**Breadth-First-Search (BFS)**

Generic Search Alg:
1. \( \text{vis} = \{s\} \) // \( \text{vis} \) = set of visited nodes
2. While \((\exists \{u,v\} \in E : (u \in \text{vis} \land v \notin \text{vis}))\):
3. \( \text{Add v to vis} \)

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

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

while ($A$ is not empty):
    . Pop $v$ from $A$
    . for each edge $\{v,w\}$:
        . if ($\text{vis}[w] = \text{false}$):
            . $\text{vis}[w] = \text{true}$
            . 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

Breadth First Search
EX:

```
A
s
a
b

\begin{array}{c|c|c|c|c}
 s & a & b & c & e \\
 \hline
 s & x & x & x & x \\
 a & x & x & x & x \\
 b & x & x & x & x \\
 c & x & x & x & x \\
 e & x & x & x & x \\
\end{array}
```

exp

```

\begin{array}{c|c}
 s & F \\
 a & F \\
 b & F \\
 c & F \\
 d & F \\
 e & F \\
\end{array}
```
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) \( O(n+n_s \cdot m_s) \)
Explain: Why is runtime $O(n+ms)$

- 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.

$O(ms)$ to do while loop

$\leftarrow$ # edges connected to s

$+ $

Initialization: $O(n)$

Answer: B $O(n+ms)$
Trees - connected graph with no cycle or self loops

- Tree
- Not tree
- Not tree
- Not tree
- Tree
Rooted tree

- special vertex is called root

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

- distance 1 = children of root

- leaves = nodes without children
Q: Consider family tree. If I am the root, which nodes are child nodes of me?

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

def: An $k$-ary 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