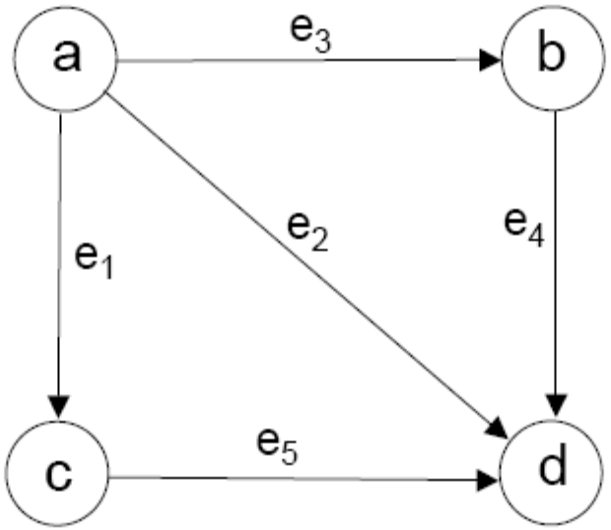
Breadth-First Search

Note Title 11/15/2007

First we'll overview the graph representations.

Then we'll look at problem of finding a shortest path in a directed unweighted graph

Adjacency List



vertices $\{a, b, c, d\}$

outedges

$a \rightarrow \{e_3, e_1, e_2\}$

$b \rightarrow \{e_4\}$

$c \rightarrow \{e_5\}$

$d \rightarrow \{\}$

inedges

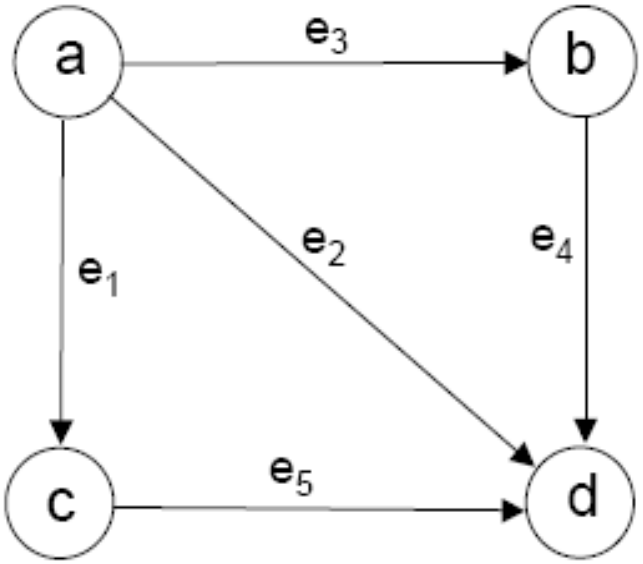
$a \rightarrow \{\}$ $b \rightarrow \{e_3\}$

$c \rightarrow \{e_1\}$ $d \rightarrow \{e_2, e_4, e_5\}$

we've called Augmented Adj List

Data Structure	*storeIncomingEdges*	TaggedBucketCollection type
AdjacencyList	*false*	$V \rightarrow$ List<E>
AugmentedAdjacencyList	*true*	$V \rightarrow$ List<E>
Adjacency Set (no multi-edges)	*false*	$V \rightarrow$ Set<E>
Augmented Adjacency Set (no multi-edges)	*true*	$V \rightarrow$ Set<E>
Adjacency Set (with multi-edges)	*false*	$V \rightarrow$ BucketMapping <V, List <E>>
Augmented Adjacency Set (with multi-edges)	*true*	$V \rightarrow$ BucketMapping <V, List <E>>

Adjacency matrix



ids

a → 0

c → 2

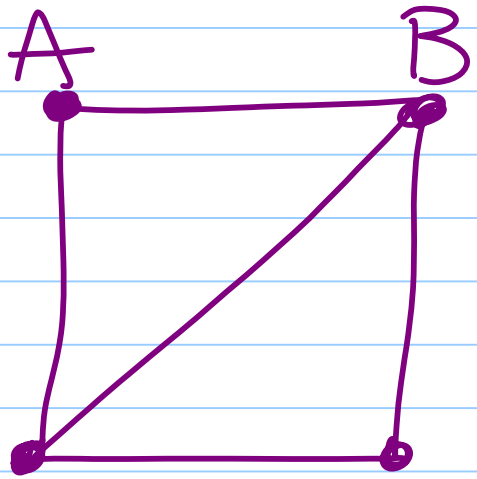
d → 3

b → 1

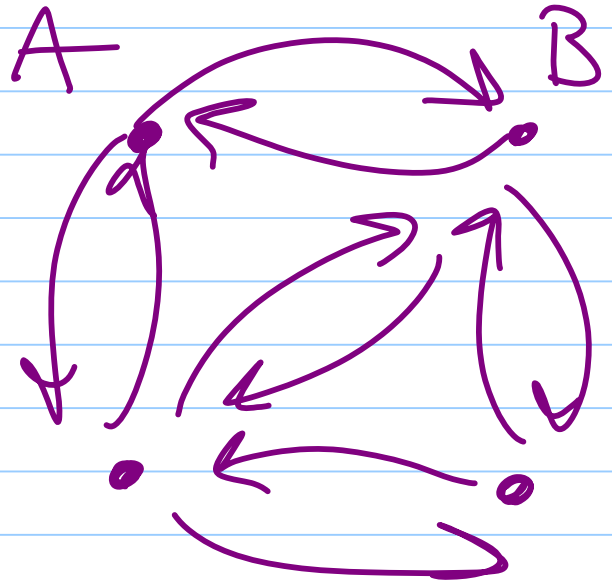
edges

	0	1	2	3
0	ø	ø	e_1	e_2
1	ø	ø	ø	ø
2	ø	ø	ø	e_5
3	ø	ø	ø	ø

Representing Undirected Graph



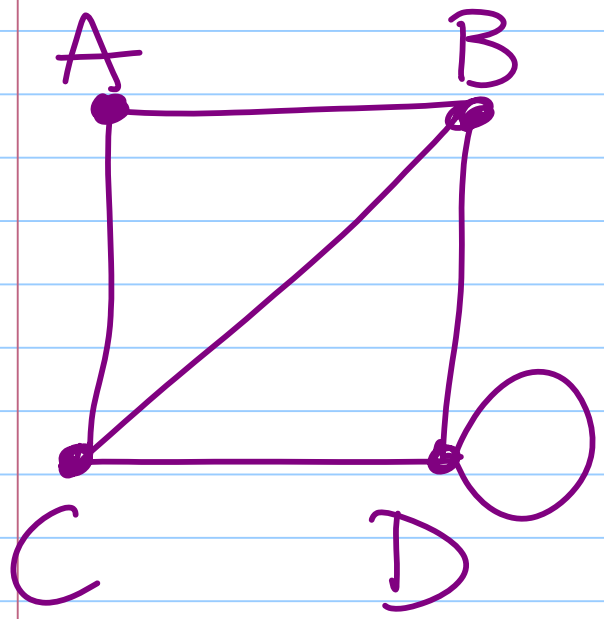
represent as



m edges

2m edges

1 edge 0 no edge



	A	B	C	D
A	0	1	1	0
B	1	0	1	1
C	1	1	0	1
D	0	1	1	1

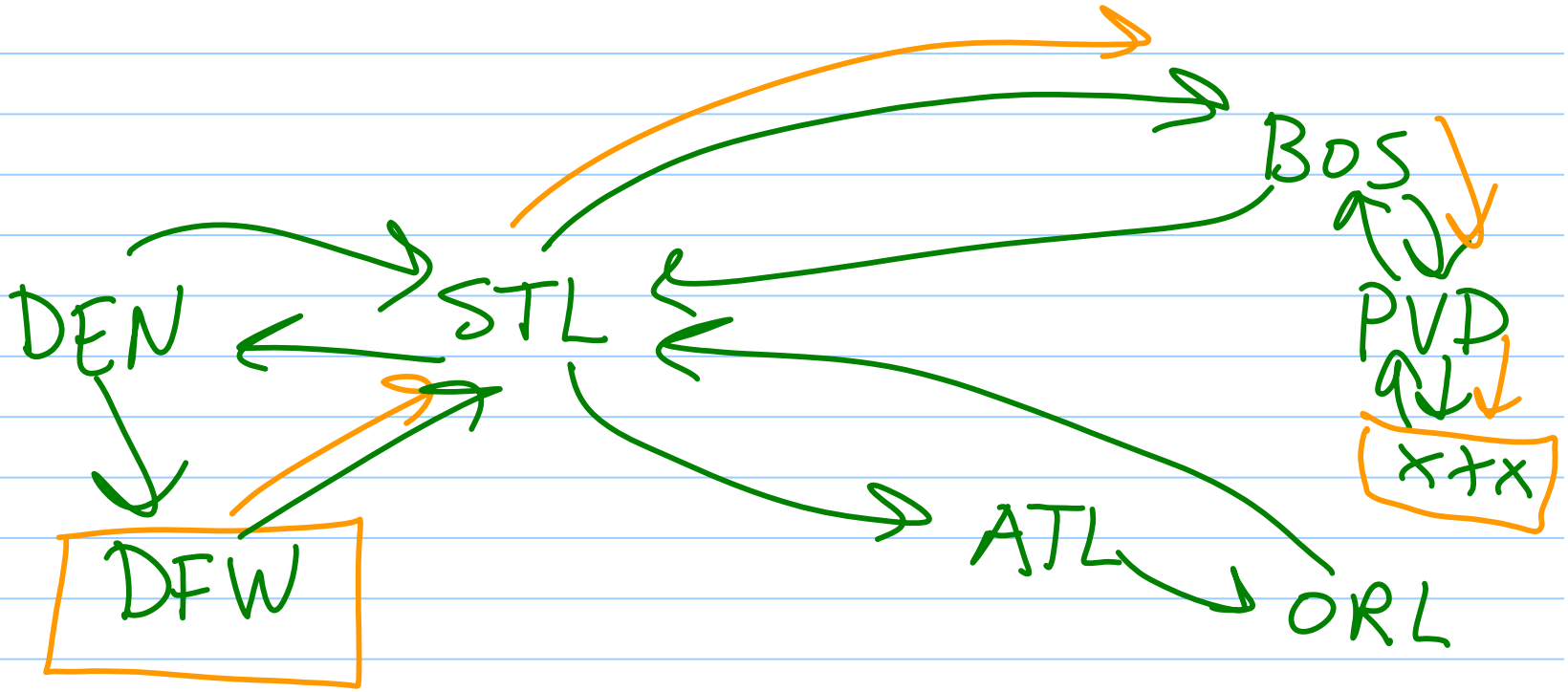
Undirected graph, only need to store this part

Analysis (directed graph, no multi-edges)

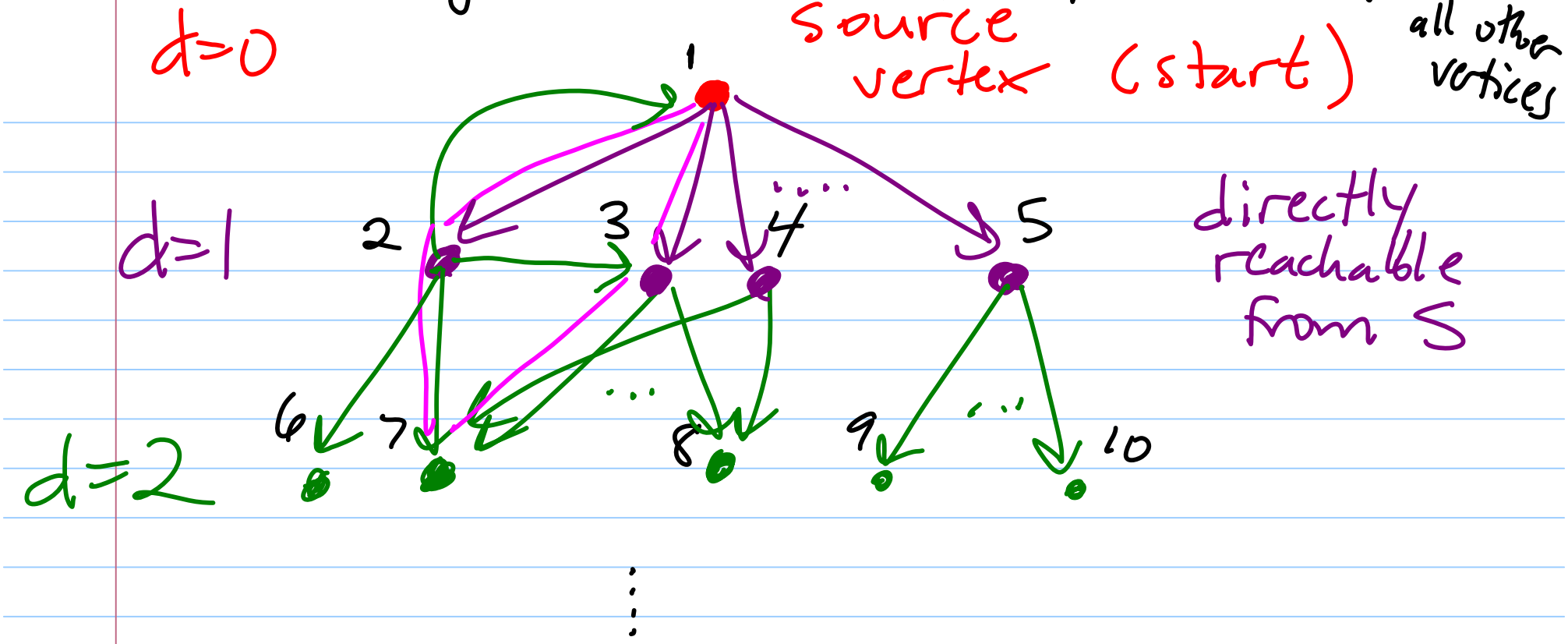
n vertices
m edges

	Adj List	Adj Set	Adj Matrix
Contains edge $v_i v_j$	$O(\#\text{ edges out of } v_i)$	expected $O(1)$	$O(1)$
iterate over out edges v	$O(\#\text{ edges out of } v)$	$O(\#\text{ edges out of } v)$	$O(n)$
iterate over all edges	$O(n+m)$	$O(n+m)$	$O(n^2)$
space	$O(n+m)$	$O(n+m)$	$O(n^2)$

Finding Shortest Paths in Unweighted Directed Graph

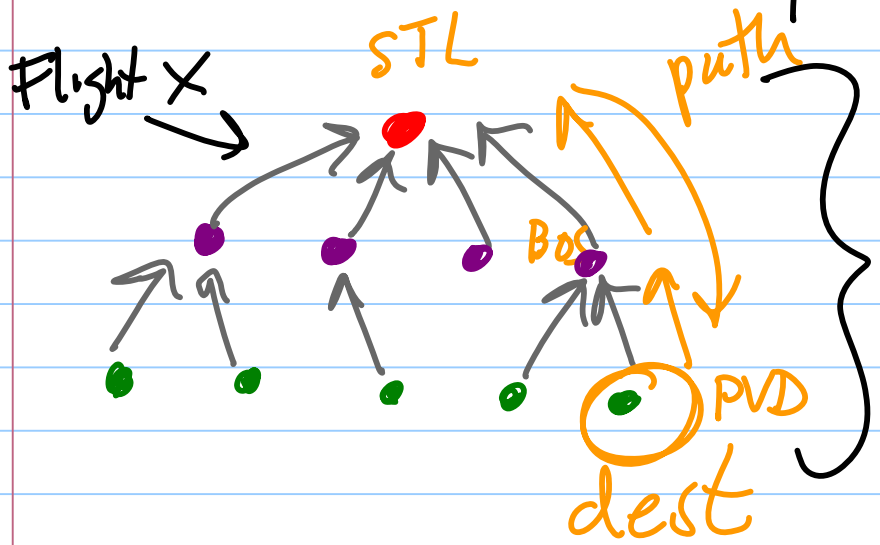


Single source shortest path - find path to all other vertices



Keep going until the desired destination is reached

How do we represent the solution?

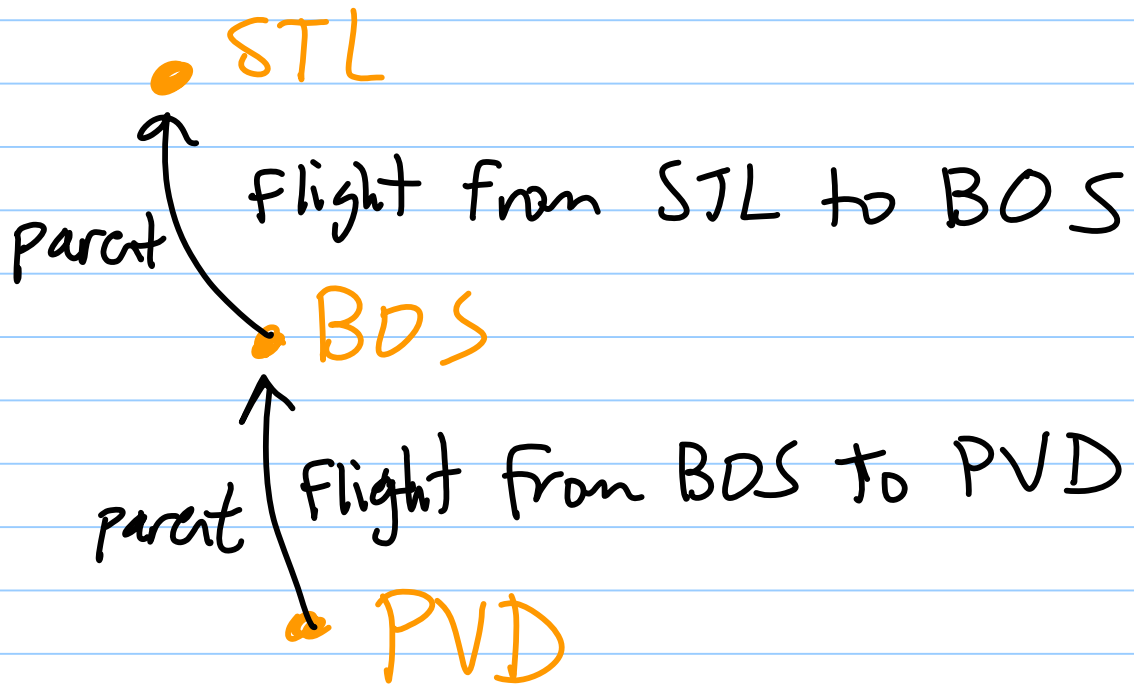


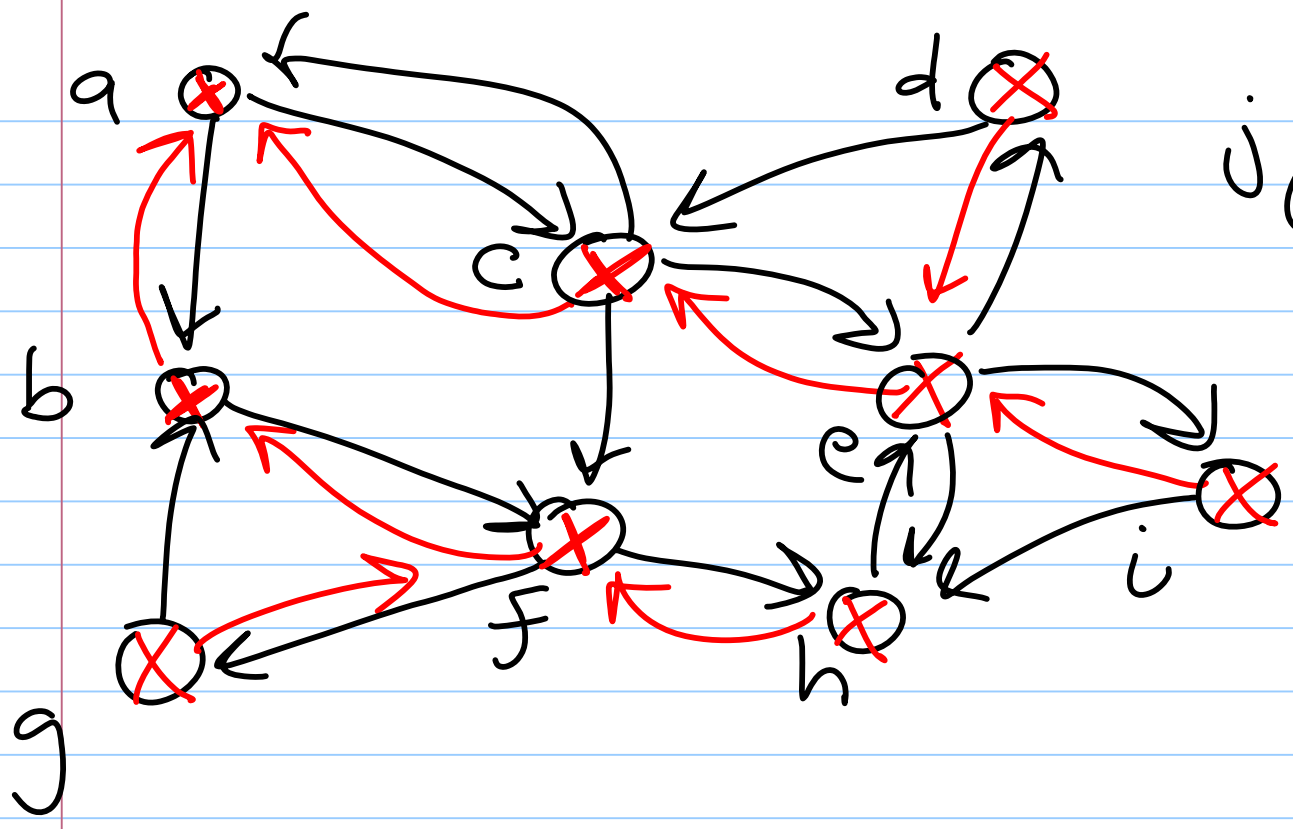
shortest path tree

Source is root

[represent implicitly by storing the edge from parent.

Shortest path tree



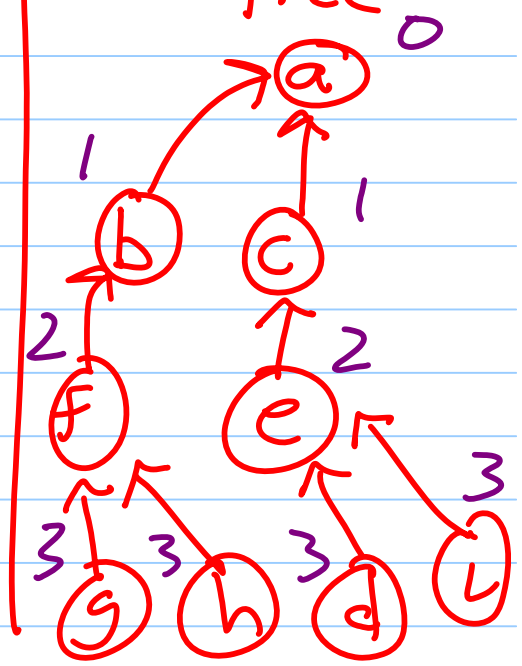


Source a

parent

extra

tree



Queue : ~~a~~ ~~b~~ ~~c~~ ~~d~~ ~~e~~ ~~f~~ ~~g~~ ~~h~~ ~~i~~ ~~j~~

Time complexity

entire adj $O(n)$: every vertex is placed into & removed from queue ≤ 1 time

adj list $O(n+m)$: iterate over outedges of each vertex at most once each

$O(n+m)$ adj list, $O(n^2)$ adj matrix