

Graphs

Note Title

11/13/2007

Set (accessed
by final currency)

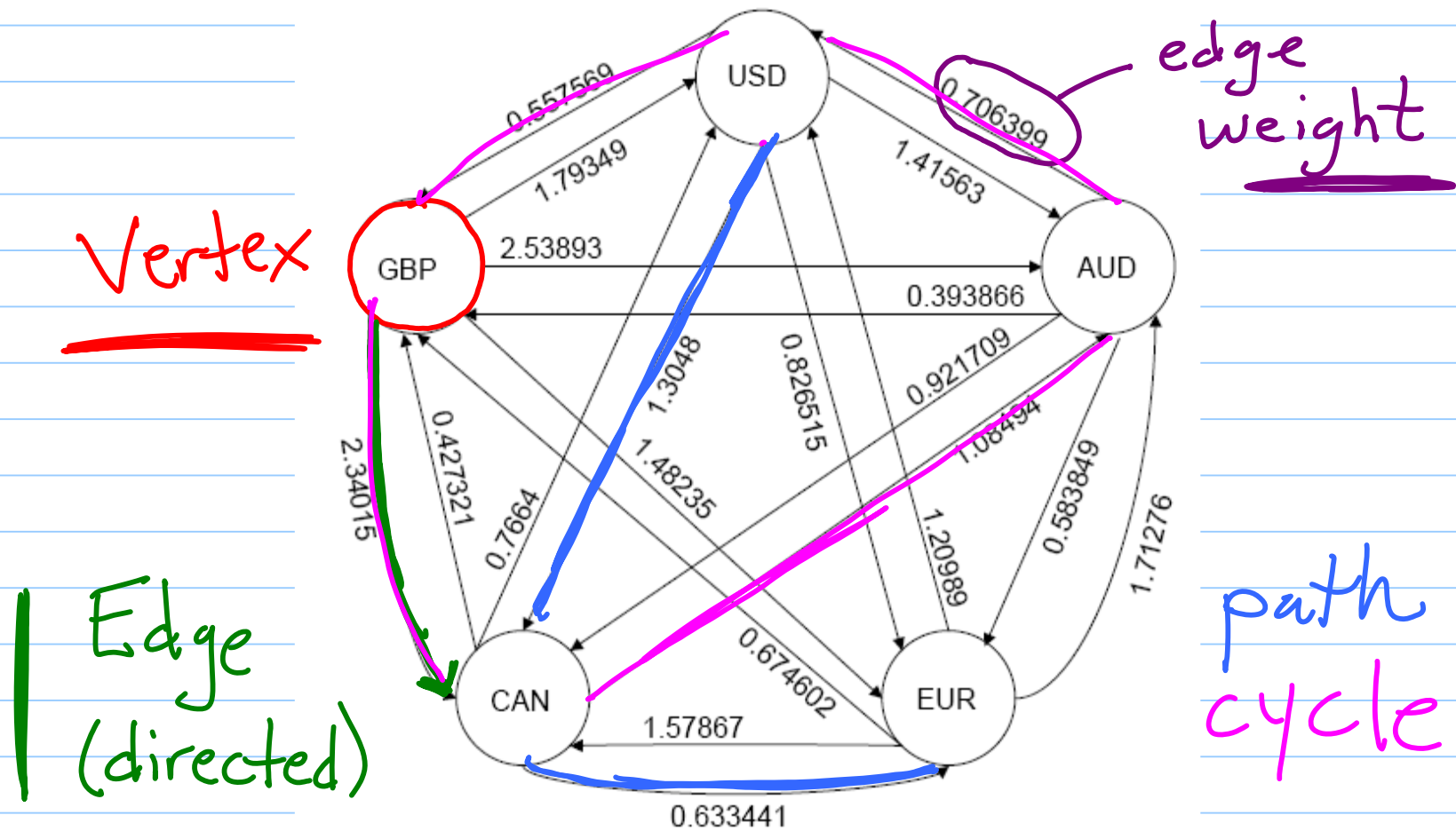
	USD	GBP	CAD	EUR	AUD
USD	1	0.557569	1.3048	0.826515	1.41563
GBP	1.79349	1	2.34015	1.48235	2.53893
CAD	0.7764	0.427321	1	0.633441	1.08494
EUR	1.20989	0.674602	1.57867	1	1.71276
AUD	0.706399	0.393866	0.921709	0.583849	1

exchange
rates in
August 2004

What data structure is best if you just want to look up an exchange rate?

What if you want to determine if an arbitrage scheme exists?

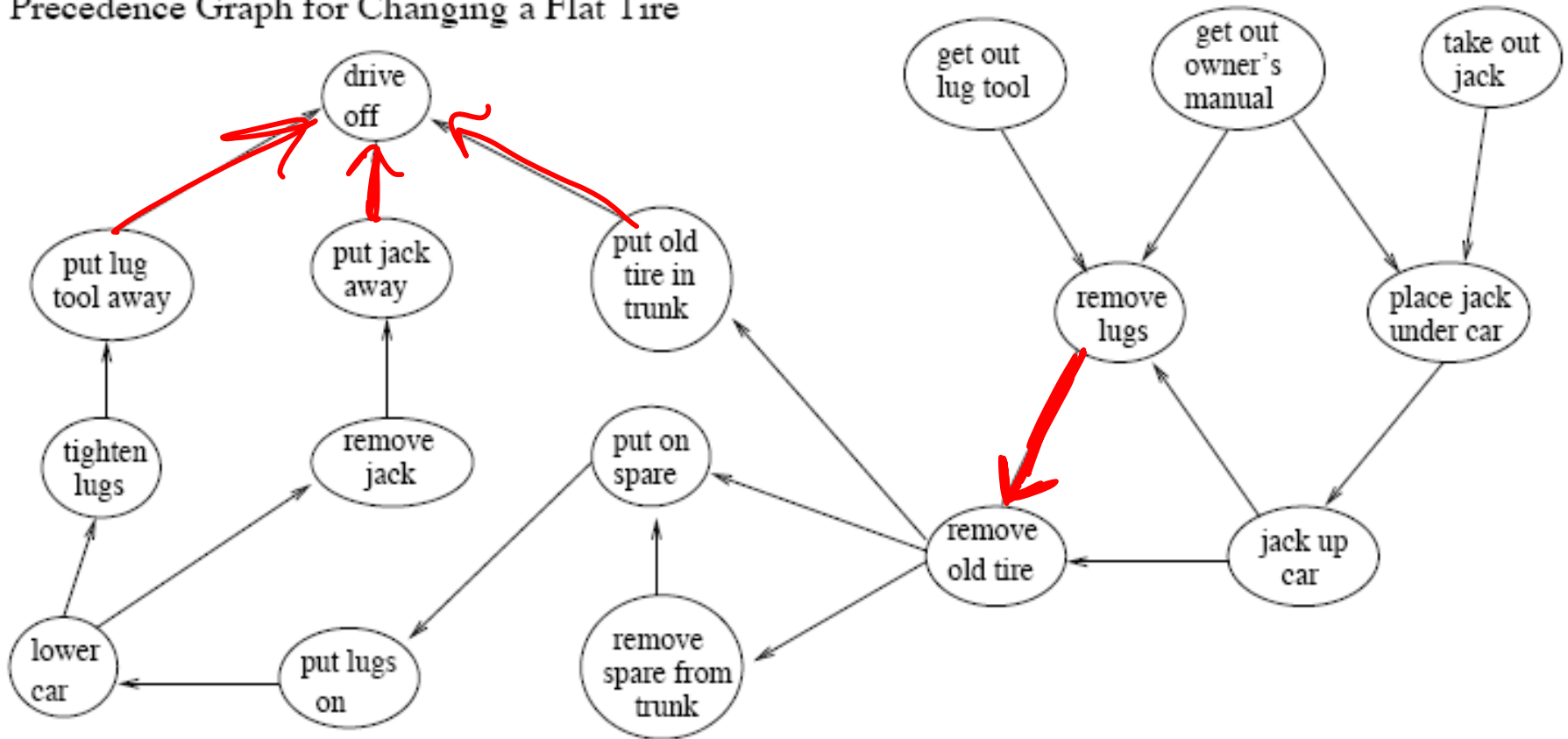
Graph Representing Exchange Rates



Task Scheduling

$a \rightarrow b$
a must precede b

Precedence Graph for Changing a Flat Tire

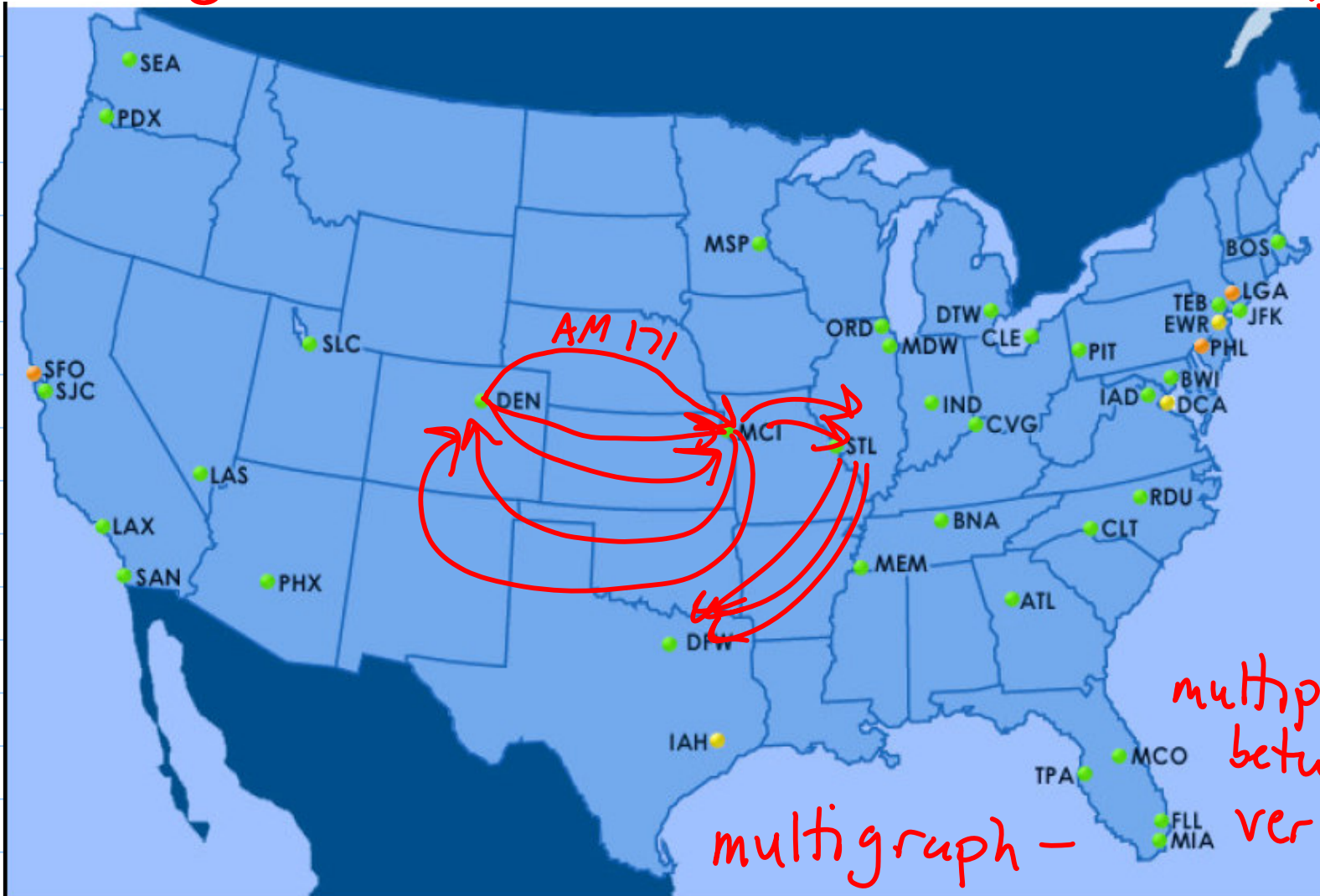


Finding Shortest Travel Routes

#edges
=#flights

each
airport
is
a
vertex

sample
edges
(flight)



multiple edges
between
vertices

multigraph -

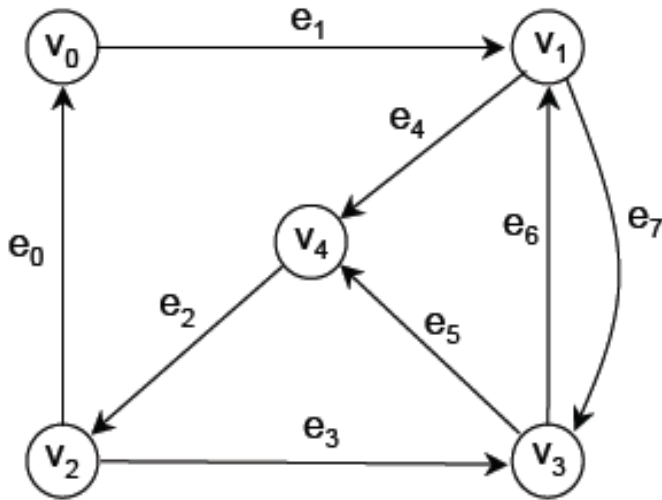
and many more

- image segmentation
 - minimizing infrastructure cost (e.g. laying optical fiber) to allow travel/communication between a set of locations
 - executing a makefile
 -
 -
- A lot of problems can be formulated as graph problems

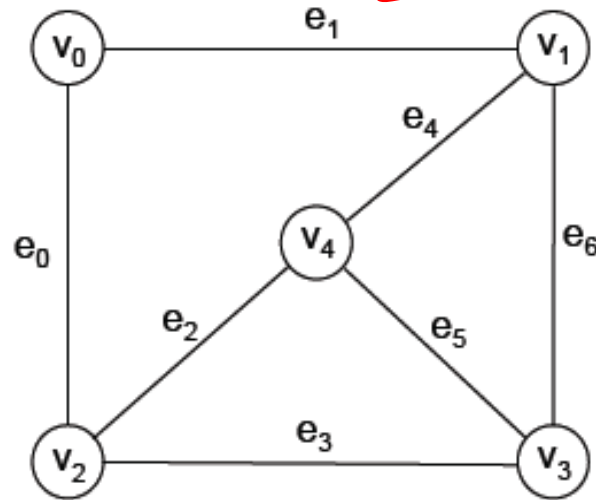
Types of graphs

unweighted

name of edge e

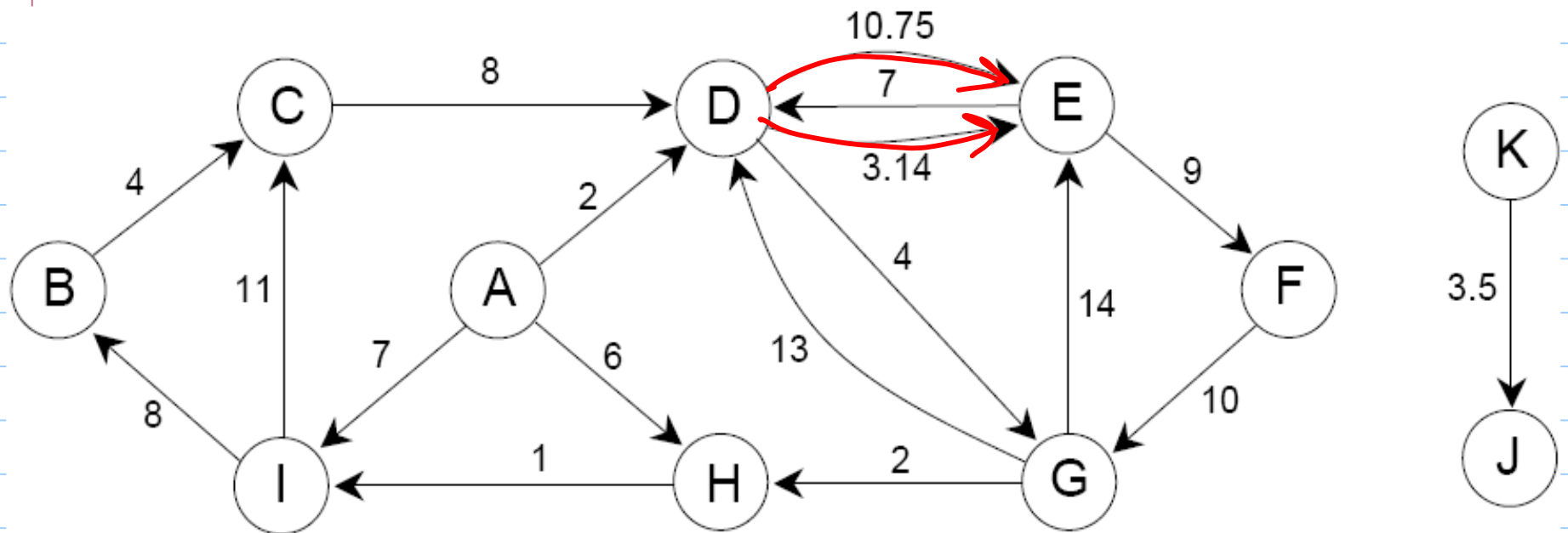


directed



undirected

Weighted directed multigraph



How can we represent a graph?

What are basic things we might want to do?

iterate
over
all
such
edges
in a
multigraph

Is there an edge from V_i to V_j ?

Iterate over all edges from V_i ? ✓

Less often, iterate over all edges to V_u ? ✓

$V =$ set of vertices in graph

For Lab 4

STL \rightarrow airport
~~~~~  
object

3-letter  
acronym for  
airport

Set of airports

Object "STL"

city name St. Louis

time zone

location

⋮

could use  
a Set  
versus a  
list.

Expected constant  
time search by dest.

provide an  
comparator to constructor

List of edges (Flights) that  
originate in St. Louis

list of  
outgoing  
edges

Could also keep a list of edges that  
terminate in St. Louis

# Adjacency List

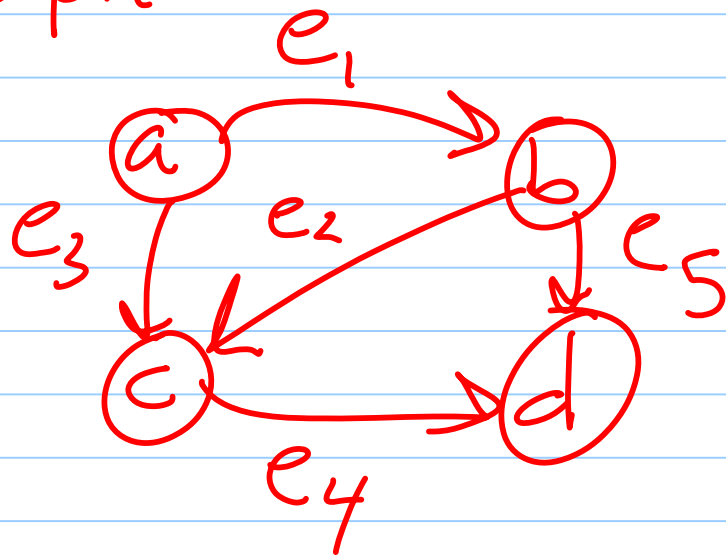
Representation of a graph  
where for each vertex you

store a list of all

outgoing edges

adjacent (in a directed sense)

Graph



Typical way to draw

a : ~~b~~, ~~c~~<sup>e<sub>1</sub>, e<sub>3</sub></sup>  
b : ~~c~~, ~~e<sub>2</sub>~~<sup>e<sub>5</sub></sup>  
c : ~~e<sub>4</sub>~~  
d :



implicit representation  
of an edge when  
no multi-edges

incoming edges

also use edges here

|   |     |      |
|---|-----|------|
| ( | a : |      |
|   | b : | a    |
|   | c : | a, b |
|   | d : | b, c |

## bucket Mapping for multigraph

If we keep a set of outgoing edges for each vertex (with comparison defined by dest)

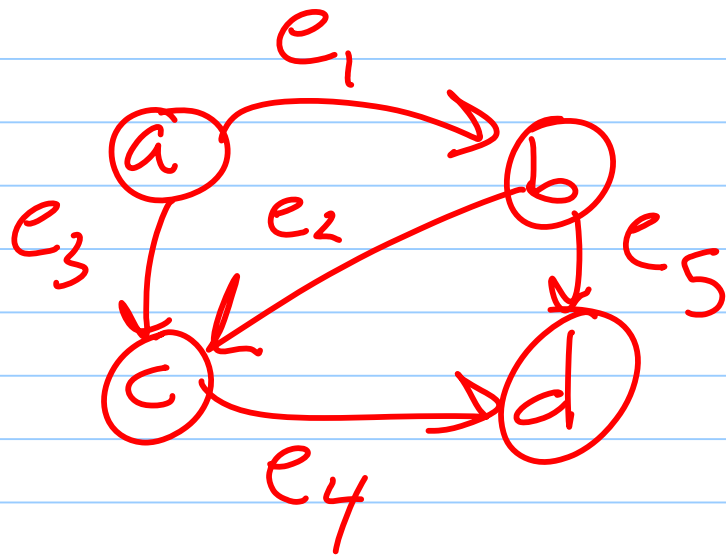
answer is edge from  $V_i$  to  $V_j$   
in expected constant time.

## Adjacency Set

Augmented Adjacency Set also keep  
a set of incoming edges for each vertex

# Adjacency Matrix

x null  
mapping  $\begin{cases} a \rightarrow 0, b \rightarrow 1 \\ c \rightarrow 2, d \rightarrow 3 \end{cases}$



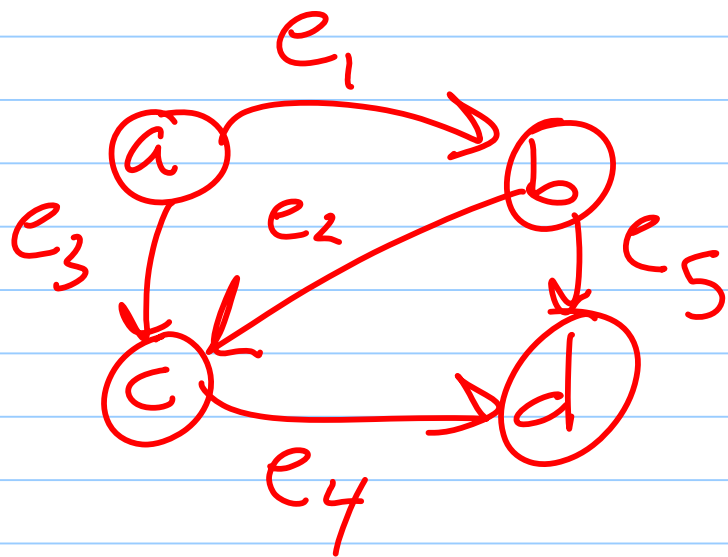
|   | a | b              | c              | d              |
|---|---|----------------|----------------|----------------|
| a | x | e <sub>1</sub> | e <sub>3</sub> | x              |
| b | x | x              | e <sub>2</sub> | e <sub>5</sub> |
| c | x | x              | x              | e <sub>4</sub> |
| d | x | x              | x              | x              |

$O(n^2)$  time to iterate over

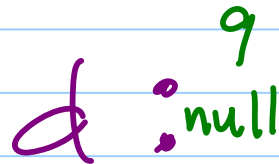
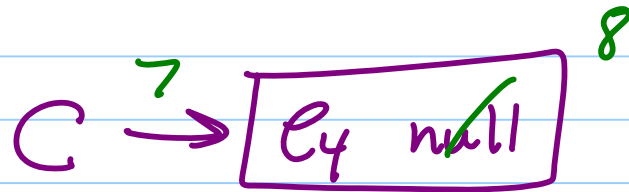
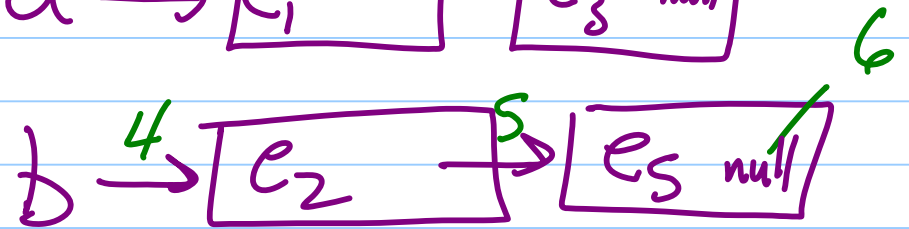
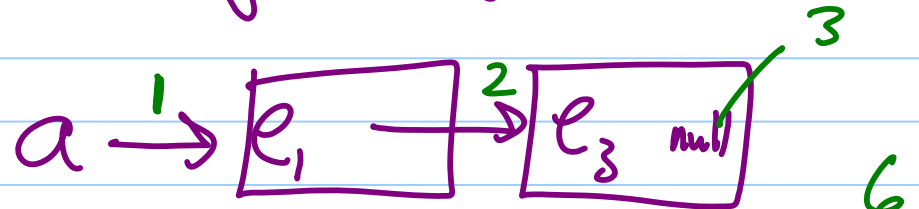
$n$  vertices  
 $m$  edges

all edges  
(if not multigraph)

$n \times n$  matrix



adj list



⏟  
# objects among all lists is m

$O(m+n)$  time  
to iterate over all edges

Next time

Summarize time & space  
complexities

Talk about rep. undirected  
graph