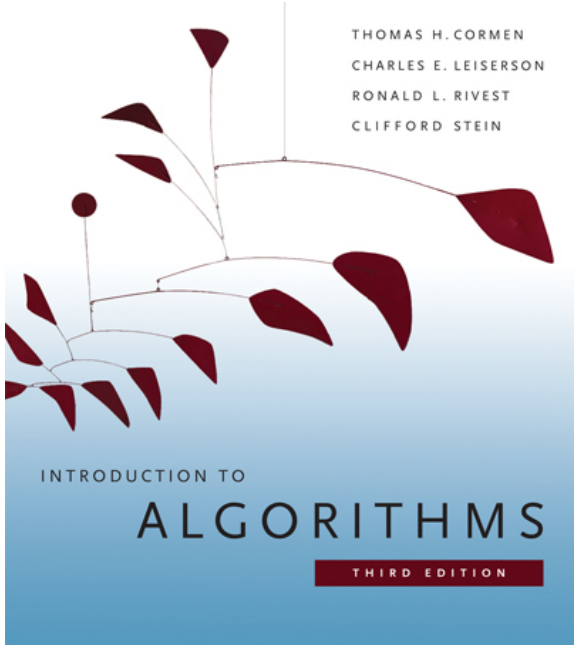
6.006- *Introduction to Algorithms*



Lecture 12

Prof. Silvio Micali

CLRS 22.2-22.3

Graphs

- $G = (V, E)$
- V a set of vertices

  $|V|$ denoted by n. Often: $V = \{1, \ldots, n\}$
- $E \subset V \times V$ a set of *edges* (pairs of vertices)

  $|E|$ denoted by m $\leq n(n-1) = O(n^2)$

Graph Flavors

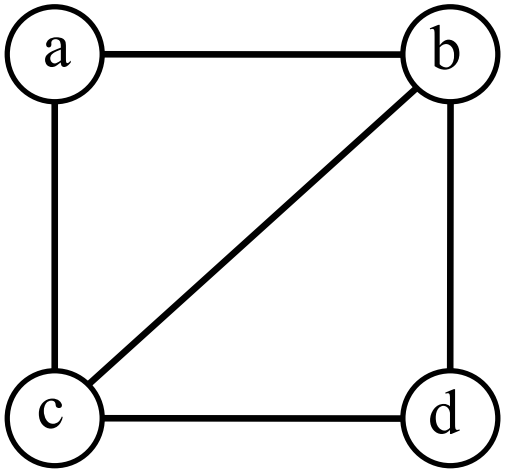
- *Directed*: "edges have a direction" I.e., $(i, j) \equiv i \rightarrow j$
- *Undirected*: "$\{i, j\}$ not (i, j)": $(i, j) \equiv (j, i) \equiv i - j$

  $\leq n(n-1)/2$ possible edges

Examples

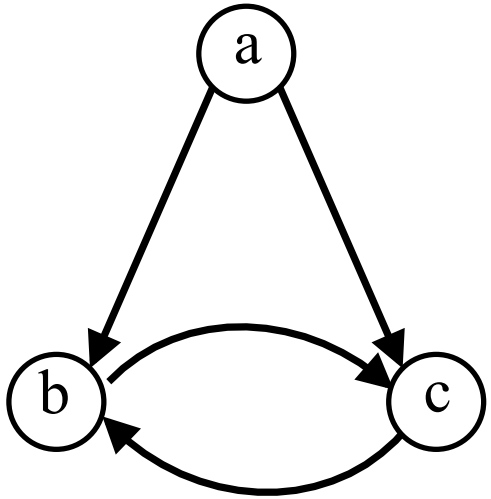
Undirected

- V={a,b,c,d}
- E={{a,b}, {a,c}, {b,c}, {b,d}, {c,d}}



Directed

- V = {a,b,c}
- E = {(a,c), (a,b) (b,c), (c,b)}



Graphs model lots of stuff:

- Friendship
- Powergrids
- Maps
- …

Why?

Functions are basic

$$F: X \to Y$$

Relations are more basic!

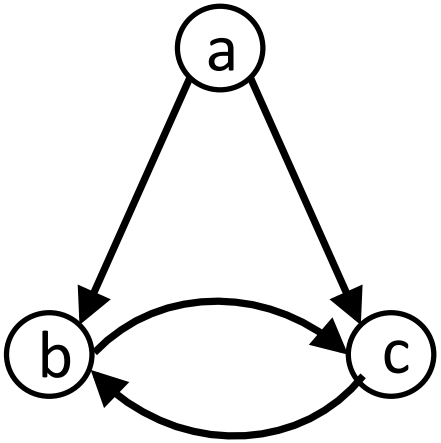
$$R \subset X \times Y$$

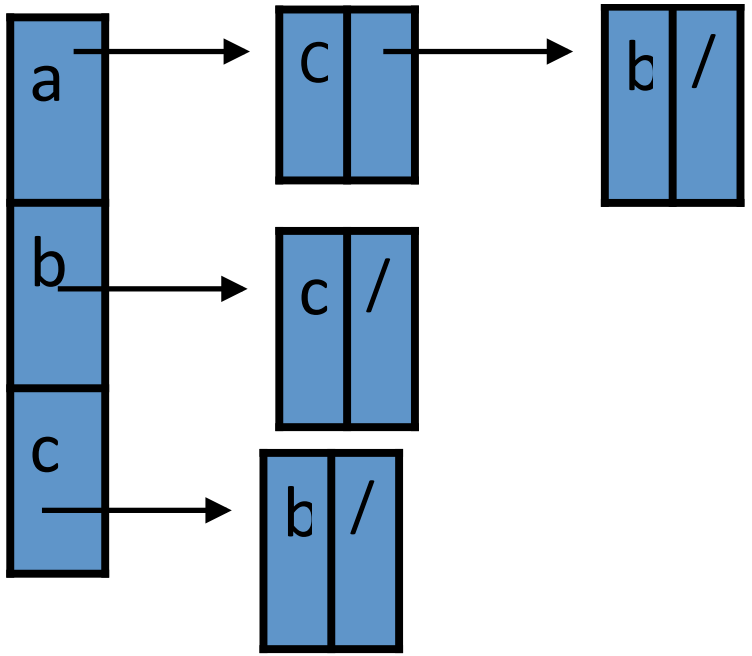
Graph Power!

Computer Representation

Four representations with pros/cons

- *Adjacency lists* (of neighbors of each vertex)

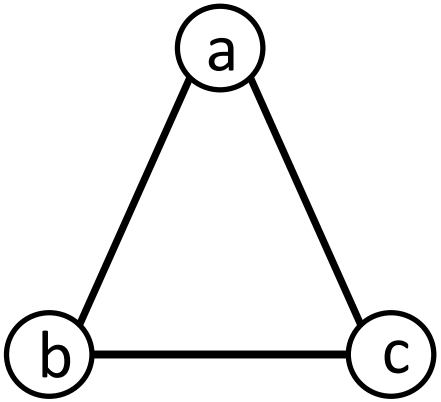


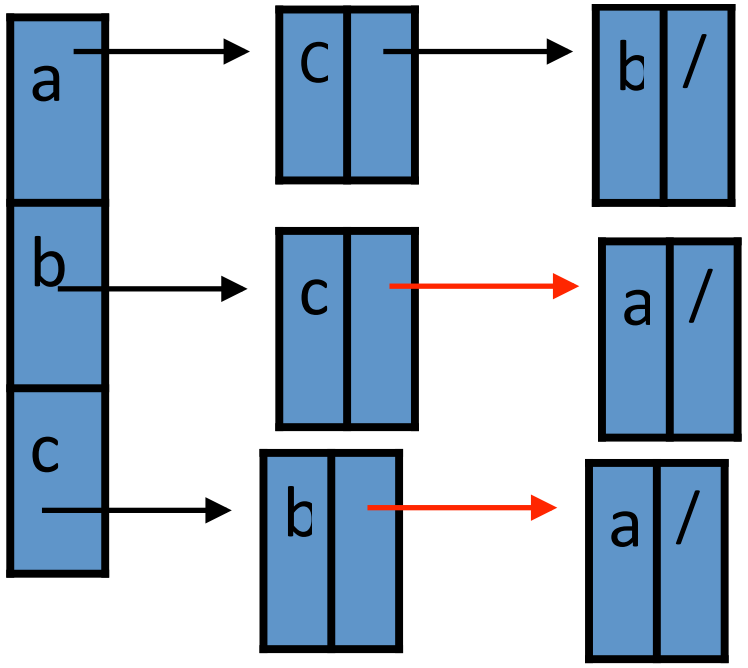


Computer Representation

Four representations with pros/cons

- *Adjacency lists* (of neighbors of each vertex)





Searching Graphs

Finding all vertices connected to a given vertex s WLOG $s = 1$

Class: Undirected Graphs
Recitation: Directed Ones

- s connected to t if there is a path P from s to t
- $P \equiv s = s_0, s_1, \ldots, s_k = t$ such that $\{s_i, s_{i+1}\} \in E$
- Length of $P = k$ (we count edges!)
- Distance between s and t = length of shortest path from s to t

Plan

$Pseudo^2Code$	first	for "mathematical" correctness
Pseudo Code	next	(implementation ideas) for complexity
Real Code	home	for getting an output!

Breadth First Search (Wrong)

All vertices initially unmarked, but s

1. Until all vertices are marked, mark all neighbors of currently marked vertices

Breadth First Search ($Pseudo^2$)

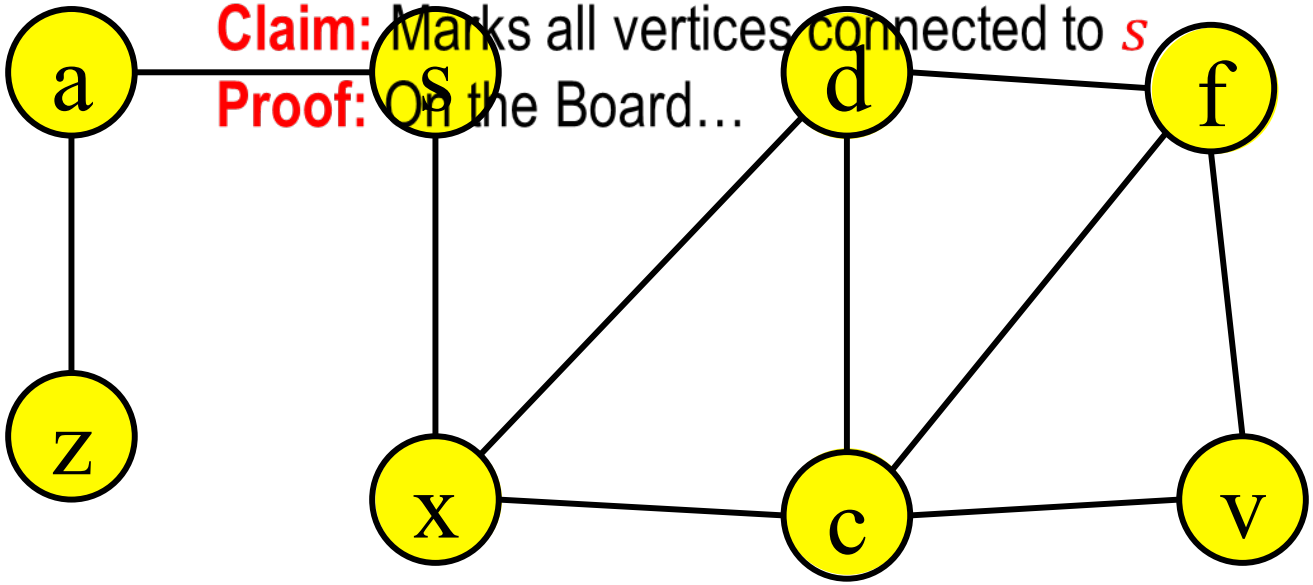
All vertices initially unmarked, but s

1. Until no new vertices are marked, mark all neighbors of currently marked vertices

Example

Claim: Marks all vertices connected to s

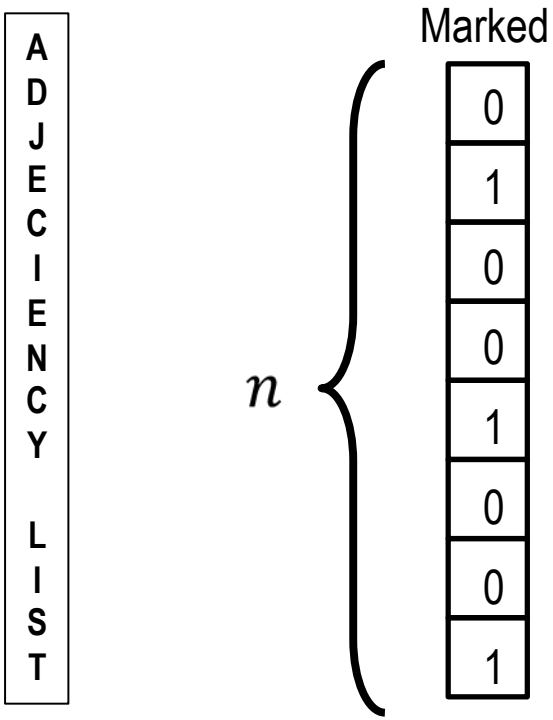
Proof: On the Board…

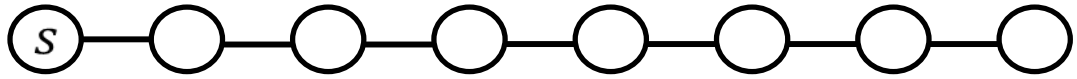


Breadth First Search ($Pseudo^2$)

Complexity?

At least: $Pseudo\ Code$ or Implementation Details!





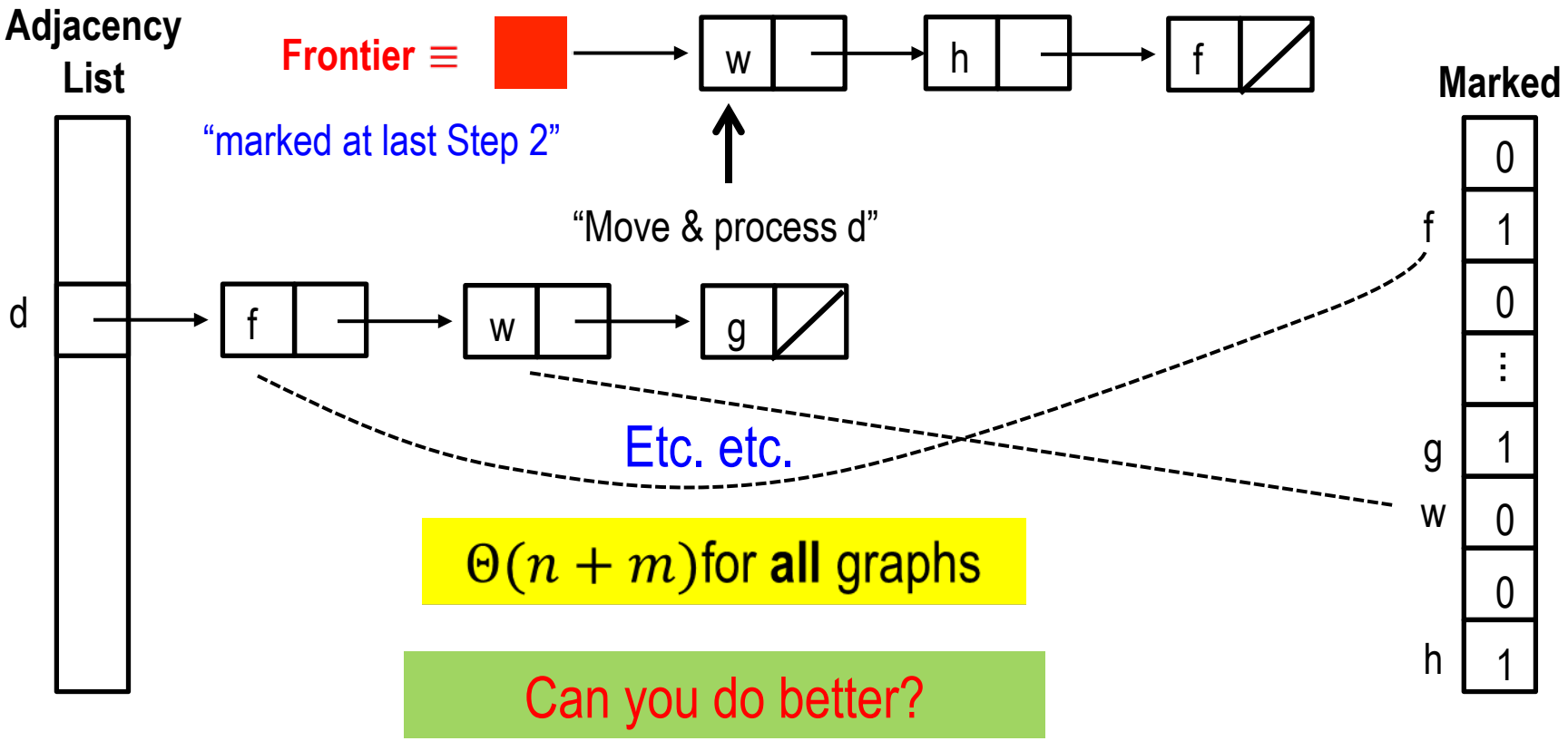
$\Theta(n^2)$ for such graphs

Can it get worse?

Breadth First Search (Better *Pseudo* Code)

All vertices initially unmarked, but s

1. Until no new vertices are marked, mark all neighbors of currently marked vertices



Augmented Breadth First Search =Shortest Path Alg

$(Pseudo^2)$

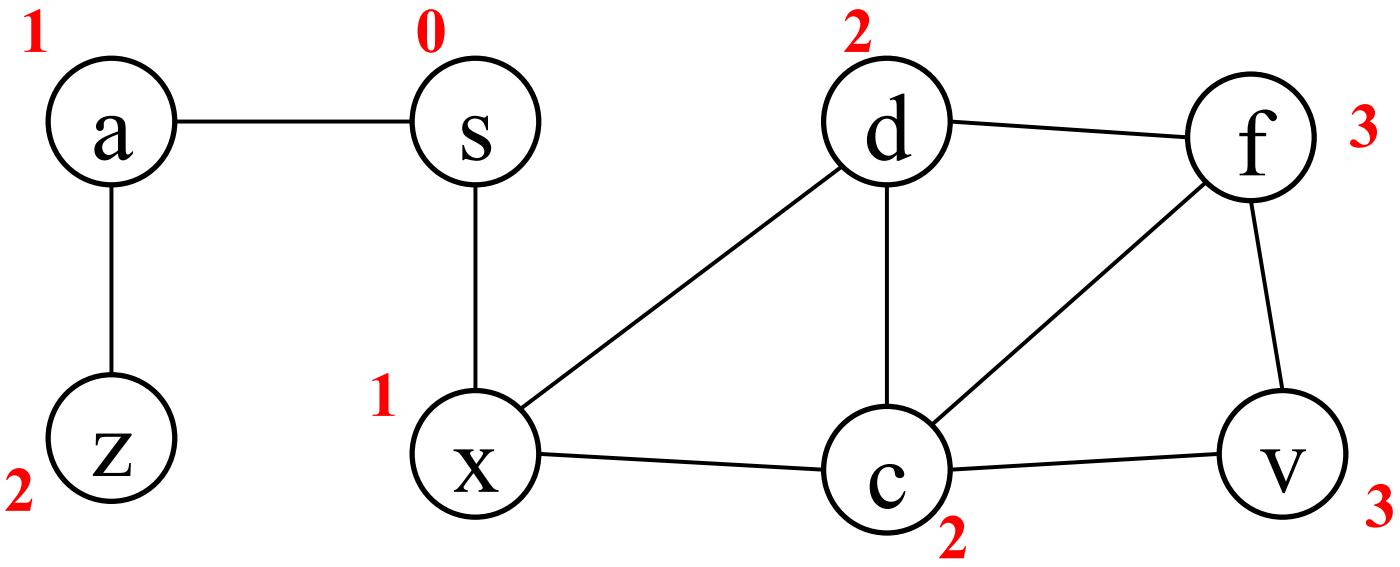
Initially, s is marked 0, all other vertices are marked ∞

1. $i \leftarrow 0$
2. Find all neighbors of at least one vertex marked i. If none, STOP.
3. Mark all vertices found in (3) with $i + 1$.
4. $i \leftarrow i + 1$

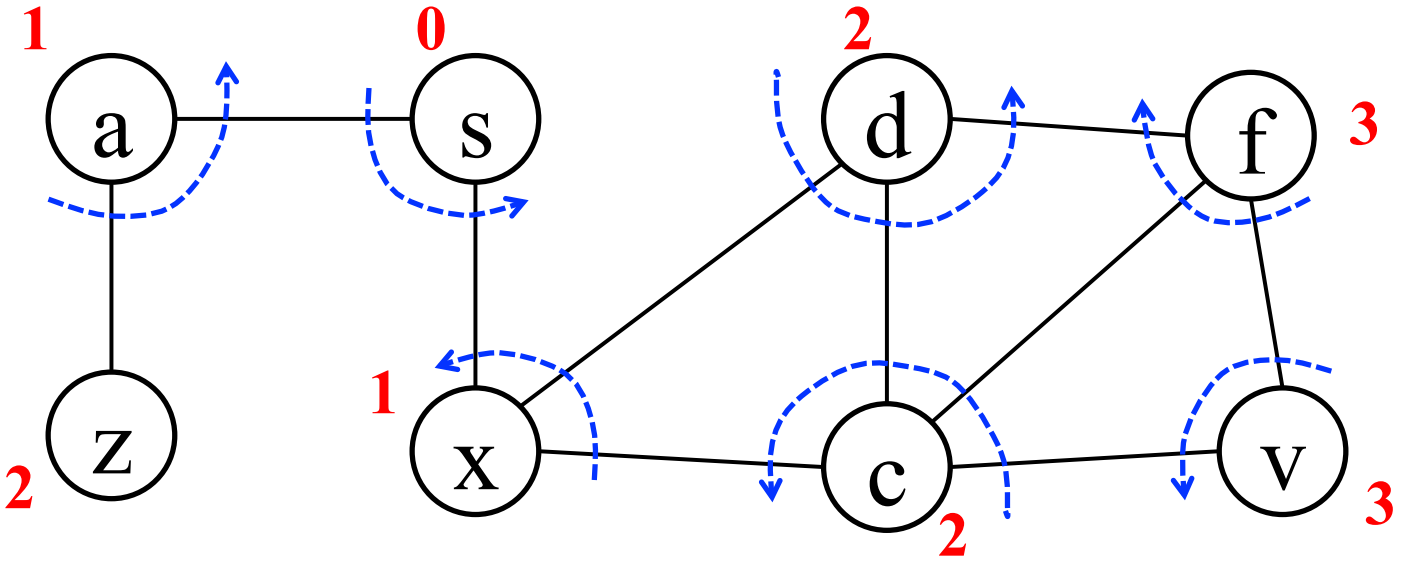
Claim: Every vertex is marked with its distance form s
Proof: ...

Complexity: ...

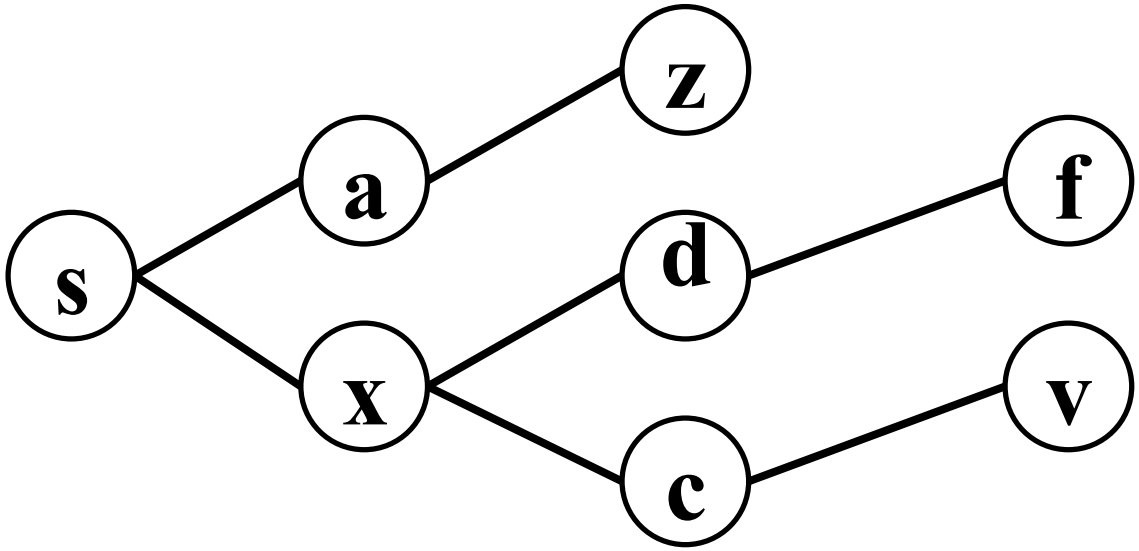
Example ($Pseudo^2$)



Example (*Pseudo*)



If you keep track of the edge through which you got your mark: **BFS Tree!**

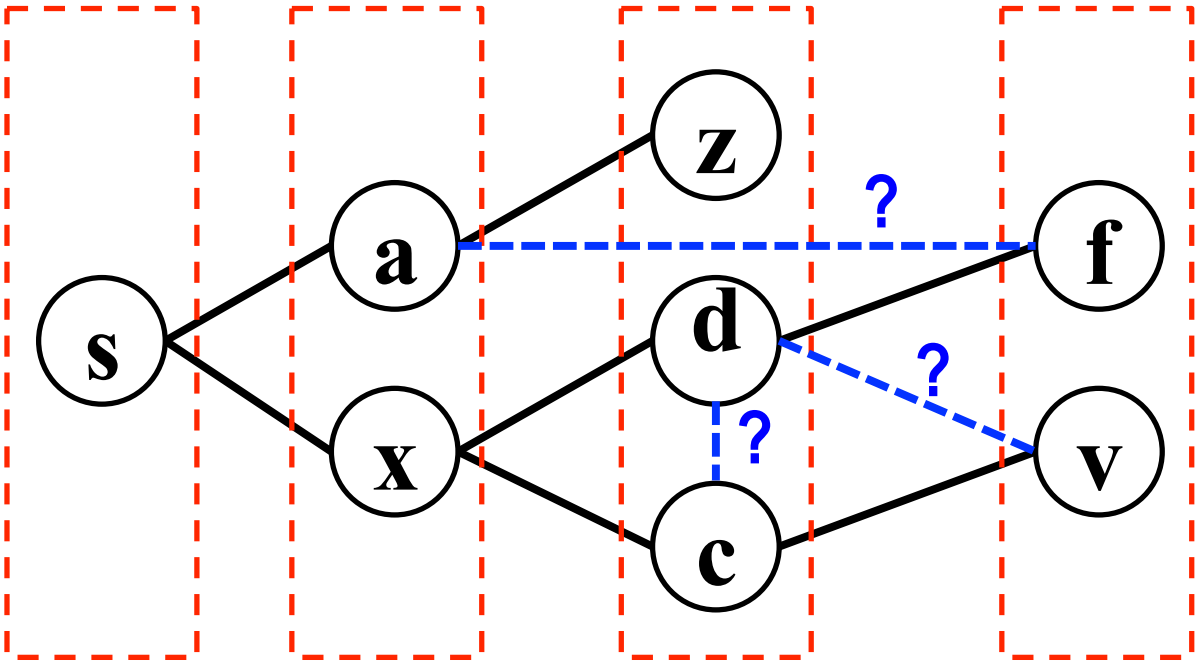


BFS Tree Structure

- Spanning Tree: subgraph (V', E') that
  1. is a tree
  2. $V' = V$

(V', E') subgraph of (V, E) iff $V' \subset V$ and $E' \subset E$

- Extra Structure:



Nooooooo!!!!

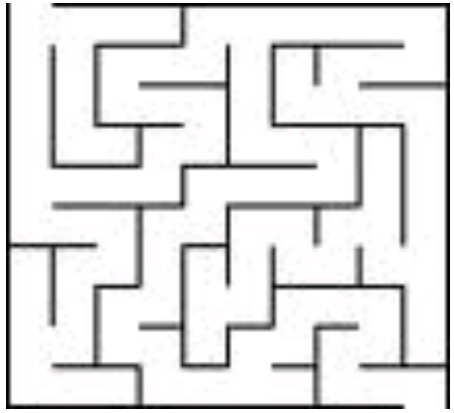
Possible!

Possible Too

BFS Tree: Few Data, Very Informative!

Note!

Graphs model mazes.
But: Searching Graphs ≠ Searching Mazes



1800s Depth First Search
$(Pseudo^3)$

In spocken Englsh (sort of…)

How to visit the Louvre in an hour and come out ALIVE!

No strings allowed!

(No questions either!)

Claim 1: No edge is traversed twice in the same direction

Claim 2: Upon Termination each edge has been traversed once in each direction

Hopcroft's & Tarjan's DFS

- Mark edges rather than their ``entrances" and "exits"
- Number vertices (augmentation for future use)
- Remember your father node rather than the edge who discovered you

0. Mark all edges "unused". For all $v \in V, \#(v) := 0$. Let $i := 0$ and $CoA := s$.

1. $i \leftarrow i + 1$ $\#(CoA) \leftarrow i$

2. If CoA has no unused edges, go to (4)

3. Choose an unused edge $CoA \overset{e}{\leftrightarrow} u$. Mark e used. If $\#(u) \neq 0$ go to (2). Else $F(u) \leftarrow CoA$ $CoA \leftarrow u$ and go to (1)

4. If $\#(CoA) = 1$ HALT

5. $CoA \leftarrow F(CoA)$ and go to (2)

Thm: DFS Visits all vertices connected to s

Proof: …

DFS Tree

- Tree edges
- Back edges

Odds & Ends

- Queues vs. Stacks
- Strings??!

Good News

More Board Explanations!

Good Implications

More Reasons to come to class!

Enjoy it!

Lecture is over!

Please walk **calmly** to the nearest EXIT