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)

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
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 (∃ u, v ∈ E : (u ∈ Vis ∨ v ∈ Vis))
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
**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. Add \(v\) to \(\text{vis}\)

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:

- $\text{vis}[v] = \text{false} \ \forall \ v \in V$ \hspace{1cm} // mark true when visited
- $A = \emptyset$ \hspace{1cm} // $A$ 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

*You do not need to write code to create a Queue for programming assignment.*

Python + Java have packages to do this

---

Breadth First Search
Ex:

\[
\begin{array}{c}
S \\
\downarrow \\
A \\
\begin{bmatrix}
S & a & b & c & d & e \\
\end{bmatrix}
\end{array}
\]

\[
\begin{array}{c}
\text{out} \\
\downarrow \\
\text{in}
\end{array}
\]

exp

<table>
<thead>
<tr>
<th></th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>T</td>
</tr>
<tr>
<td>a</td>
<td>T</td>
</tr>
<tr>
<td>b</td>
<td>F</td>
</tr>
<tr>
<td>c</td>
<td>F</td>
</tr>
<tr>
<td>d</td>
<td>F</td>
</tr>
<tr>
<td>e</td>
<td>F</td>
</tr>
</tbody>
</table>