S. KIMMEL

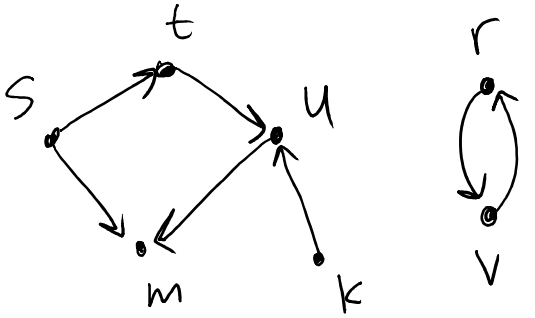
# Graph Search

## Desired Properties

1. Finds all nodes on all possible paths from starting node.
2. Efficient (doesn't look at the same vertex over and over)
   ↖ Will discuss next week

## Uses:

- Maps
- Web crawlers (find new web pages)
- Find new friends
- ??
- .
- .




Q: Which nodes are on paths from s?

A) t, m

B) t, m, u ⟸

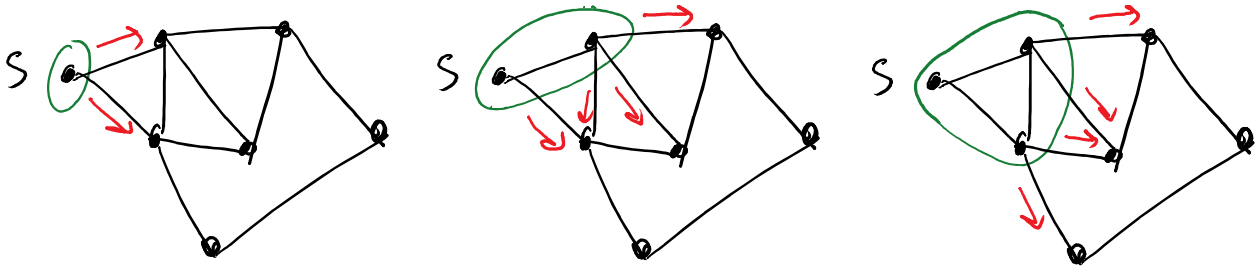
C) t, m, u, k

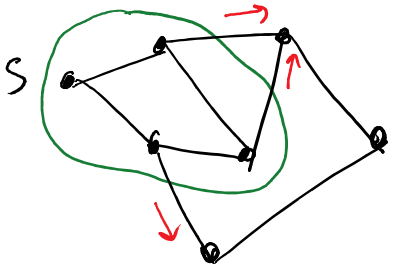
D) all nodes.

S. KIMMEL

**Idea:**

Cross an edge from the visited region to unvisited region



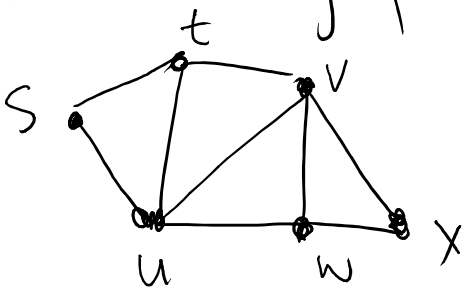




## Graph Search Algorithm

Input: $G=(V,E)$, starting node $s$

1. $Vis = \{s\}$ // Vis = set of visited nodes
2. While $\left(\exists\ \{u,v\} \in E : \left(u \in Vis \wedge v \notin Vis\right)\right)$
3. Add $v$ to Vis

Q: Consider the graph:




Which sequence of visited vertices is not possible?

A) s, t, u, w, x, v

B) s, u, v, x, w, t

C) s, u, v, t, x, w

D) s, t, w, x, u, v

S. KIMMEL

# Breadth-First-Search (BFS)

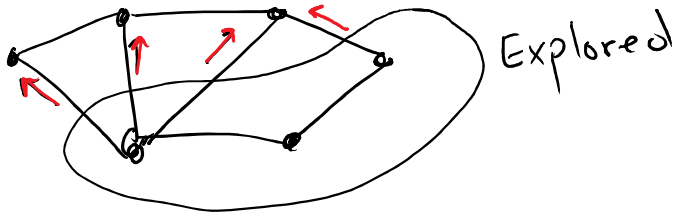
← Remind why this works

## Generic Search Alg:

1. Vis = {s} // vis = set of visited nodes
2. While $\left(\exists \{u,v\} \in E : \left(u \in vis \wedge v \notin vis\right)\right)$:
3. Add v to vis

## Big Question:

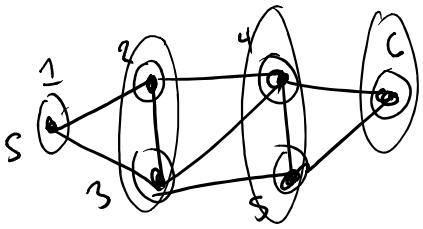
If multiple edges cross boundary between explored and unexplored, which to explore first?




Which new edge to explore?

## Breadth-First Search Strategy:

explore all edges crossing current boundary, then look at new boundary & explore




⟸ Breadth-First Search

S.KIMMEL

Input: Graph $G=(V,E)$, starting vertex $s \in V$
Output: List of found vertices:

- vis[v] = false $\forall$ $v \in V$ // mark true when visited
- A = $\emptyset$ // A is a queue
- Add s to A
- vis[s] = true

- while (A is not empty):
  - Pop v from A
  - for each edge {v, w}:
    - if (vis[w] = false):
      - Vis[w] = true
      - Add w to A

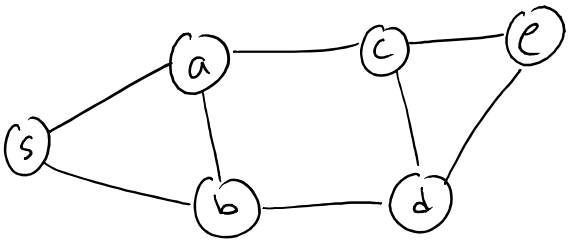
"First in first out" like a line at a dining hall. First in line is first to get food. Last in line is last to get food
Add: put in line
Pop: take out of line

* You do not need to write code to create a Queue for Programming assignment. Python + Java have packages to do this

Breadth First Search

S. KIMMEL

ex:




↑ out

A

| |
|---|
| s |
| ~~a~~ |
| ~~b~~ |
| ~~c~~ |
| ~~d~~ |
| ~~e~~ |
| e |

↑ in

exp

| | |
|---|---|
| s | T |
| a | F |
| b | F |
| c | F |
| d | F |
| e | F |