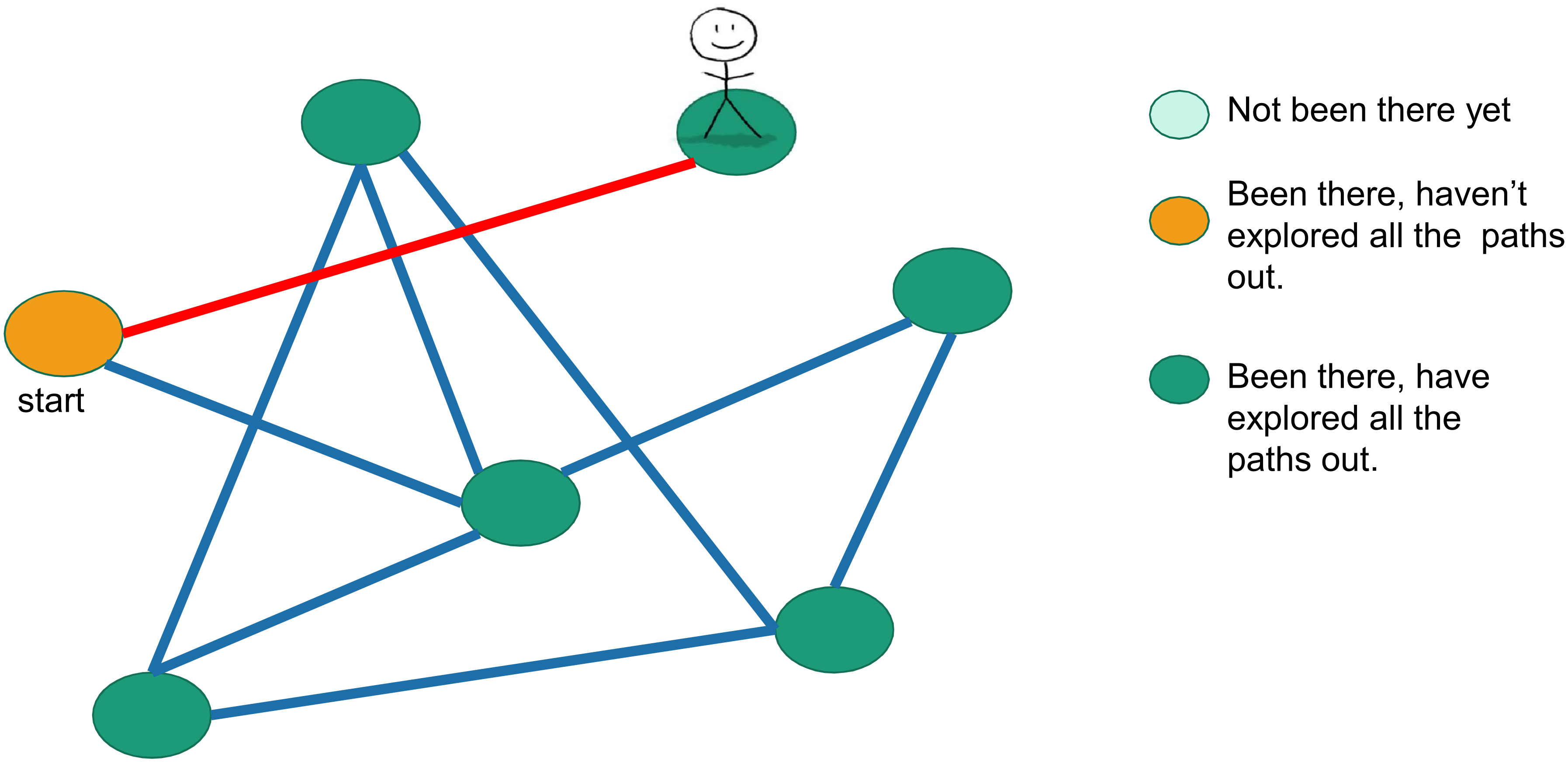
Been there, haven't explored all the paths out.

Been there, have explored all the paths out.

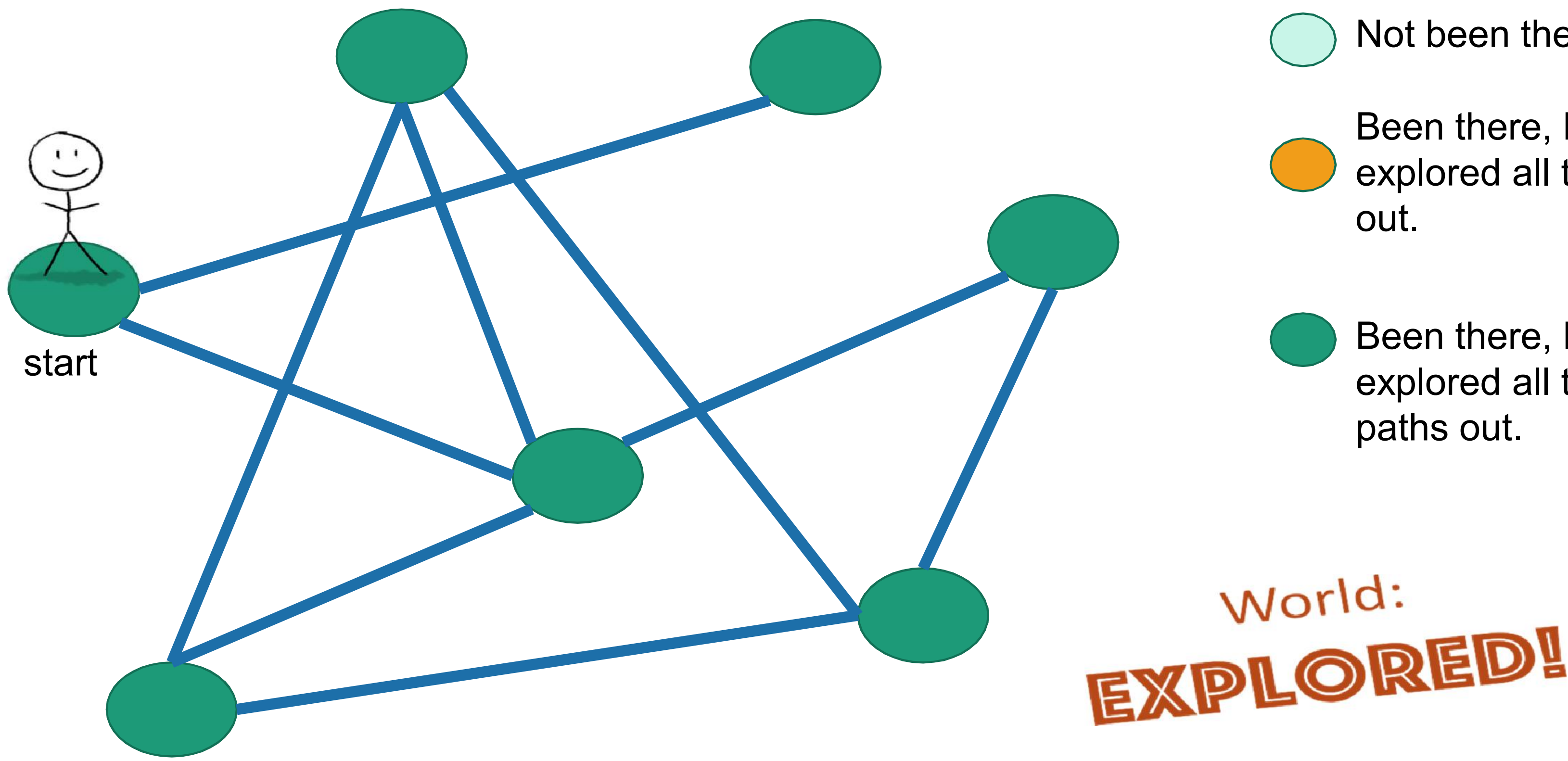
Depth First Search



Depth First Search



Depth First Search



BFS and DFS - comparison

- Space complexity of DFS is lower than that of BFS.
- Time complexity of both is same – $O(|V|+|E|)$.
- The behavior differs for graphs where not all the vertices can be reached from the given vertex s.
- Predecessor subgraphs produced by DFS may be different than those produced by BFS. The BFS product is just one tree whereas the DFS product may be multiple trees.