S. KIMMEL

Goals

- Analyze pseudocode using adjacency lists / matrices
- Describe graph search

Input: Adj Matrix A for $G = (V, E)$ (undirected, unweighted, no self loops)

Output: ??

```
1. S = 0
2. for u, v ∈ V        // alternate: for u = 1 to |V|:
                                     for v = 1 to |V|:
       S += A[u, v]
3. return S
```

A) $|V|$ B) $|V| \times |V|$ C) $|E|$ D) $2|E|$

We count each edge twice: if $\{b, c\} \in E$ then $A[b, c] = 1$ and $A[c, b] = 1$.

S. KIMMEL

Time complexity of previous

for loop is over every element of matrix: there are $|V|^2$ elements. Each iteration does $O(1)$ work.

$\Rightarrow$ Time complexity $= O(|V|^2)$

Input: Adj List A (unweighted, no self-loops) for $G = (V, E)$
(undirected)

1. $S = 0$

2. for $u \in V$
   $S \mathrel{+}= A[u].length$

3. return S

Output ??

A) $|V|$ B) $|V| \times |V|$ C) $|E|$ D) $2|E|$

We count each edge twice. If $\{b, c\} \in E$, then c is in A[b].length and b is in A[c].length

S.KIMMEL

What is time complexity?

A) $O(1)$ B) $O(|V|)$ C) $O(|E|)$ D) $O(|V|^2)$

The time complexity is O(|V|) because we have a for-loop that iterates through each vertex (and so does |V| loops), and does O(1) operations at each iteration

S. KIMMEL

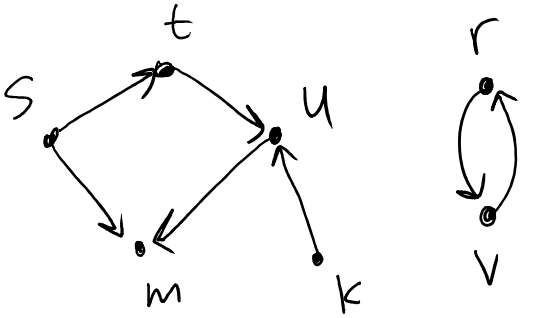
Graph Search

Desired Properties

1. Finds all nodes reachable from starting node
2. Efficient (doesn't look at the same vertex over and over)

Uses:

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



Q: Which nodes are reachable from s?

A) t, m

B) t, m, u ⟸

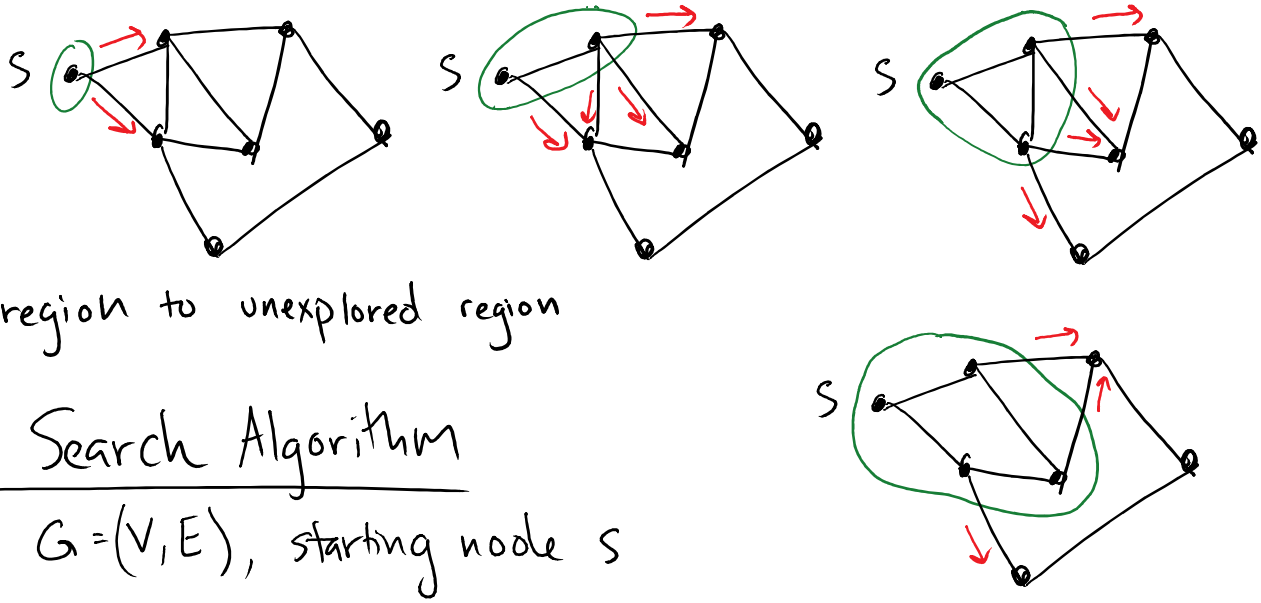
C) t, m, u, k

D) all nodes.

S. KIMMEL

Idea:

Explore an edge from the explored region to unexplored region

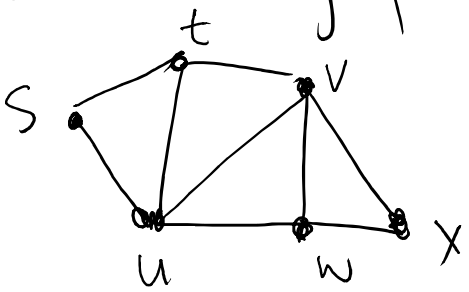


Graph Search Algorithm

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

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

Q: Consider the graph:



Which sequence of explored vertices is not possible?

A) s, t, u, w, x, v $\quad$ B) s, u, v, x, w, t

C) s, u, v, t, x, w $\quad$ D) s, t, w, x, u, v

S. KIMMEL

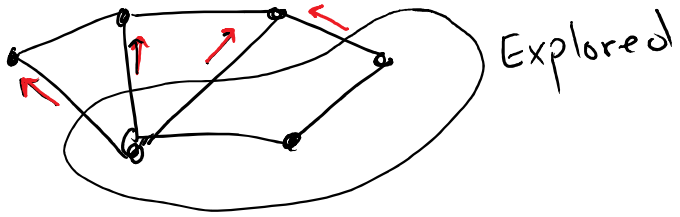
Breadth-First-Search (BFS)

← Remind why this works

Generic Search Alg:

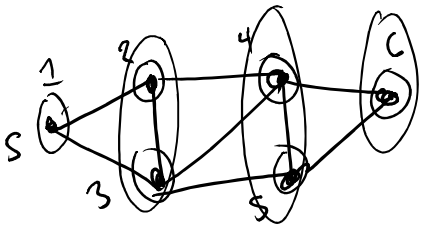
1. Exp = {s} Exp = set of explored nodes
2. While $\left(\exists \{u,v\} \in E : \left(u \in Exp \wedge v \notin Exp\right)\right)$
3. Add v to Exp

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