

# Breadth-First Search

Note Title

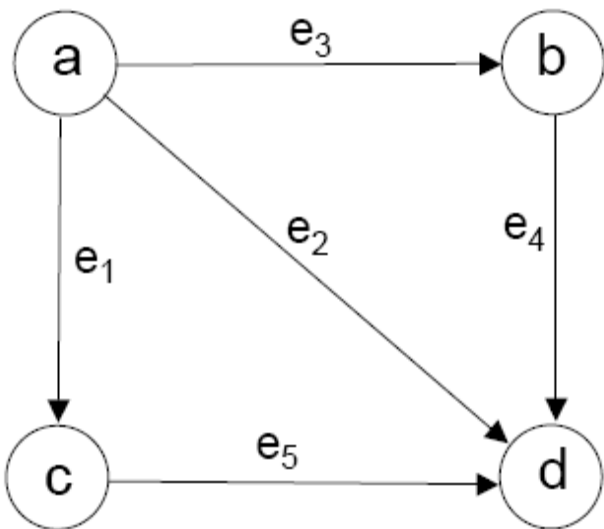
11/15/2007

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

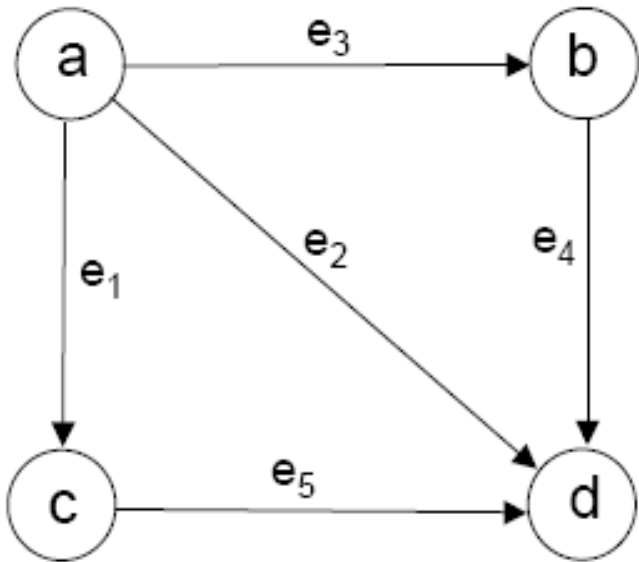
$a \rightarrow \{\}$      $b \rightarrow \{e_3\}$

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

we've called  
Augmented Adj List

Data Structure	<i>storeIncomingEdges</i>	TaggedBucketCollection type
AdjacencyList	<i>false</i>	$V \rightarrow \text{List}\langle E \rangle$
AugmentedAdjacencyList	<i>true</i>	$V \rightarrow \text{List}\langle E \rangle$
Adjacency Set (no multi-edges)	<i>false</i>	$V \rightarrow \text{Set}\langle E \rangle$
Augmented Adjacency Set (no multi-edges)	<i>true</i>	$V \rightarrow \text{Set}\langle E \rangle$
Adjacency Set (with multi-edges)	<i>false</i>	$V \rightarrow \text{BucketMapping} \langle V, \text{List} \langle E \rangle \rangle$
Augmented Adjacency Set (with multi-edges)	<i>true</i>	$V \rightarrow \text{BucketMapping} \langle V, \text{List} \langle E \rangle \rangle$

# Adjacency matrix



ids

a → 0

c → 2

d → 3

b → 1

edges

0 1 2 3

0

∅ ∅ e<sub>1</sub> e<sub>2</sub>

1

∅ ∅ ∅ ∅

2

∅ ∅ ∅ e<sub>5</sub>

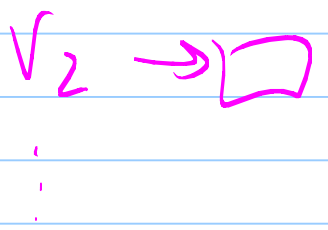
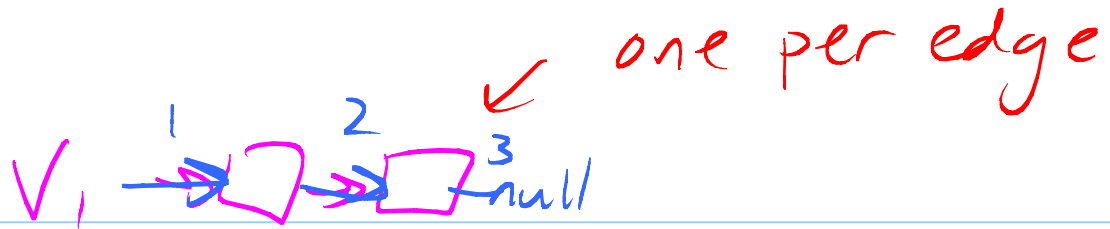
3

∅ ∅ ∅ ∅

# Analysis (directed graph, no multi-edges)

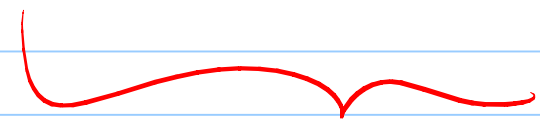
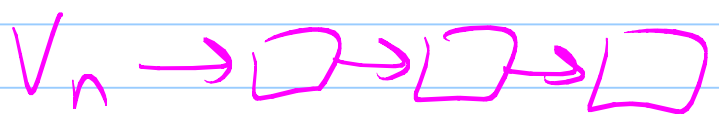
$n$  vertices  
 $m$  edges

	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	$O(n+m)$	$O(n+m)$	$O(n^2)$



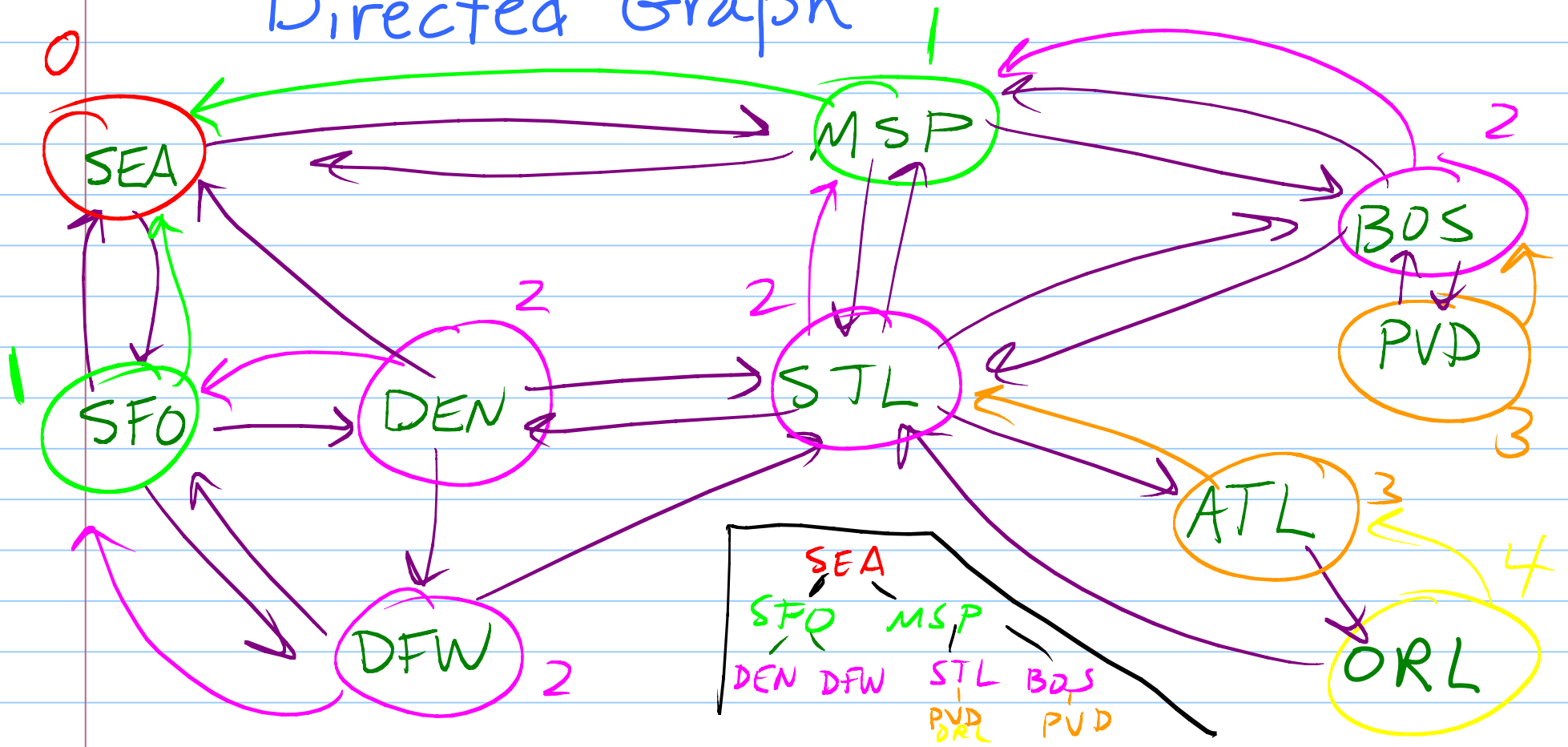
# refs to follow

$$m + n$$



m total edges

# Finding Shortest Paths in Unweighted Directed Graph



# Breadth first search

$O(n+m)$

each vertex store

parent (Edge) — how you were reached from parent

maybe boolean to know if discovered

dist (length of shortest path)

Use a queue to maintain order to process



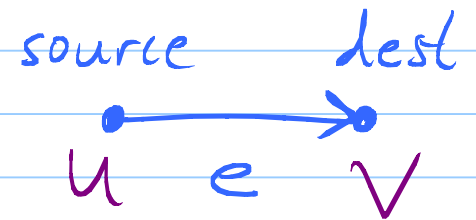




BFs(V source)

```
for each  $u \in V$   
   $u.\text{discovered} = \text{false}$   
   $u.\text{dist} = \infty$   
   $u.\text{parentEdge} = \text{null}$   
 $\text{source}.\text{discovered} = \text{true}$   
 $\text{source}.\text{dist} = 0$   
 $\text{queue}.\text{enqueue}(\text{source})$ 
```

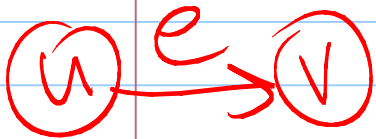
initialization

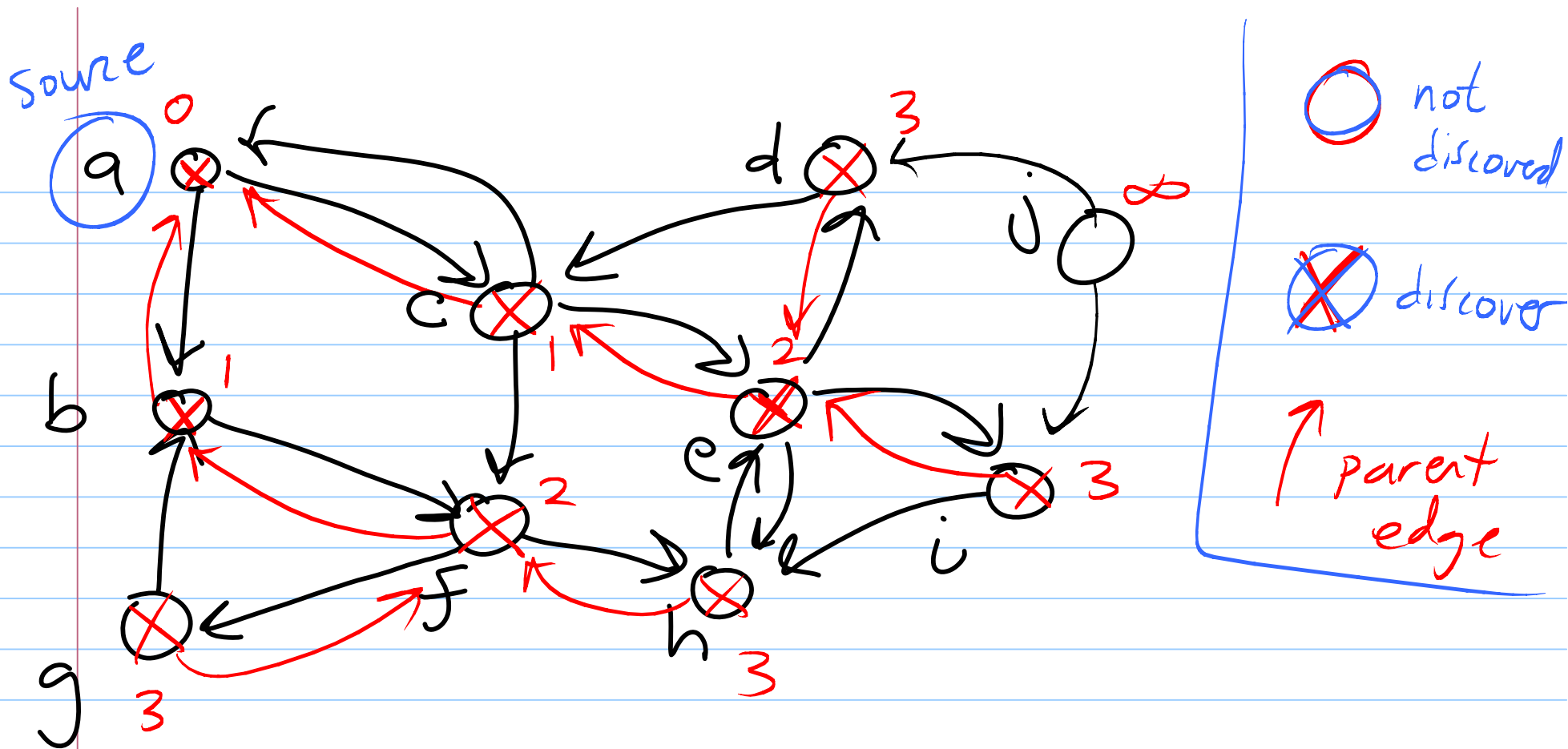


```
while (!queue.isEmpty())  
   $u = \text{q}.\text{dequeue}()$ 
```

```
  for each edge  $e$  in outgoing edges from  $u$   
     $v = e.\text{dest}$ 
```

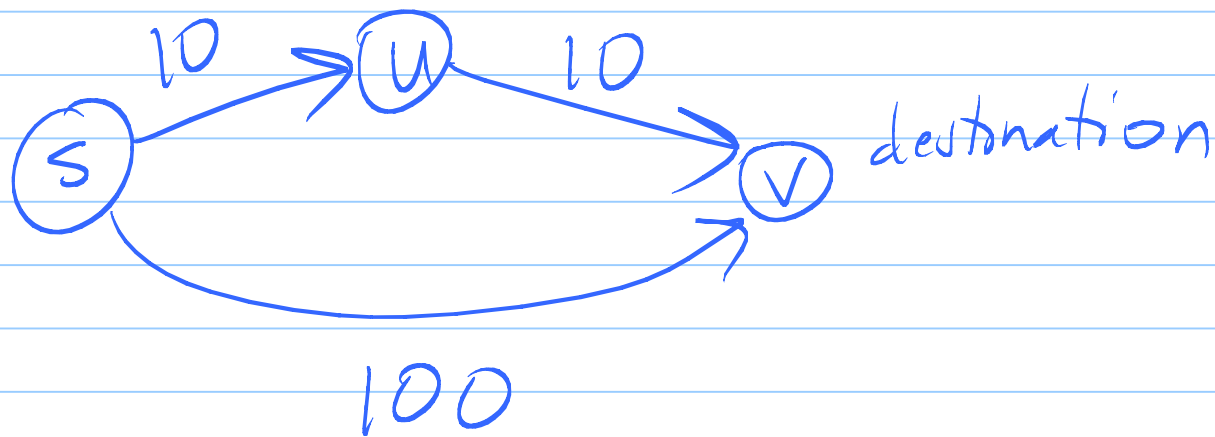
```
    if (! $v.\text{discovered}$ )  
       $v.\text{discovered} = \text{true}$   
       $v.\text{dist} = u.\text{dist} + 1$   
       $v.\text{parentEdge} = e$   
       $\text{queue}.\text{enqueue}(v)$ 
```





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

What if the edges are weighted?



Assume:

No negative weight edges

$$2^{\log(a \times b \times c \times d)} = a \times b \times c \times d$$

$$\log(a \times b \times c \times d) = \log a + \log b + \log c + \log d$$

