Dijkstra’s Shortest Path Algorithm

First let’s review breadth first search (BFS) that solves the single-source shortest path algorithm for an **unweighted** graph.
**bfs(V source)**

For each \( u \in V \)
- \( u\text{.discovered} = \text{False} \)
- \( u\text{.dist} = \infty \)
- \( u\text{.parent Edge} = \text{null} \)
- \( \text{Source}\text{.discovered} = \text{true} \)
- \( \text{Source}\text{.dist} = 0 \)
- \( \text{queue}\text{.enqueue}(\text{source}) \)

while (!\( \text{queue}\text{.isEmpty}() \))
- \( u = \text{queue}\text{.dequeue}() \)

for each edge \( e \) in outgoing edges from \( u \)
- \( v = e\text{.dest} \)
  - if (!\( v\text{.discovered} \))
    - \( v\text{.discovered} = \text{true} \)
    - \( v\text{.dist} = u\text{.dist} + 1 \)
    - \( v\text{.parent Edge} = e \)
    - \( \text{queue}\text{.enqueue}(v) \)
Upon completion of BFs

any vertices not discovered are not reachable from source

For all discovered vertices \( V \)
\( V.dist \) is \# of edges in a shortest path from source to \( V \)

Following parent references from \( V \)
to \( S \) gives a shortest path
from \( S \) to \( V \) (in reverse order)

Time complexity \( O(N+M) \) with adj list
Example from last class

Shortest path tree

X means it was discovered

Shortest path from a to h

$e_{11}, e_3, e_5$
How can we solve the single source shortest path algorithm in a weighted graph where all weights $\geq 0$?

What goes wrong with BFS?

![Graph with weights]  

Weight $3 + 4 + 2 = 9$
Use of queue in BFS was to process vertices in order of distance from source.

Reachable from 5:

\[ 5 \rightarrow \ldots \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow \ldots \]

Picked the reachable vertex in queue (not yet "placed in shortest path tree") with shortest distance from source.
**Dijkstra Single Source Shortest Path (Vertex S)**

1. \( S\_\text{tracker} = \text{pq}.\text{addTracked}(0, S) \)
2. \( \text{tag: semantics} \)
3. \( \text{For all } v \in V - \{S\} \)
4. \( \text{length of shortest path from } S \)
5. \( \text{Found so far} \)
6. \( \text{V} \_\text{tracker} = \text{pq}.\text{addTracked}(\infty, S) \)
7. \( \text{V} \_\text{parent} = \text{null} \)
8. \( u \text{dist} \) \( \Rightarrow \) \( e \text{, weight} \)
9. \( \text{While (!pq.isEmpty())} \)
10. \( \text{tag} = \text{pq}\_\text{minTag}() \)
11. \( \text{if (tag} = \infty) \text{return} \) // remain vertices in pq are not reachable from S 
12. \( u = \text{pq}\_\text{extractMin}() \)
13. \( u\_\text{dist} = \text{tag} \)
14. \( \text{For each edge from } u \)
15. \( v = u\_\text{dest} \)
16. \( \text{if (v}._\text{tracker} \in \text{Collection}() \&\& \)
17. \( u\_\text{dist} + e\_\text{weight} < v\_\text{current tag} \)
18. \( v\_\text{tracker}.\text{update}(u\_\text{dist} + e\_\text{weight}) \)
19. \( v\_\text{parent} = e \)