Breadth-first Search

Repeat until the queue is empty:

* Remove vertex *v* from queue
* Add to queue all unmarked vertices adjacent to v and mark them

DFS vs BFS

DFS: puts unvisited vertices on a stack

BFS: puts unvisited vertices on a queue

Shortest path: Find path from s to t that uses fewest number of edges (BFS)

---

BFS(from source vertex s)

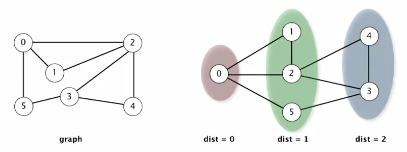
Put s onto a FIFO queue, and mark s as visited  
Repeat until the queue is empty

* Remove least recently added vertex v
* Add each of v’s unvisited neighbors to the queue, and mark them as visited

---

BFS examines vertices in increasing distance from s…

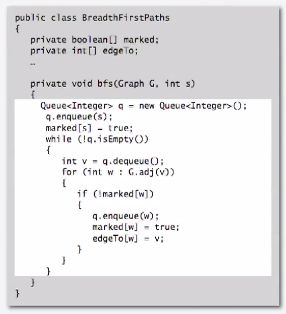
**Proposition:** BFS computes shortest paths (fewest number of edges) from s to all other vertices in graph in time proportional to E + V)



Proof: queue always consists of zero or more vertices of distance k from s, followed by zero or more vertices of distance k +1. THEREFORE, first time we get to a vertex is the shortest path to said vertex

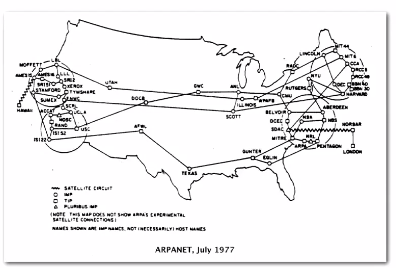
Proof (running time) : each vertex connected to s is visited only once (E + V)

BFS Implementation



BFS Application

* Routing (fewest number of hops in a communication network)



* Degrees of Kevin Bacon
  + Include one vertex for each perform and one for each movie
  + Connect a movie to all performers that appear in that move
  + Compute the shortest path from s = Kevin Bacon

