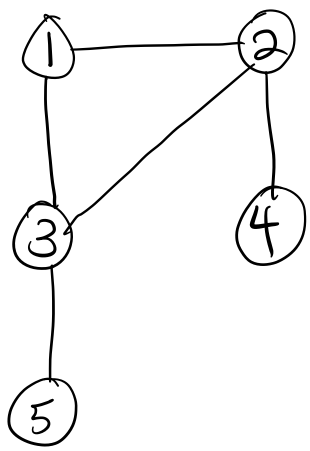
Graphs, continued.

Recall: A graph can be represented by a

Adjacency Matrix



$$\begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & & 1 & 1 & 0 & 0 \\ 2 & 1 & & 1 & 1 & 0 \\ 3 & 1 & 1 & & 0 & 1 \\ 4 & 0 & 1 & 0 & & 0 \\ 5 & 0 & 0 & 1 & 0 & \end{array}$$

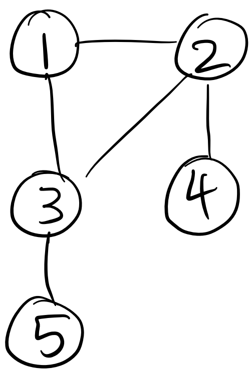
appropriate value for the diagonal may be application specific.

Pro: Fast look-up for "Is 3 adjacent to 4?"

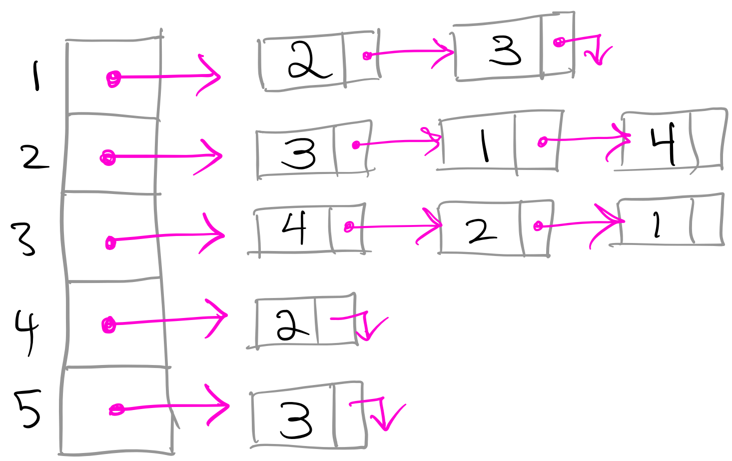
Con: - $\Theta(n^2)$ space even if m is $\Theta(n)$

- $\Theta(n)$ to process all neighbours of a vertex v even if v has no neighbours

Adjacency List Representation.



Logically we can organize it as follows:



Add an edge (4, 5):

- insert $\boxed{5}$ into (head of) 4's list
- insert $\boxed{4}$ into (head of) 5's list

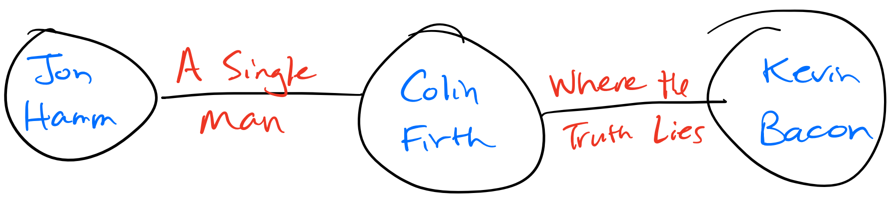
<u>Definition</u>: The <u>degree</u> of a vertex is the number of edges <u>incident</u> with the vertex

eg. vertex 1 has degree 2.

Also, u is <u>adjacent</u> to vertex v if

$(u,v) \in E$.

Important Application: Bacon number.



Bacon number: <u>length</u> of shortest path to

Kevin Bacon in the "worked with" graph.

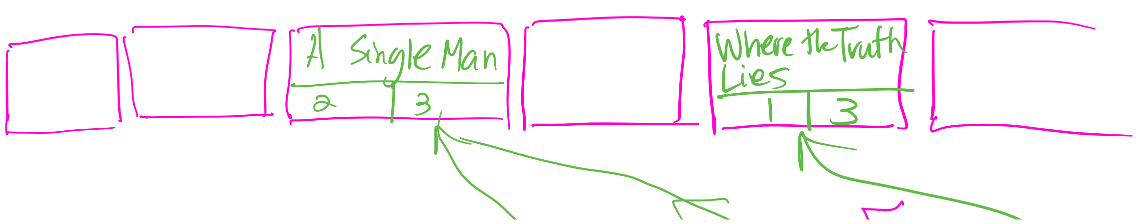
path length = # edges

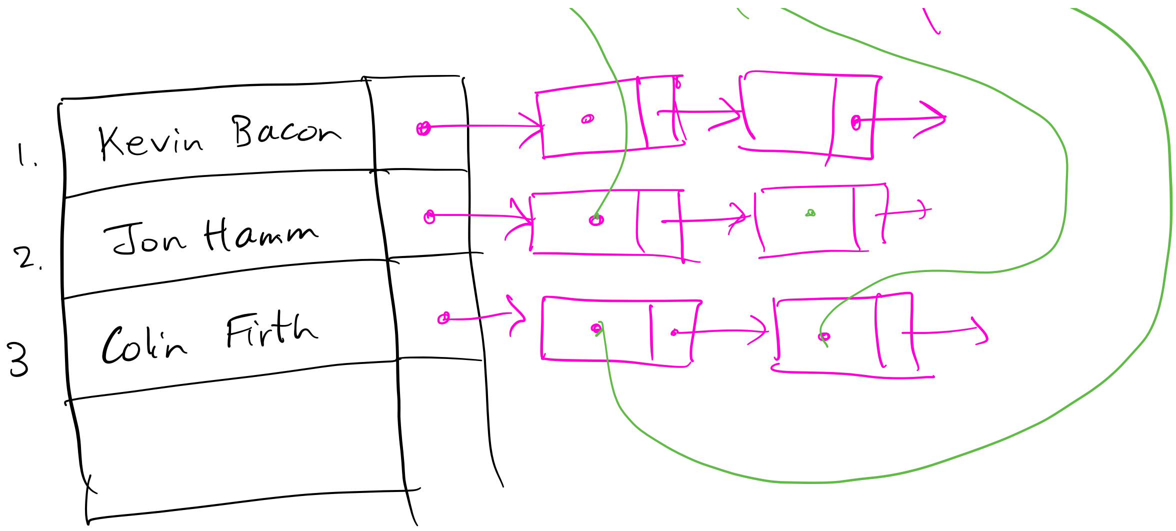
Note that a vertex can have extra info stored

"at" the node.

Also, an <u>edge</u> may have data associated with

it.





Blazingly Fast Graph Algs $O(n+m)$

GraphSearch(G, v)

Input: An undirected or directed graph $G=(V,E)$ and a start vertex $s \in V$

Goal: Identify the vertices of G reachable from s.

"reachable" means $\exists$ a path from s

Generic Search

Input: graph $G = (V, E)$, vertex $s \in V$

Postcondition: vertex $u \in V$ is reachable from s iff marked[u] = true

marked[s] = true

while $\exists$ an edge (v, w) with // underdetermined
marked[v] == true and
marked[w] == false,
- Choose such an edge (v, w)
- marked[w] = true.

Claim: (Correctness of Generic Search)
At the conclusion of Generic Search, a vertex v has marked[v] == true iff $\exists$ an s-v path in G.

Proof: We will assume G is undirected, but this proof can be adapted to the directed case.

I. ($\Rightarrow$) BWOC. Suppose Generic Search is run, and let u be the <u>first</u> vertex that is <u>not</u> reachable from s to be marked. Then $\exists$ an edge (w, u) where w was already marked by this time.

$\therefore$ w is reachable from s, ie $\exists$ path

$s, v_1, v_2, \dots, w$.

But then $s, v_1, v_2, \dots, w, u$ is also a path in G.

Then u is reachable from s in G

$\Rightarrow\Leftarrow$

$\therefore$ <u>if</u> u is marked, <u>then</u> u is reachable.

II ($\Leftarrow$) By induction on the <u>distance</u> of a vertex from s.

distance $= 0$: s is marked in first step.

Suppose true for dist $= i \geq 0$

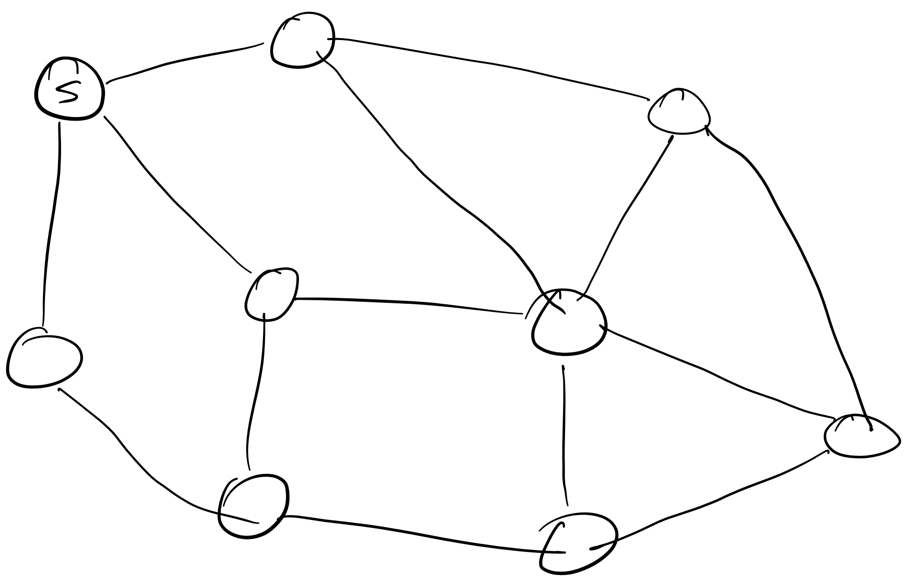
Let u be a vertex at distance $i+1$

Then $\exists$ edge (w, u) where w is at distance i.

Then w is marked.

Then u will eventually be marked

(algorithm cannot halt while $\exists$ an edge like w). $\boxtimes$

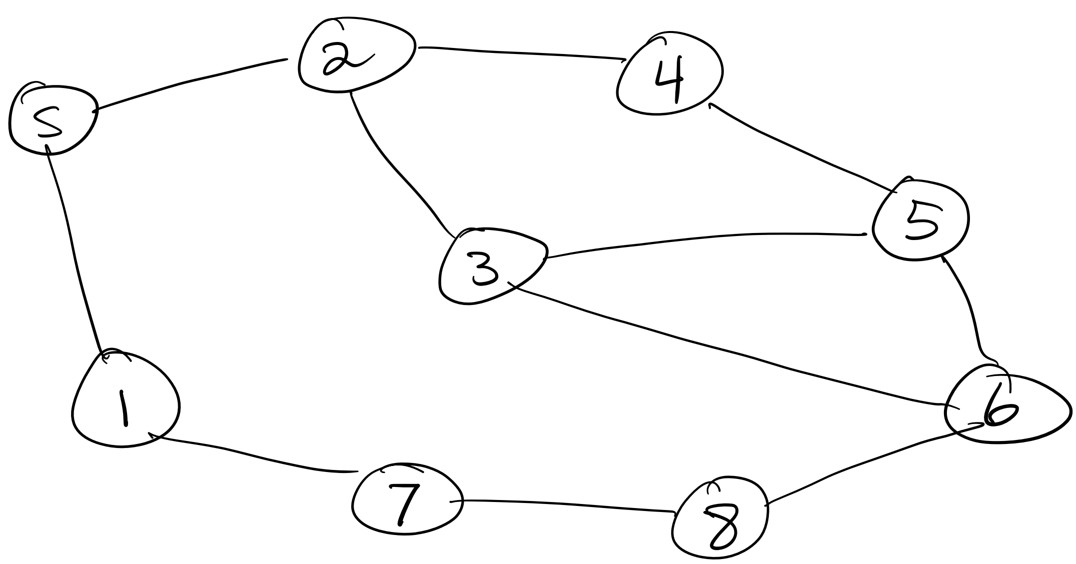


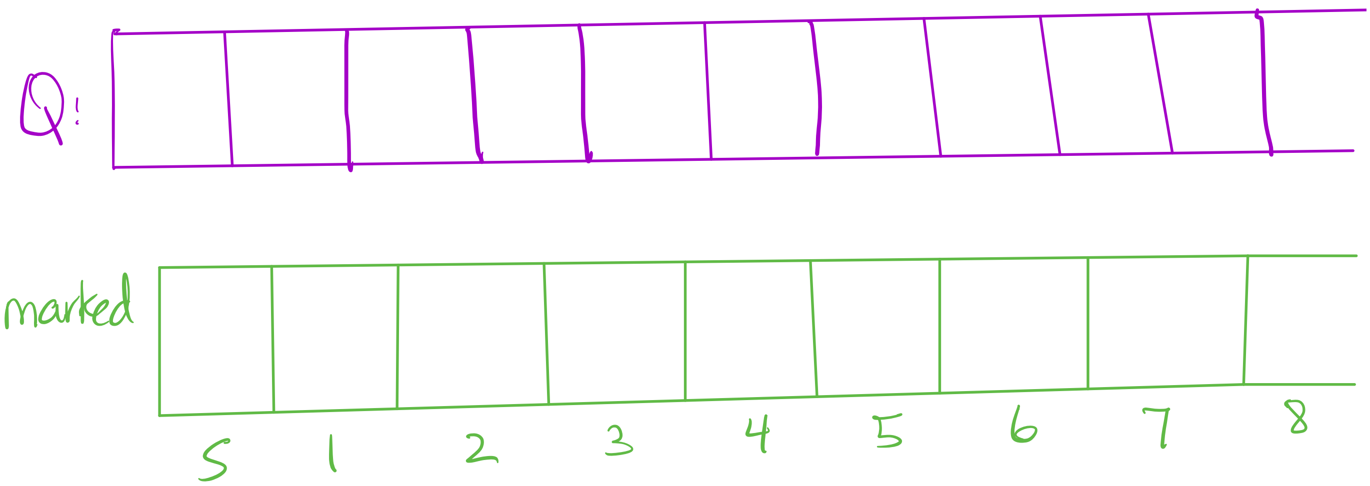
Breadth First Search.

Input: graph $G=(V,E)$ as adj-list; vertex s.

Postcondition: marked[v] == true $\iff$ v is reachable from s.

1. marked[s] = true
2. Q = queue data structure, initially containing just s
3. while Q is not empty do
4. remove vertex v from front of Q
5. for each edge (v, w) in v's adj-list do
6. if !marked[w]
7. marked[w] = true
8. add w to end of Q.




Q:
marked
S
1
2
3
4
5
6
7
8