Shortest Paths

**Input:** Graph $G = (V, E)$, $s \in V$

**Output:** $\forall v \in V$, $\ell(v)$ = shortest path from $s$ to $v$

$\ell(v) = \infty$ if $s, v$ not connected

Applications: Bacon #
Linked in Degree

**Idea, explore layers.**

$\ell[v] = \infty \quad \forall v \in V$ // will store shortest paths

$Ex[v] = True \quad \forall v \in V$ // mark True when explored

$A = [\{}$

$A.add(s)$

$\ell[s] = 0$

$Ex[s] = True$

```
while (A is not empty) {
    v = A.pop

    For each edge $(v, w)$
        If ($Ex[w] = false$)
            A.add(w); $Ex[w] = true; \ell[w] = \ell[v] + 1$;
```

This is Breadth First Search - slowly move away in layers from initial node.

Without red: BFS, With red: BFS for shortest path

---

*$\star$ QUEUE: FIFO $\star$

or $\star$ STACK: FILO $\star$
Now need to figure out...
- is it correct?
- what is the runtime

Q: What strategy can you use to prove correct?
Discuss...
Q: What is runtime of BFS using adjacency list.

If \( n \) is total # of vertices, \( m \) is total # edges, \( n_s \) is # of vertices reachable from \( s \), \( m_s \) is # edges reachable from \( s \).

A. \( O(m_s) \)  
B. \( O(n + m_s) \)  
C. \( O(n_s \cdot m_s) \)  
D. \( O(n + n_s \cdot m_s) \)

Answer: B.

1. Initializing \( \text{Exp}[v] \) \( \forall v \in V \) takes time \( O(n) \).

2. (After initialization)

   Seems like should be \( n_s \cdot m_s \) b/c while loop runs \( O(n_s) \) times, for loop runs \( O(m_s) \) times. Can do better!

⇒ Inside while loop, do a constant # of operation for each edge that is examined. We only examine edges of vertices that end up in queue. Each edge only has two vertices connected to it. Each vertex can only be in queue once. (Only get to edges connected to \( s \).) ⇒ \( \sim 2m_s \) operations.
What kind of algorithm is BFS shortest path?

Greedy

Local Search

- look for best choice in nearby area

Dynamic - Since stores solutions to previous problems, one could view as dynamic...

but most textbooks would **not**

describe as dynamic