S. KIMMEL

Breadth-First-Search (BFS)

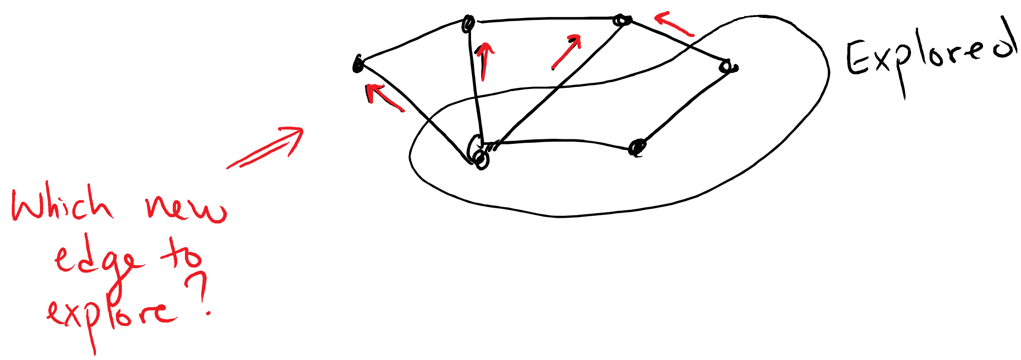
Remind why this works

Generic Search Alg:

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

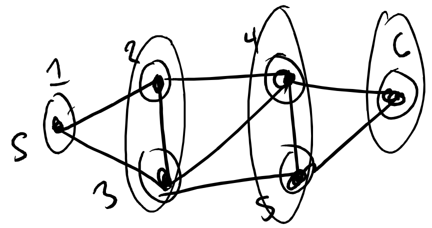
Big Question:

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



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:

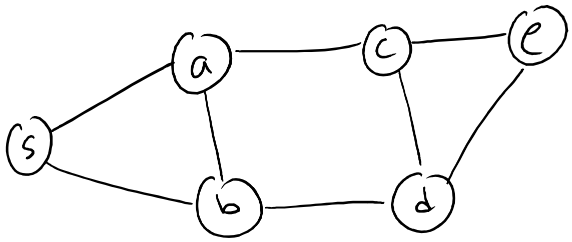
- $vis[v]=false \ \forall \ v \in V$ // mark true when visited
- $A=\emptyset$ // A is a $\boxed{\text{queue}}$
- Add s to A
- $vis[s]=true$

"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

- 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

Breadth First Search

ex:



out

A $\begin{bmatrix} \cancel{s} \\ \cancel{a} \\ \cancel{b} \\ \cancel{c} \\ \cancel{d} \\ e \\ \; \end{bmatrix}$

In

exp

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

S. KIMMEL

Q: What is the runtime of BFS using an adjacency list if $n = |V|$, $m = |E|$, n_s = # of vertices with paths from s, m_s = # of edges on paths from s.

A) $O(m_s)$

B) $O(n + m_s)$

C) $O(n_s \cdot m_s)$

D) $O(n + n_s m_s)$

S.KIMMEL

Explain: Why is runtime $O(n+m_s)$

- Runtime dominated by looking at edges
- Each edge can only be examined when its adjoining vertex is popped. Each vertex can only show up in QUEUE one time. ⇒ Each edge is examined twice
- Finding next edge to visit takes constant time b/c use adjacency list.

$O(m_s)$ to do while loop

edges connected to s

+

Initialization: $O(n)$

$$\boxed{O(n+m_s)}$$

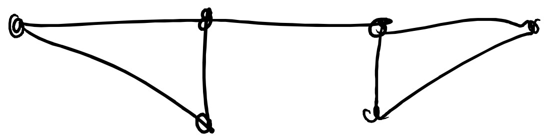
Answer: B

S.KIMMEL

Trees - connected graph with no cycle or self loops

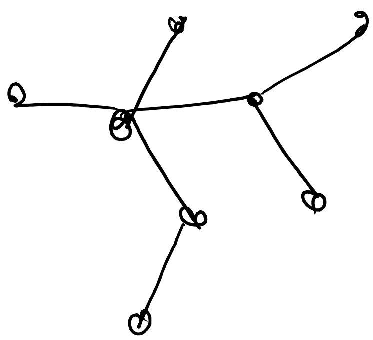
Tree

Not tree



Not tree

Not tree

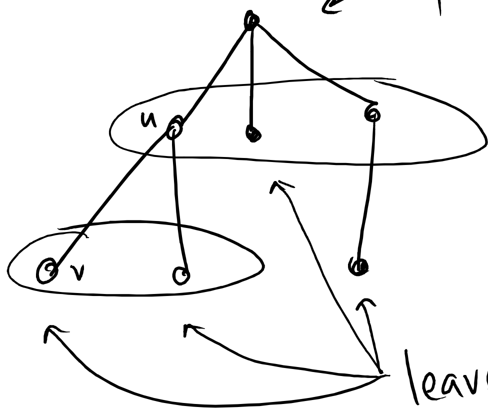


Tree

S. KIMMEL

Rooted tree

special vertex is called root



distance 1 = children of root

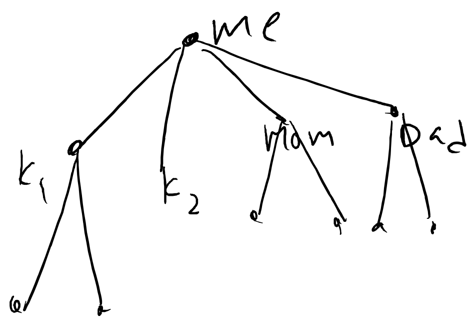
$\{u,v\} \in E$, if u closer to root, u is "parent" of v, v is "child" of u.

leaves = nodes without children

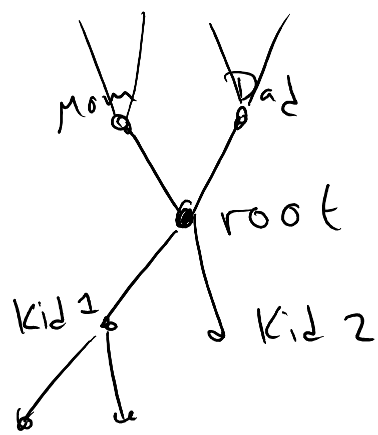
S. KIMMEL

Q: Consider family tree tree. If I am the root, which nodes are child nodes of me?

A) My children B) My Parents C) Children & Parents



=



<u>def</u>: An <u>k-ary</u> tree is a rooted tree where every node has at most K children.

Most Famous in Computer Science: Binary Tree = 2-ary tree.

Applications:
- Data structures
- Codes