# Recitation 5

CSCI 2270
July 12, 2018

# Outline

Announcement: Moving office hours from Monday to Tuesday 11-12:30 next week

Homework common mistakes

Depth First Search

Breadth First Search

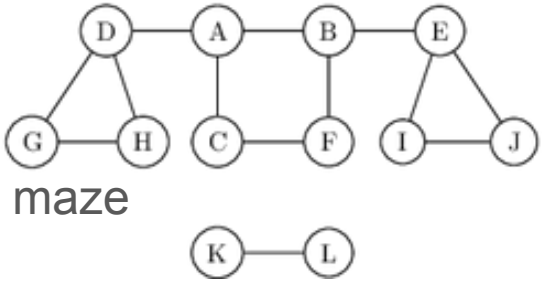
Dijsktra’s Algorithm

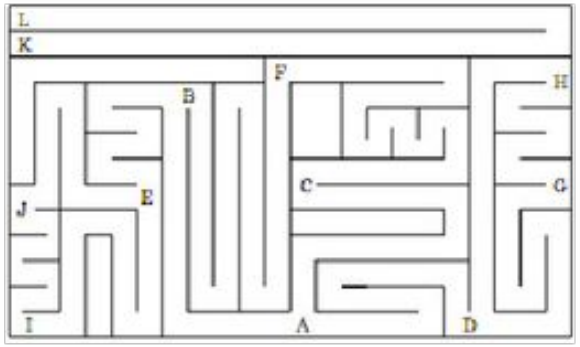
Recitation Quiz

# Homework

Counting the number of movies (642 not 50)

# Depth First Search




Let's say you're stuck in a corn maze

Apparently used by Kayak and CleverLayover

*Which parts of the graph are reachable from a given vertex*

Works on directed and undirected graphs

Keep track of:

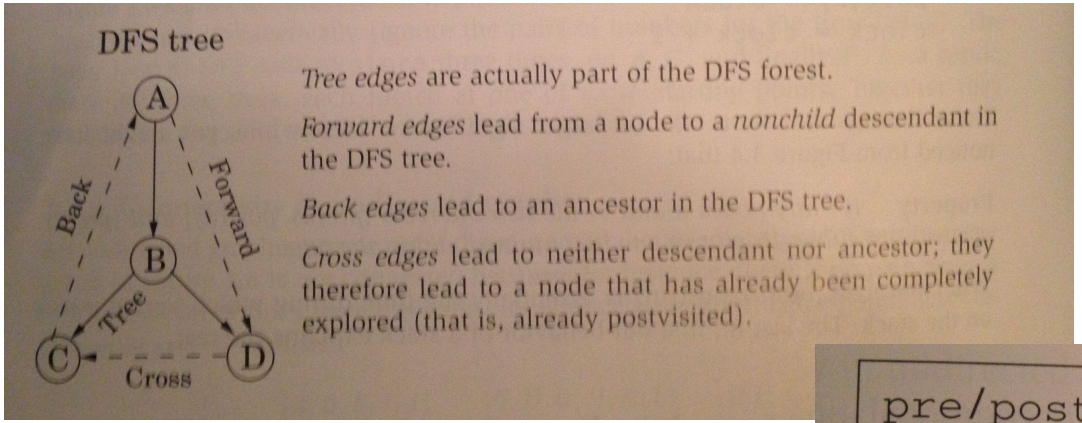
- pre/post visit
- visited (property of node)
- stack

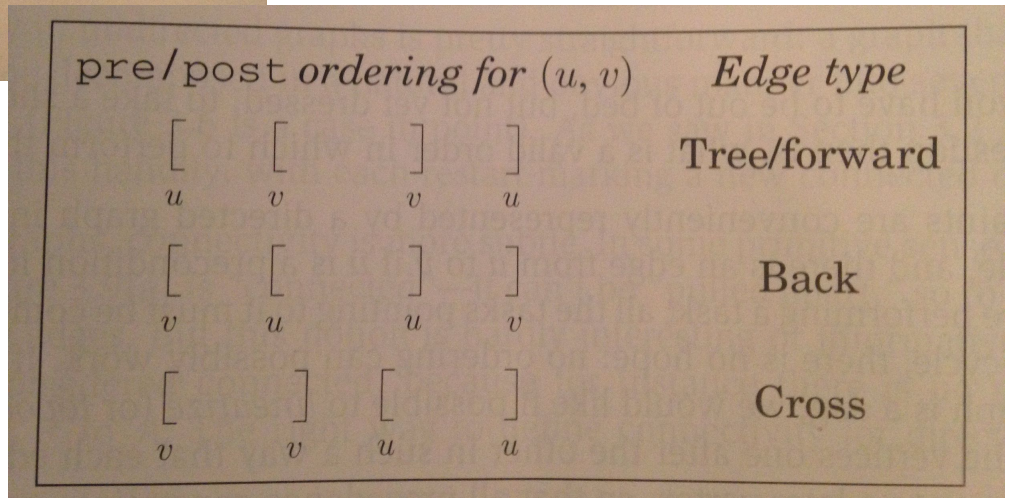
# Depth First Search Pseudocode

```
function explore(G,v)
    v.visited = true
    v.previsit = clock
    clock += 1
    for nei in v.neighbors
        If nei.visited = false
            explore(G, nei)
    v.postvisit = clock, clock += 1
```

```
function DFS(G)
    for all v in G:
        v.visited = false
    for all v in G:
        if v.visited =  false
            explore(G, v)
            distirctNum += 1
```

# Why keep track of visit time?



# Try it out

# Breadth First Search

BFS(G, starting_node)

Used to find *connected components*

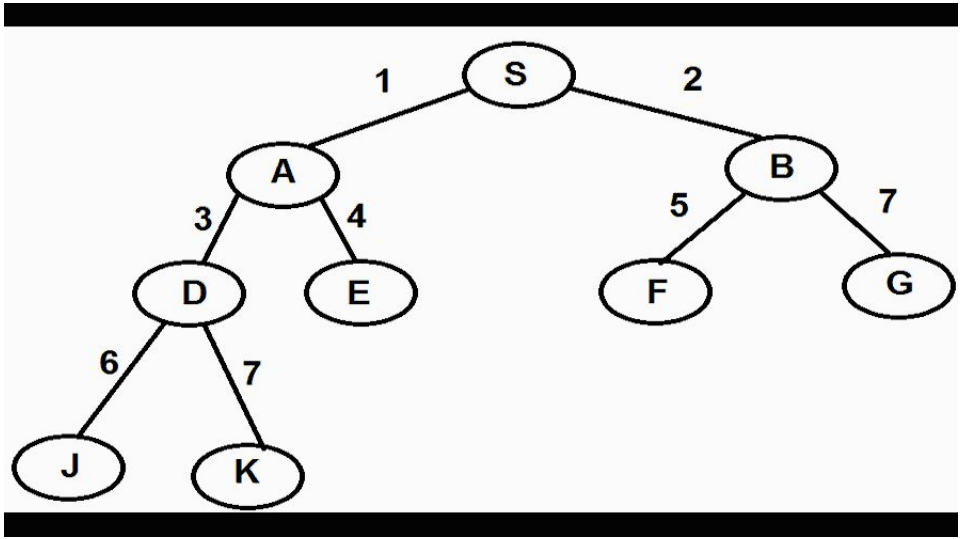
Keep track of:

- Queue (vector)
- Visited (boolean property of node)

O(|V| + |E|) algorithm

On directed or undirected, unweighted graphs

Can find shortest path on *unweighted* graphs



# Breadth First Search Pseudocode (without shortest distance)

```
BFS(starting_node)
    current = starting_node
    queue.enqueue(starting_node)
    while (queue is not empty)
        for neighbor in current.neighbors
            if neighbor.visited = false
                queue.enqueue(neighbor)
                neighbor.visited = true
        current = queue.pop_front()
    return visited neighbors
```

# Try it out

# Dijkstra’s Algorithm

Like BFS, but keeping track of weighted distances

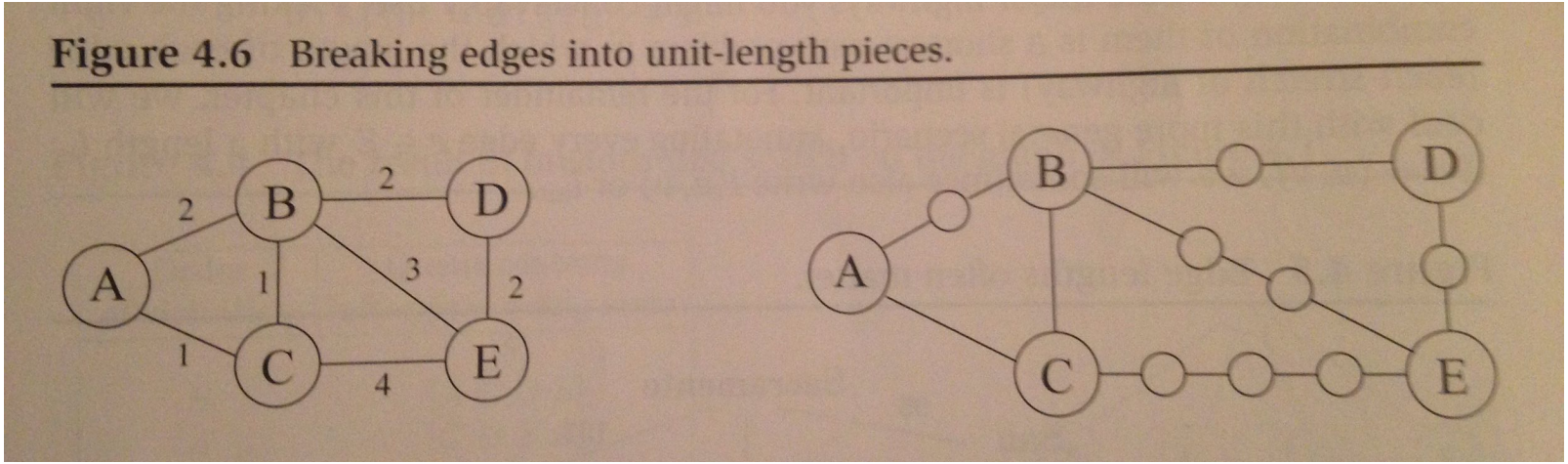
Shortest distance from starting_node to every other node in graph

Relies on the property that the shortest path from *s* to *t* is also the shortest path to any of the vertices along the path.

This is exactly what BFS does.

**Nonnegative edge weights**

# Couldn’t we just tweak the graph and run BFS?




Well, yeah, but it gets really inefficient.

# Dijkstra’s pseudocode

```
dijkstra(G,w,s)

    for all u in V: u.dist = infinity, u.prev = NULL

    s.dist = 0

    H = Makequeue(V)

    while H is not empty:

        u = H.pop_front()

        for all v in u.neighbors:

            if v.dist > u.dist + w(u,v):

                v.dist = u.dist + w(u,v)

                v.prev = u
```

# Quiz Password

# Quiz Password

FirstWatch

(Does that exist here???)