Graphs

Information Technology University

Discrete Structures
February 16, 2026

Graph Basics, Definitions, Terminology.

Why Graphs

- Social Networks: analysis of patterns of connectivity and the identification of key influencers or communities.

Why Graphs

- Social Networks: analysis of patterns of connectivity and the identification of key influencers or communities.
- Transportation networks: patterns of usage, the identification of bottlenecks, and the design of efficient routing schemes.

Why Graphs

- Social Networks: analysis of patterns of connectivity and the identification of key influencers or communities.
- Transportation networks: patterns of usage, the identification of bottlenecks, and the design of efficient routing schemes.
- The World Wide Web: information flow and the identification of important or central parts of web.

Why Graphs

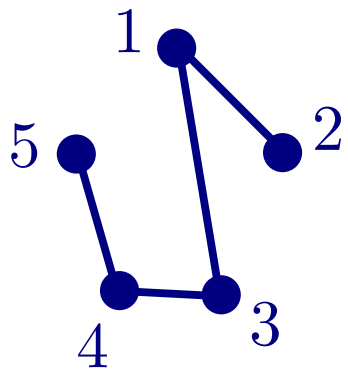
- Social Networks: analysis of patterns of connectivity and the identification of key influencers or communities.
- Transportation networks: patterns of usage, the identification of bottlenecks, and the design of efficient routing schemes.
- The World Wide Web: information flow and the identification of important or central parts of web.
- Biological networks: interactions between different molecules, genes or proteins, in order to understand the biological systems and pathways.

GRAPH:

GRAPH: a list of pairs of "things"

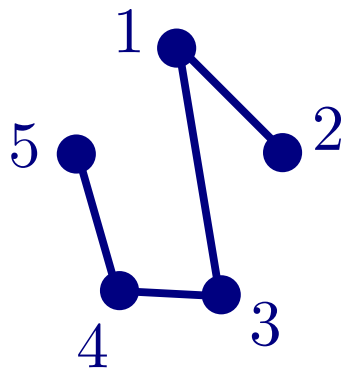
GRAPH: a list of pairs of "things"

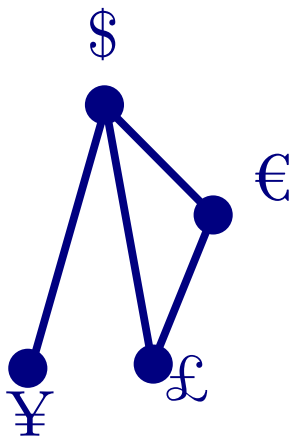
$$A = \{(1,2),(1,3),(3,4),(4,5)\}$$



GRAPH: a list of pairs of "things"

$$A = \{(1,2),(1,3),(3,4),(4,5)\}$$
$$B = \{(\$,€),(£,€),(\$,¥),(\$,£)\}$$



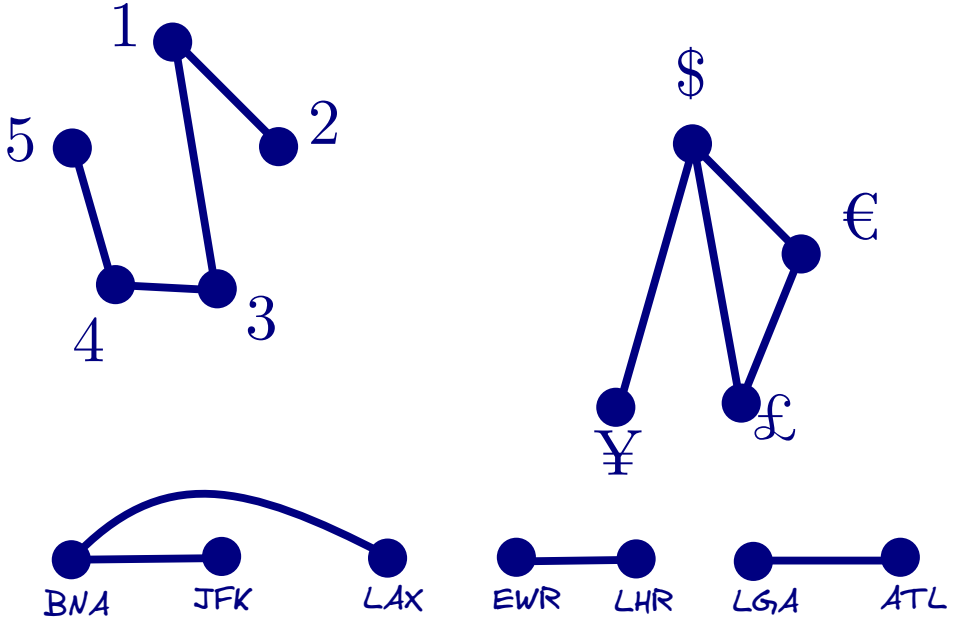


GRAPH: a list of pairs of "things"

$$A = \{(1,2),(1,3),(3,4),(4,5)\}$$

$$B = \{(\$,€),(£,€),(\$,¥),(\$,£)\}$$

$$C = \{(\text{JFK},\text{BNA}),\ (\text{BNA},\text{LAX}),\ (\text{EWR},\text{LHR}),\ (\text{LGA},\text{ATL})\}$$



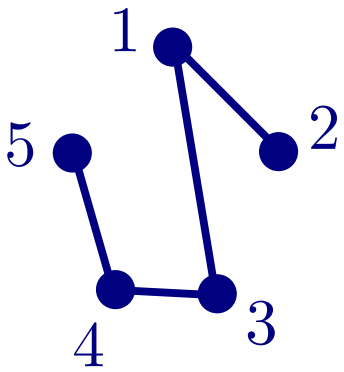
GRAPH: a list of pairs of "things"

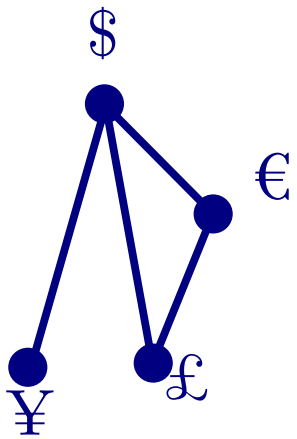
$A = \{(1,2),(1,3),(3,4),(4,5)\}$

$B = \{(\$,€),(£,€),(\$,¥),(\$,£)\}$

$C = \{$(JFK,BNA), (BNA,LAX), (EWR,LHR), (LGA,ATL)$\}$

Vertex/Vertices





GRAPH: a list of pairs of "things"

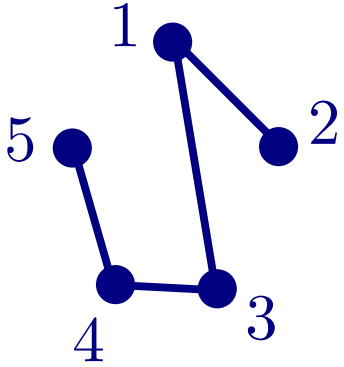
$$A = \{(1,2),(1,3),(3,4),(4,5)\}$$

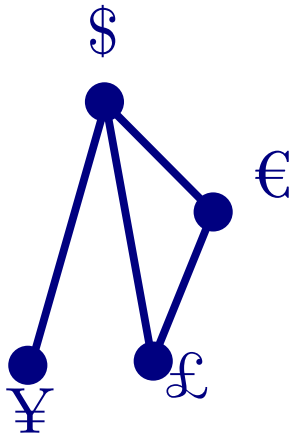
$$B = \{(\$,€),(£,€),(\$,¥),(\$,£)\}$$

$$C = \{(\text{JFK},\text{BNA}),\ (\text{BNA},\text{LAX}),\ (\text{EWR},\text{LHR}),\ (\text{LGA},\text{ATL})\}$$

Edges

Vertex/Vertices



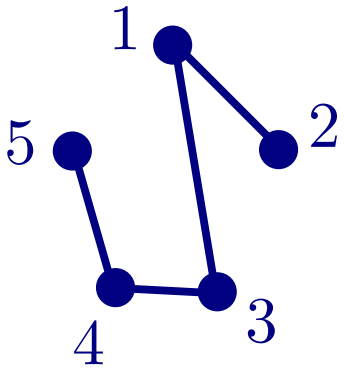


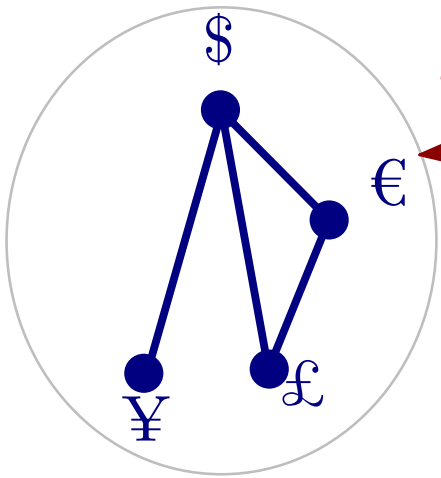
GRAPH: a list of pairs of "things"

$$A = \{(1,2),(1,3),(3,4),(4,5)\}$$
$$B = \{(\$,€),(£,€),(\$,¥),(\$,£)\}$$
$$C = \{(JFK,BNA),\ (BNA,LAX),\ (EWR,LHR),\ (LGA,ATL)\}$$

Vertex/Vertices

Edges





What is the maximum number of edges in this graph?

GRAPH: a list of pairs of "things"

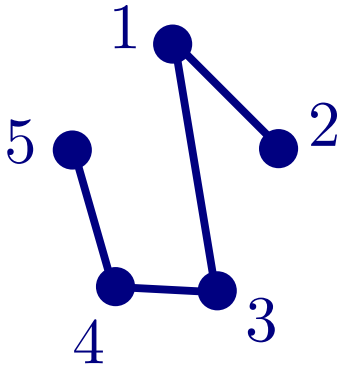
$$A = \{(1,2),(1,3),(3,4),(4,5)\}$$

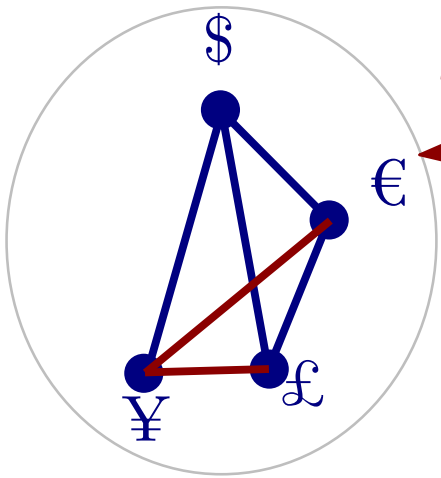
$$B = \{(\$,€),(£,€),(\$,¥),(\$,£)\}$$

$$C = \{(JFK,BNA),\ (BNA,LAX),\ (EWR,LHR),\ (LGA,ATL)\}$$

Vertex/Vertices

Edges





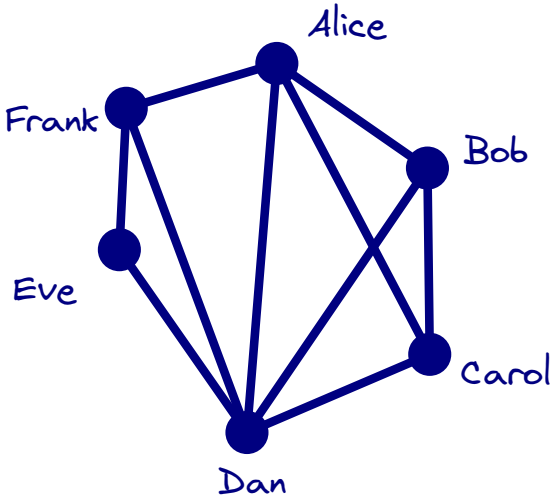
$= 6$

What is the maximum number of edges in this graph?

Counting Question?

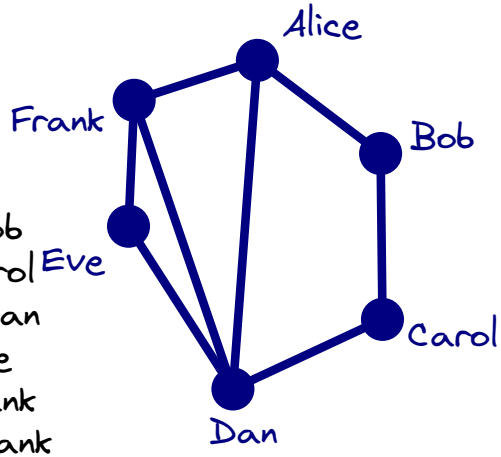
Total number of graphs on four vertices?

Facebook Friendships



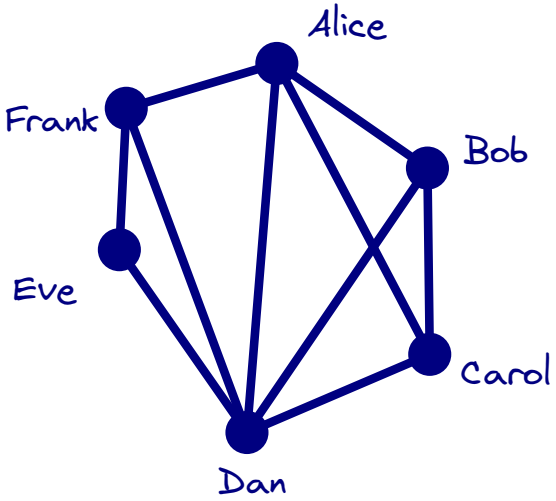
Alice is friends with Bob
Bob i.f.w. Carol
Carol i.f.w. Dan
Dan i.f.w. Eve
Eve i.f.w. Frank
Alice i.f.w. Frank
Alice i.f.w. Carol
Alice i.f.w. Dan
Bob i.f.w. Dan
Dan i.f.w. Frank

Twitter Followers



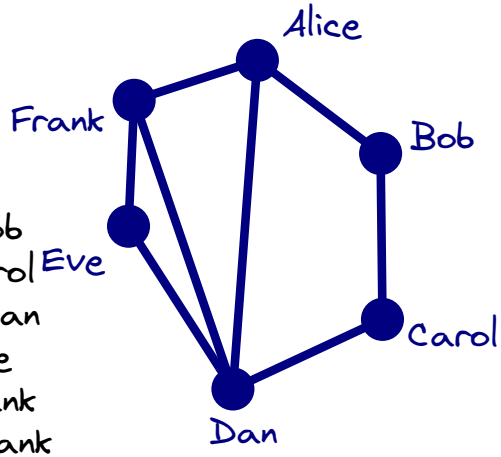
Alice follows Bob
Bob follows Carol
Carol follows Dan
Dan follows Eve
Eve follows Frank
Alice follows Frank
Alice follows Dan
Dan follows Frank

Facebook Friendships



Alice is friends with Bob
Bob i.f.w. Carol
Carol i.f.w. Dan
Dan i.f.w. Eve
Eve i.f.w. Frank
Alice i.f.w. Frank
Alice i.f.w. Carol
Alice i.f.w. Dan
Bob i.f.w. Dan
Dan i.f.w. Frank

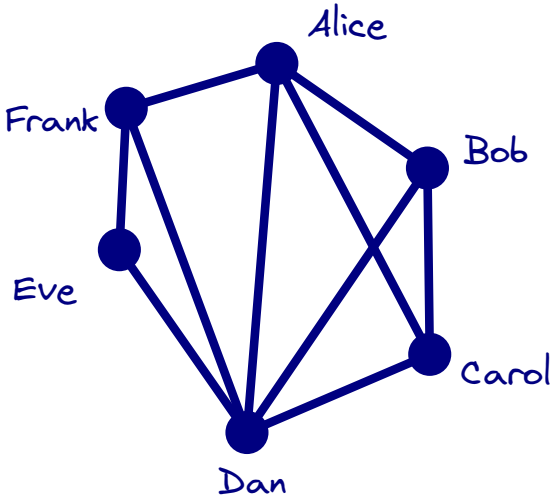
Twitter Followers



Alice follows Bob
Bob follows Carol
Carol follows Dan
Dan follows Eve
Eve follows Frank
Alice follows Frank
Alice follows Dan
Dan follows Frank

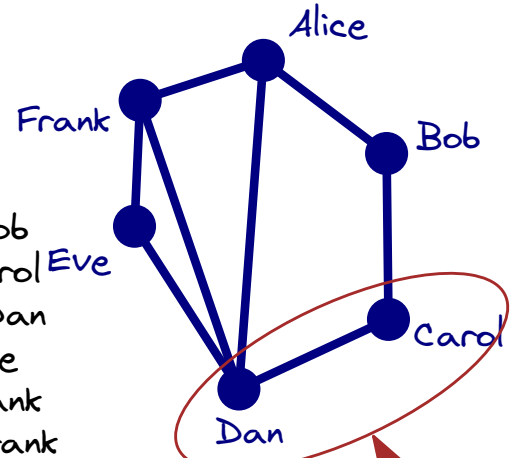
Twitter graph is fundamentally different from facebook graph!

Facebook Friendships



Alice is friends with Bob
Bob i.f.w. Carol
Carol i.f.w. Dan
Dan i.f.w. Eve
Eve i.f.w. Frank
Alice i.f.w. Frank
Alice i.f.w. Carol
Alice i.f.w. Dan
Bob i.f.w. Dan
Dan i.f.w. Frank

Twitter Followers

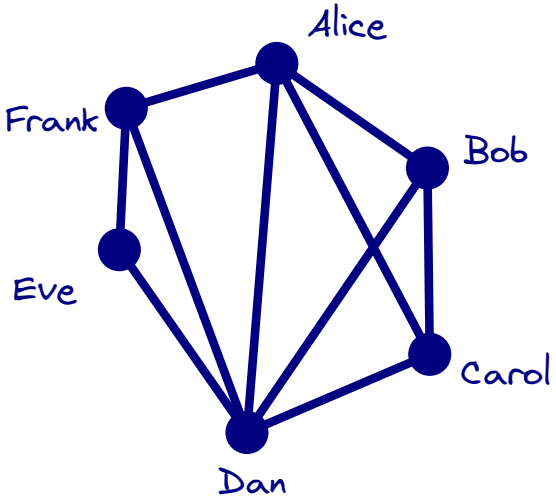


Alice follows Bob
Bob follows Carol
Carol follows Dan
Dan follows Eve
Eve follows Frank
Alice follows Frank
Alice follows Dan
Dan follows Frank

Twitter graph is fundamentally different from facebook graph!

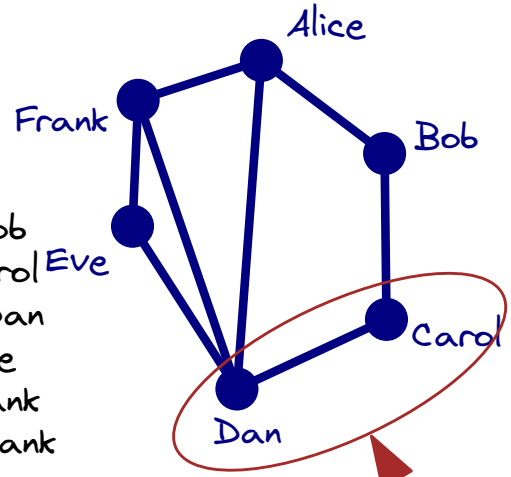
Can't tell whether
Dan follows Carol or
Carol follows Dan!

Facebook Friendships



Alice is friends with Bob
Bob i.f.w. Carol
Carol i.f.w. Dan
Dan i.f.w. Eve
Eve i.f.w. Frank
Alice i.f.w. Frank
Alice i.f.w. Carol
Alice i.f.w. Dan
Bob i.f.w. Dan
Dan i.f.w. Frank

Twitter Followers



Alice follows Bob
Bob follows Carol
Carol follows Dan
Dan follows Eve
Eve follows Frank
Alice follows Frank
Alice follows Dan
Dan follows Frank

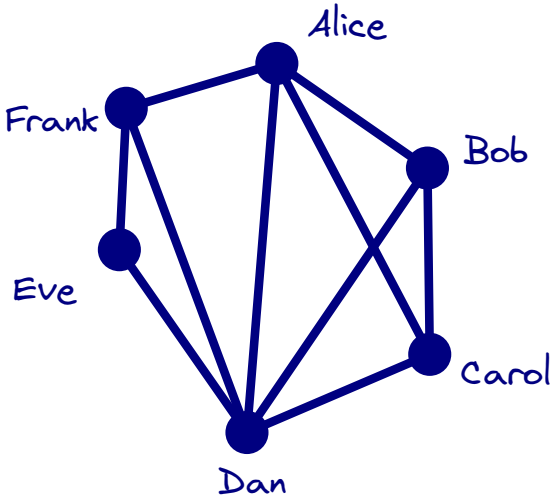
Twitter graph is fundamentally different from facebook graph!

Can't tell whether Dan follows Carol or Carol follows Dan!

We say that:

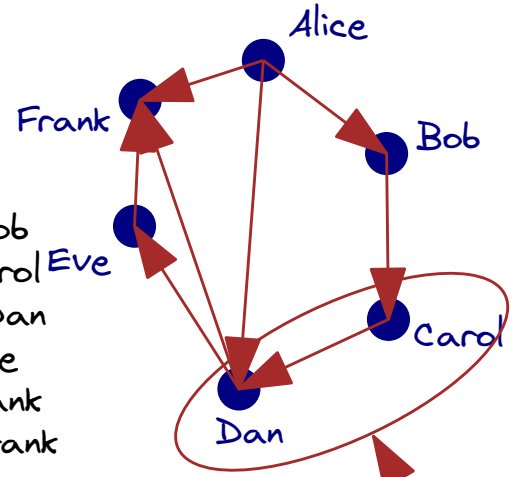
- Facebook is an undirected graph (order of edges does not matter)
- Twitter is a directed graph (order of edges does matter)

Facebook Friendships



Alice is friends with Bob
Bob i.f.w. Carol
Carol i.f.w. Dan
Dan i.f.w. Eve
Eve i.f.w. Frank
Alice i.f.w. Frank
Alice i.f.w. Carol
Alice i.f.w. Dan
Bob i.f.w. Dan
Dan i.f.w. Frank

Twitter Followers



Alice follows Bob
Bob follows Carol
Carol follows Dan
Dan follows Eve
Eve follows Frank
Alice follows Frank
Alice follows Dan
Dan follows Frank

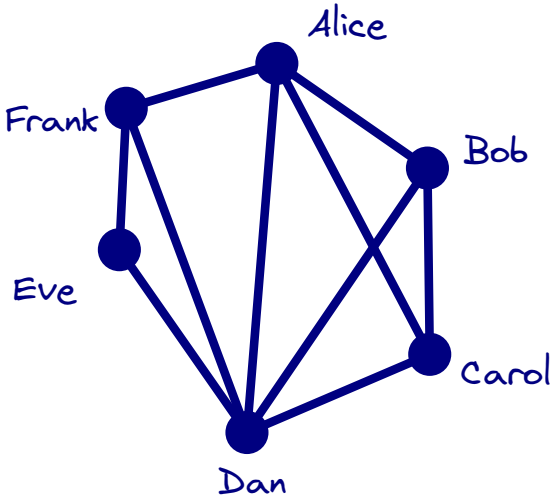
Twitter graph is fundamentally different from facebook graph!

Can't tell whether Dan follows Carol or Carol follows Dan!

We say that:

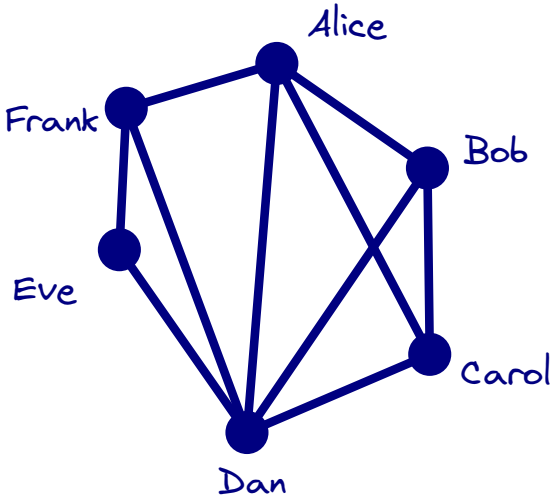
- Facebook is an undirected graph (order of edges does not matter)
- Twitter is a directed graph (order of edges does matter)

Facebook Friendships is an Undirected Graph!



Alice is friends with Bob
Bob i.f.w. Carol
Carol i.f.w. Dan
Dan i.f.w. Eve
Eve i.f.w. Frank
Alice i.f.w. Frank
Alice i.f.w. Carol
Alice i.f.w. Dan
Bob i.f.w. Dan
Dan i.f.w. Frank

Facebook Friendships is an Undirected Graph!



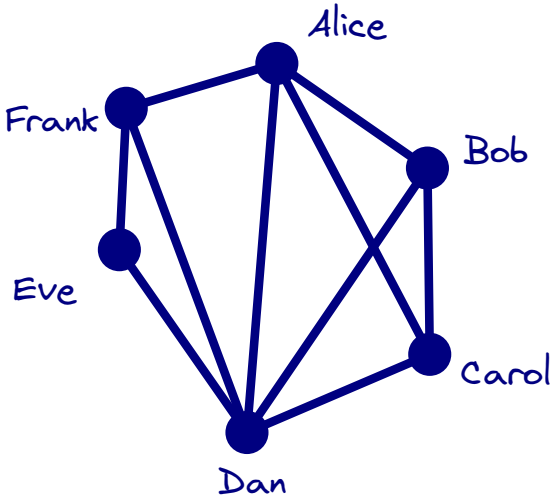
Alice is friends with Bob
Bob i.f.w. Carol
Carol i.f.w. Dan
Dan i.f.w. Eve
Eve i.f.w. Frank
Alice i.f.w. Frank
Alice i.f.w. Carol
Alice i.f.w. Dan
Bob i.f.w. Dan
Dan i.f.w. Frank

Alice is friends with Bob

implies

Bob is friends with Alice

Facebook Friendships is an Undirected Graph!



Alice is friends with Bob
Bob i.f.w. Carol
Carol i.f.w. Dan
Dan i.f.w. Eve
Eve i.f.w. Frank
Alice i.f.w. Frank
Alice i.f.w. Carol
Alice i.f.w. Dan
Bob i.f.w. Dan
Dan i.f.w. Frank

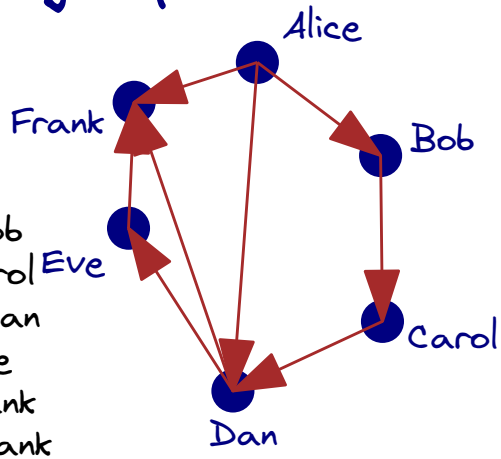
Alice is friends with Bob

implies

Bob is friends with Alice

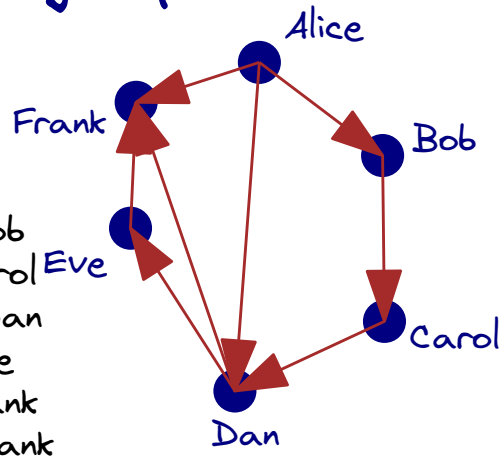
You may write Bob is friends with Alice or you may omit this information.

Twitter Followers is a directed graph!



Alice follows Bob
Bob follows Carol
Carol follows Dan
Dan follows Eve
Eve follows Frank
Alice follows Frank
Alice follows Dan
Dan follows Frank

Twitter Followers is a directed graph!



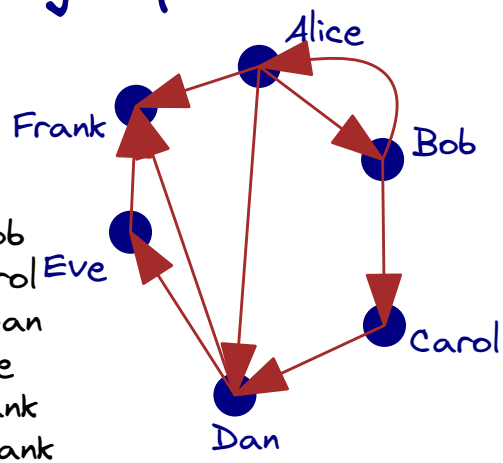
Alice follows Bob
Bob follows Carol
Carol follows Dan
Dan follows Eve
Eve follows Frank
Alice follows Frank
Alice follows Dan
Dan follows Frank

Alice follows Bob

DOES NOT imply

Bob follows Alice

Twitter Followers is a directed graph!



Alice follows Bob
Bob follows Carol
Carol follows Dan
Dan follows Eve
Eve follows Frank
Alice follows Frank
Alice follows Dan
Dan follows Frank

Alice follows Bob

DOES NOT imply

Bob follows Alice

If you also write Bob follows Alice it will create a new edge.

Graph Representations

Graph Representations

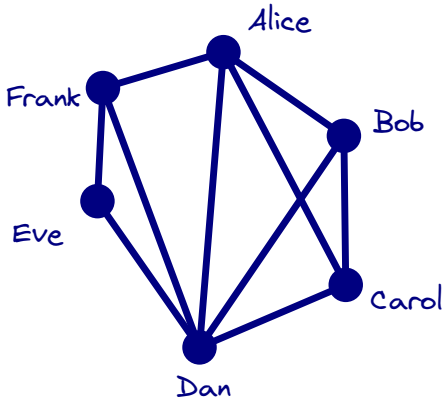
A representation is needed to:

- store graphs in memory
- share graphs information

Graph Representations

A representation is needed to:

- store graphs in memory
- share graphs information



We could just share this picture?

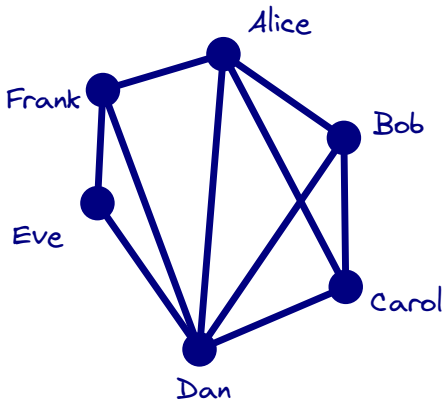
Graph Representations

A representation is needed to:

- store graphs in memory
- share graphs information

But it gets messy pretty fast!

We could just share this picture?



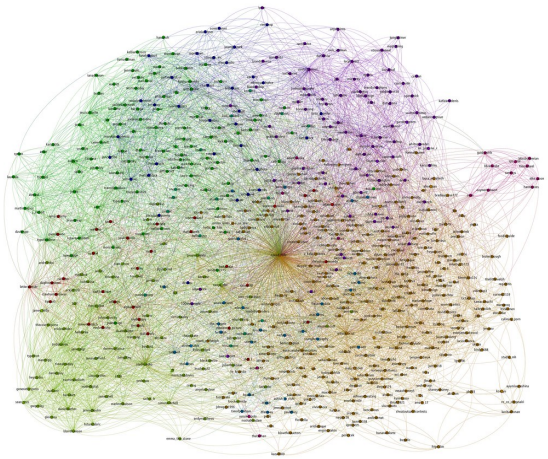
Graph Representations

A representation is needed to:

- store graphs in memory
- share graphs information

But it gets messy pretty fast!

We could just share this picture?



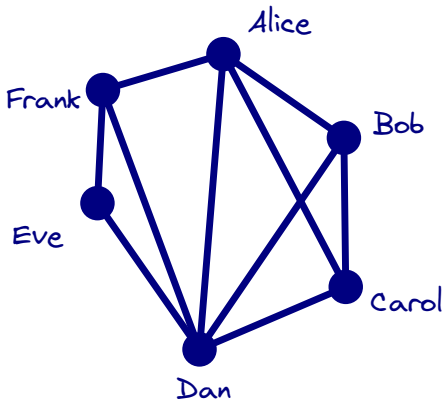
Graph Representations

A representation is needed to:

- store graphs in memory
- share graphs information

But it gets messy pretty fast!

We could just share this picture?



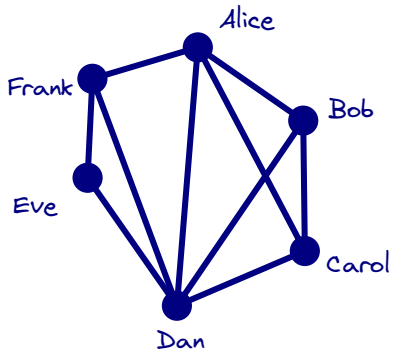
Three common ways to represent:

1. list of all edges
2. separate lists of edges for each vertex
3. yes/no for each possible edge

Graph Representations

A representation is needed to:
- store graphs in memory
- share graphs information

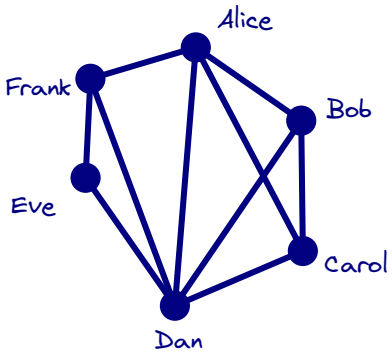
1. Edge List



Graph Representations

A representation is needed to:
- -store graphs in memory
- -share graphs information

1. Edge List



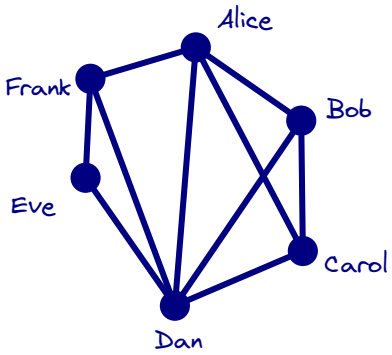
Alice i.f.w. Bob
Bob i.f.w. Carol
Carol i.f.w. Dan
Dan i.f.w. Eve
Eve i.f.w. Frank
Alice i.f.w. Frank
Alice i.f.w. Carol
Alice i.f.w. Dan
Bob i.f.w. Dan
Dan i.f.w. Frank

Graph Representations

A representation is needed to:
- -store graphs in memory
- -share graphs information

1. Edge List

Easy to store but inefficient!



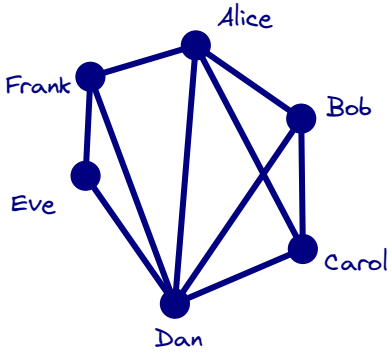
Alice i.f.w. Bob
Bob i.f.w. Carol
Carol i.f.w. Dan
Dan i.f.w. Eve
Eve i.f.w. Frank
Alice i.f.w. Frank
Alice i.f.w. Carol
Alice i.f.w. Dan
Bob i.f.w. Dan
Dan i.f.w. Frank

Graph Representations

A representation is needed to:
- -store graphs in memory
- -share graphs information

1. Edge List

Easy to store but inefficient!



Alice i.f.w. Bob
Bob i.f.w. Carol
Carol i.f.w. Dan
Dan i.f.w. Eve
Eve i.f.w. Frank
Alice i.f.w. Frank
Alice i.f.w. Carol
Alice i.f.w. Dan
Bob i.f.w. Dan
Dan i.f.w. Frank

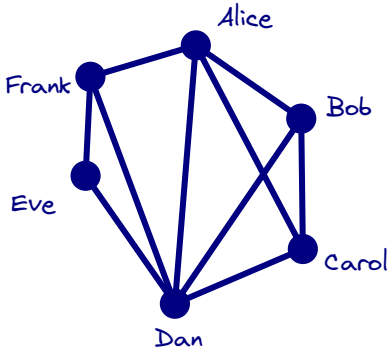
Need to go through the whole list to check wheher
Dan and Frank are friends?
OR
even to check if Eve has any friends?

Graph Representations

A representation is needed to:
- -store graphs in memory
- -share graphs information

1. Edge List

Easy to store but inefficient!



Alice i.f.w. Bob
Bob i.f.w. Carol
Carol i.f.w. Dan
Dan i.f.w. Eve
Eve i.f.w. Frank
Alice i.f.w. Frank
Alice i.f.w. Carol
Alice i.f.w. Dan
Bob i.f.w. Dan
Dan i.f.w. Frank

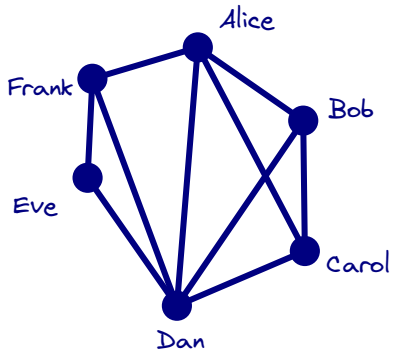
Need to go through the whole list to check wheher Dan and Frank are friends?
OR
even to check if Eve has any friends?

Commonly used in Machine Learning algorithms that deal with graphs with trillions of edges.

Graph Representations

A representation is needed to:
- -store graphs in memory
- -share graphs information

2. Adjacency List

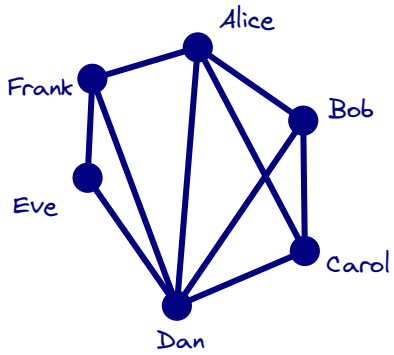


Graph Representations

A representation is needed to:
- -store graphs in memory
- -share graphs information

2. Adjacency List

Easy to store AND efficient!



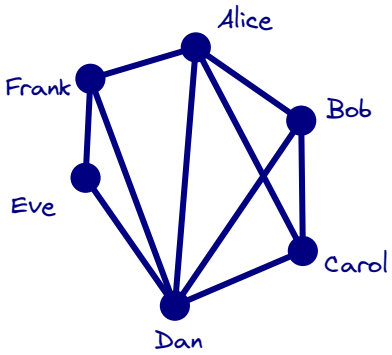
Graph Representations

A representation is needed to:

- -store graphs in memory
- -share graphs information

2. Adjacency List

Easy to store AND efficient!



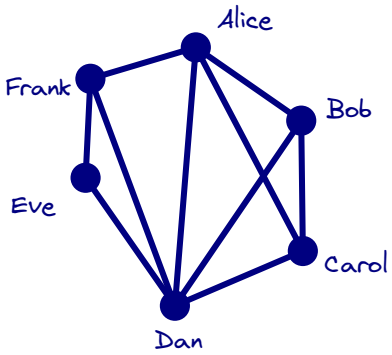
Alice -> Bob, Carol, Dan, Frank
Bob -> Alice, Carol, Dan
Carol -> Alice, Bob, Dan
Dan -> Alice, Bob, Carol, Eve, Frank
Eve -> Dan, Frank
Frank -> Dan, Eve, Alice

Graph Representations

A representation is needed to:
- -store graphs in memory
- -share graphs information

2. Adjacency List

Easy to store AND efficient!



Alice -> Bob, Carol, Dan, Frank
Bob -> Alice, Carol, Dan
Carol -> Alice, Bob, Dan
Dan -> Alice, Bob, Carol, Eve, Frank
Eve -> Dan, Frank
Frank -> Dan, Eve, Alice

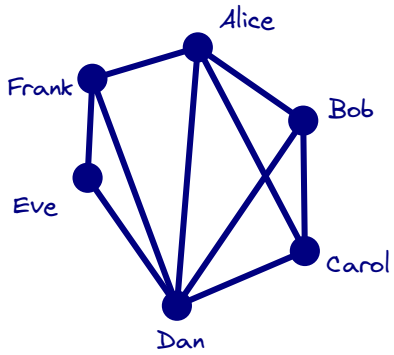
Dan and Frank are friends? Only check Dan's list!
Eve has any friends? Only check Eve's list!

Graph Representations

A representation is needed to:

- -store graphs in memory
- -share graphs information

3. Adjacency Matrix

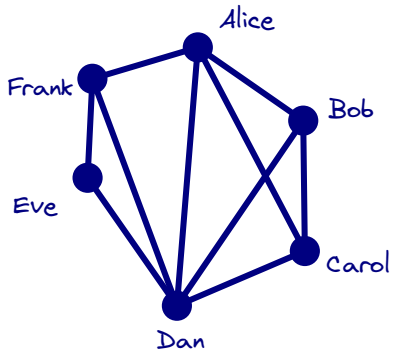


Graph Representations

A representation is needed to:
- -store graphs in memory
- -share graphs information

3. Adjacency Matrix

IDEA: for each possible edge answer YES/NO.

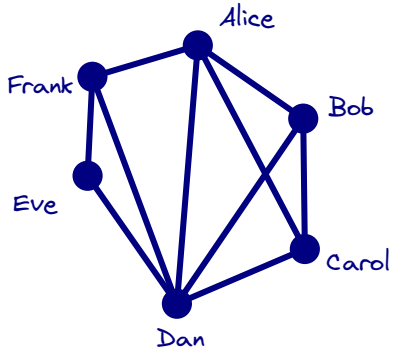


Graph Representations

A representation is needed to:
- -store graphs in memory
- -share graphs information

3. Adjacency Matrix

IDEA: for each possible edge answer YES/NO.



Alice i.f.w. Bob YES
Bob i.f.w. Carol YES
Carol i.f.w. Dan YES
Dan i.f.w. Eve YES
Eve i.f.w. Frank YES
Alice i.f.w. Frank YES
Alice i.f.w. Carol YES
Alice i.f.w. Dan YES
Bob i.f.w. Dan YES
Dan i.f.w. Frank YES
Alice i.f.w. Eve NO
Bob i.f.w. Eve NO
Carol i.f.w. Eve NO
Bob i.f.w. Frank NO
Carol i.f.w. Frank NO

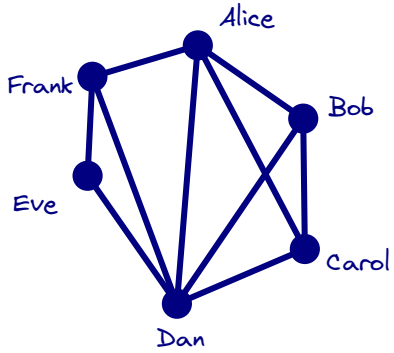
Graph Representations

A representation is needed to:
- -store graphs in memory
- -share graphs information

3. Adjacency Matrix

IDEA: for each possible edge answer YES/NO.

Very efficient but memory costly!



Alice i.f.w. Bob YES
Bob i.f.w. Carol YES
Carol i.f.w. Dan YES
Dan i.f.w. Eve YES
Eve i.f.w. Frank YES
Alice i.f.w. Frank YES
Alice i.f.w. Carol YES
Alice i.f.w. Dan YES
Bob i.f.w. Dan YES
Dan i.f.w. Frank YES
Alice i.f.w. Eve NO
Bob i.f.w. Eve NO
Carol i.f.w. Eve NO
Bob i.f.w. Frank NO
Carol i.f.w. Frank NO

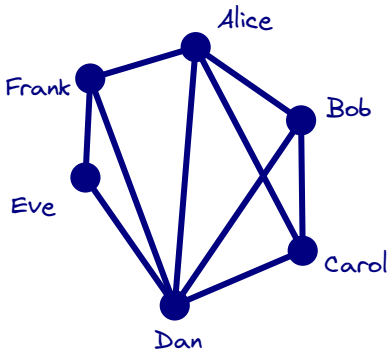
Graph Representations

A representation is needed to:
- -store graphs in memory
- -share graphs information

3. Adjacency Matrix

IDEA: for each possible edge answer YES/NO.

Very efficient but memory costly!



Alice i.f.w. Bob YES
Bob i.f.w. Carol YES
Carol i.f.w. Dan YES
Dan i.f.w. Eve YES
Eve i.f.w. Frank YES
Alice i.f.w. Frank YES
Alice i.f.w. Carol YES
Alice i.f.w. Dan YES
Bob i.f.w. Dan YES
Dan i.f.w. Frank YES
Alice i.f.w. Eve NO
Bob i.f.w. Eve NO
Carol i.f.w. Eve NO
Bob i.f.w. Frank NO
Carol i.f.w. Frank NO

	A	B	C	D	E	F
A						
B						
C						
D						
E						
F						

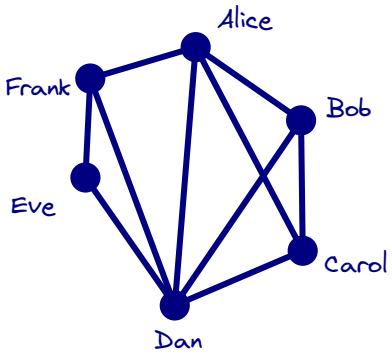
Graph Representations

A representation is needed to:
- -store graphs in memory
- -share graphs information

3. Adjacency Matrix

IDEA: for each possible edge answer YES/NO.

Very efficient but memory costly!



Alice i.f.w. Bob YES
Bob i.f.w. Carol YES
Carol i.f.w. Dan YES
Dan i.f.w. Eve YES
Eve i.f.w. Frank YES
Alice i.f.w. Frank YES
Alice i.f.w. Carol YES
Alice i.f.w. Dan YES
Bob i.f.w. Dan YES
Dan i.f.w. Frank YES
Alice i.f.w. Eve NO
Bob i.f.w. Eve NO
Carol i.f.w. Eve NO
Bob i.f.w. Frank NO
Carol i.f.w. Frank NO

	A	B	C	D	E	F
A	0	1	1	1	0	1
B	1	0	1	1	0	0
C	1	1	0	1	0	0
D	1	1	1	0	1	1
E	0	0	0	1	0	1
F	1	0	0	1	1	0

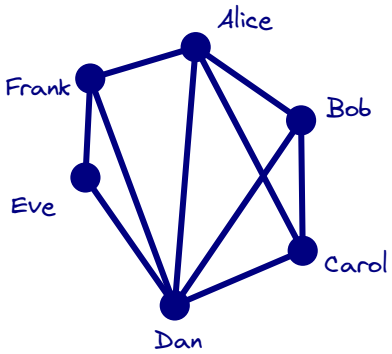
Graph Representations

A representation is needed to:
- -store graphs in memory
- -share graphs information

3. Adjacency Matrix

IDEA: for each possible edge answer YES/NO.

Very efficient but memory costly!



Alice i.f.w. Bob YES
Bob i.f.w. Carol YES
Carol i.f.w. Dan YES
Dan i.f.w. Eve YES
Eve i.f.w. Frank YES
Alice i.f.w. Frank YES
Alice i.f.w. Carol YES
Alice i.f.w. Dan YES
Bob i.f.w. Dan YES
Dan i.f.w. Frank YES
Alice i.f.w. Eve NO
Bob i.f.w. Eve NO
Carol i.f.w. Eve NO
Bob i.f.w. Frank NO
Carol i.f.w. Frank NO

	A	B	C	D	E	F
A	0	1	1	1	0	1
B	1	0	1	1	0	0
C	1	1	0	1	0	0
D	1	1	1	0	1	1
E	0	0	0	1	0	1
F	1	0	0	1	1	0

Dan and Frank are friends? Only check cell with Dan's column and Frank's row!
Eve has any friends? Only check Eve's row!

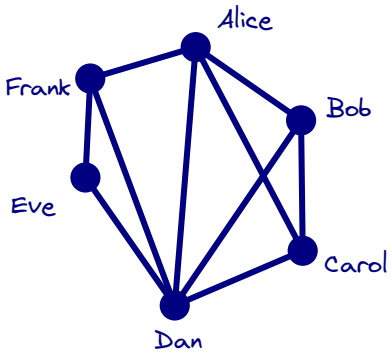
Graph Representations

A representation is needed to:
- -store graphs in memory
- -share graphs information

3. Adjacency Matrix

IDEA: for each possible edge answer YES/NO.

Very efficient but memory costly!



Alice i.f.w. Bob YES
Bob i.f.w. Carol YES
Carol i.f.w. Dan YES
Dan i.f.w. Eve YES
Eve i.f.w. Frank YES
Alice i.f.w. Frank YES
Alice i.f.w. Carol YES
Alice i.f.w. Dan YES
Bob i.f.w. Dan YES
Dan i.f.w. Frank YES
Alice i.f.w. Eve NO
Bob i.f.w. Eve NO
Carol i.f.w. Eve NO
Bob i.f.w. Frank NO
Carol i.f.w. Frank NO

	A	B	C	D	E	F
A	0	1	1	1	0	1
B	1	0	1	1	0	0
C	1	1	0	1	0	0
D	1	1	1	0	1	1
E	0	0	0	1	0	1
F	1	0	0	1	1	0

Dan and Frank are friends? Only check cell with Dan's column and Frank's row!
Eve has any friends? Only check Eve's row!

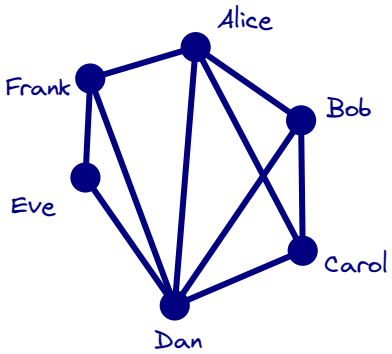
Graph Representations

A representation is needed to:
- -store graphs in memory
- -share graphs information

3. Adjacency Matrix

IDEA: for each possible edge answer YES/NO.

Very efficient but memory costly!

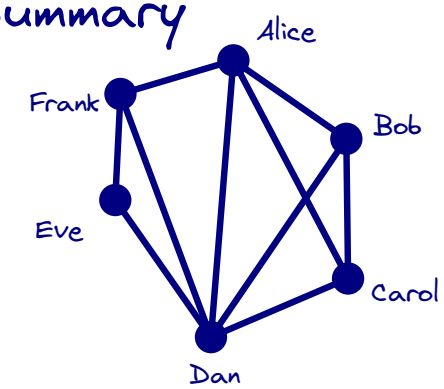


Alice i.f.w. Bob YES
Bob i.f.w. Carol YES
Carol i.f.w. Dan YES
Dan i.f.w. Eve YES
Eve i.f.w. Frank YES
Alice i.f.w. Frank YES
Alice i.f.w. Carol YES
Alice i.f.w. Dan YES
Bob i.f.w. Dan YES
Dan i.f.w. Frank YES
Alice i.f.w. Eve NO
Bob i.f.w. Eve NO
Carol i.f.w. Eve NO
Bob i.f.w. Frank NO
Carol i.f.w. Frank NO

	A	B	C	D	E	F
A	0	1	1	1	0	1
B	1	0	1	1	0	0
C	1	1	0	1	0	0
D	1	1	1	0	1	1
E	0	0	0	1	0	1
F	1	0	0	1	1	0

Dan and Frank are friends? Only check cell with Dan's column and Frank's row!
Eve has any friends? Only check Eve's row!

Graph Representations: Summary

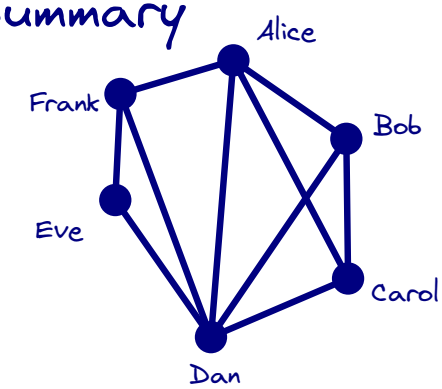


1. Edge List

2. Adjacency List

3. Adjacency Matrix

Graph Representations: Summary



1. Edge List

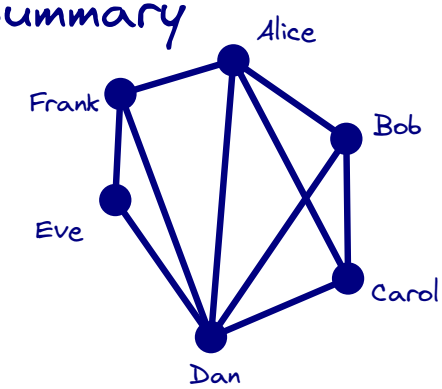
Memory footprint:

$\Theta(|E|)$ we keep all the edges.

2. Adjacency List

3. Adjacency Matrix

Graph Representations: Summary



1. Edge List

Memory footprint:

$\Theta(|E|)$ we keep all the edges.

2. Adjacency List

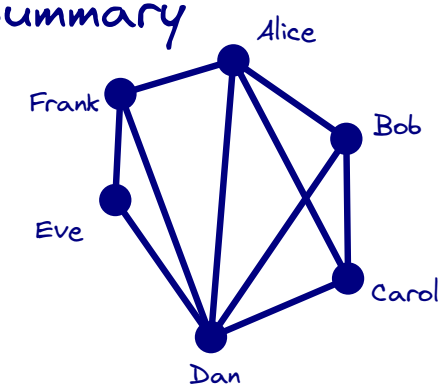
Memory footprint:

$\Theta(|V| + |E|)$ We keep neighbors for each vertex.

3. Adjacency Matrix

Alice -> Bob, Carol, Dan, Frank
Bob -> Alice, Carol, Dan
Carol -> Alice, Bob, Dan
Dan -> Alice, Bob, Carol, Eve, Frank
Eve -> Dan, Frank
Frank -> Dan, Eve, Alice

Graph Representations: Summary



1. Edge List

Memory footprint:

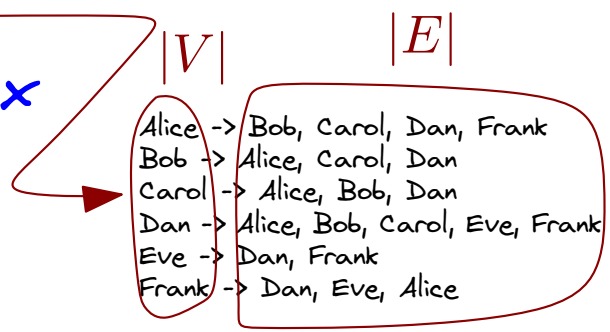
$\Theta(|E|)$ we keep all the edges.

2. Adjacency List

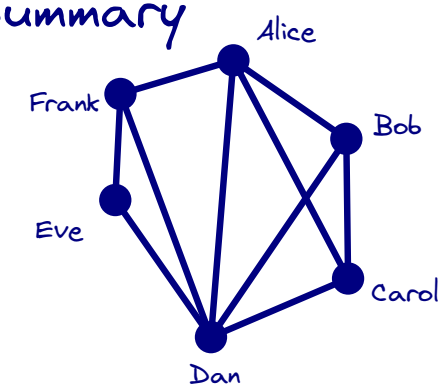
Memory footprint:

$\Theta(|V| + |E|)$ We keep neighbors for each vertex.

3. Adjacency Matrix



Graph Representations: Summary



1. Edge List

Memory footprint:

$\Theta(|E|)$ we keep all the edges.

2. Adjacency List

$$\Theta(|V|) \text{ if } |V| > |E|$$
$$\Theta(|E|) \text{ if } |V| < |E|$$
$$= \max(|V|, |E|)$$

Memory footprint:

$\Theta(|V| + |E|)$ We keep neighbors for each vertex.

3. Adjacency Matrix

$|V|$ $|E|$

Alice -> Bob, Carol, Dan, Frank
Bob -> Alice, Carol, Dan
Carol -> Alice, Bob, Dan
Dan -> Alice, Bob, Carol, Eve, Frank
Eve -> Dan, Frank
Frank -> Dan, Eve, Alice

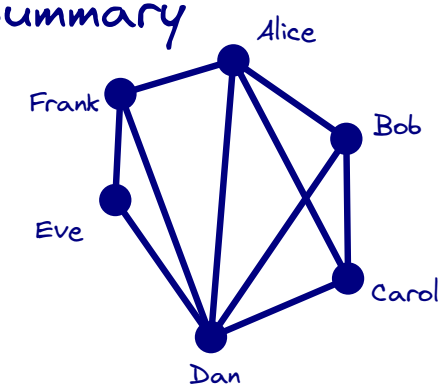
TOPHAT Question

- For which graphs, adjacency list takes $\Theta(|E|)$ memory?

TOPHAT Question

- For which graphs, adjacency list takes $\Theta(|E|)$ memory?
- All "connected graphs" because they have $\Omega(|V|)$ edges.

Graph Representations: Summary



1. Edge List

Memory footprint:

$\Theta(|E|)$ we keep all the edges.

2. Adjacency List

Memory footprint:

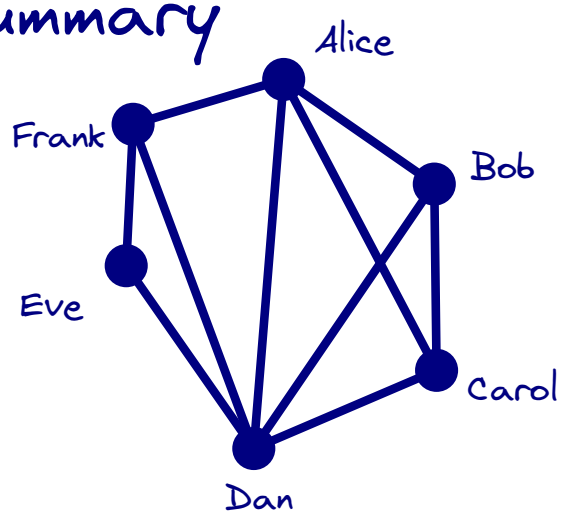
$\Theta(|V| + |E|)$ We keep neighbors for each vertex.

3. Adjacency Matrix

Memory footprint:

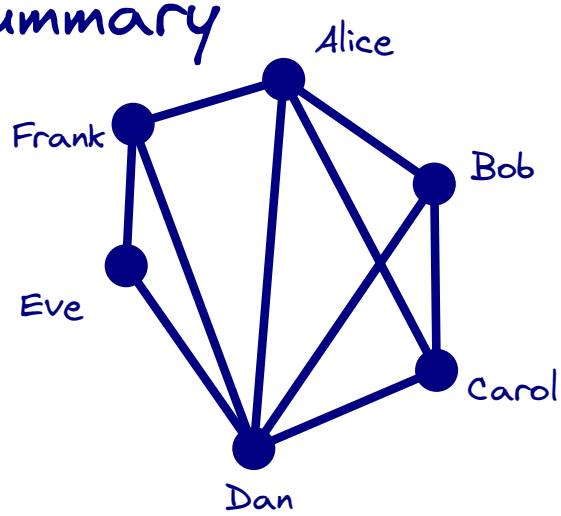
$\Theta(|V|^2)$ We keep a matrix of all possible edges.

Graph Representations: Summary



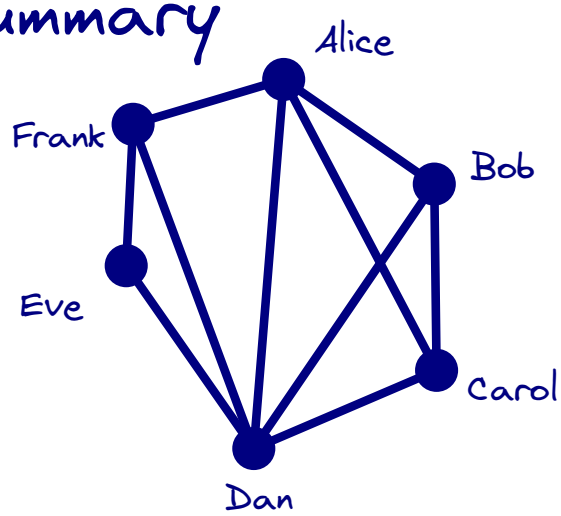
Graph Representations: Summary

1. Edge List



Graph Representations: Summary

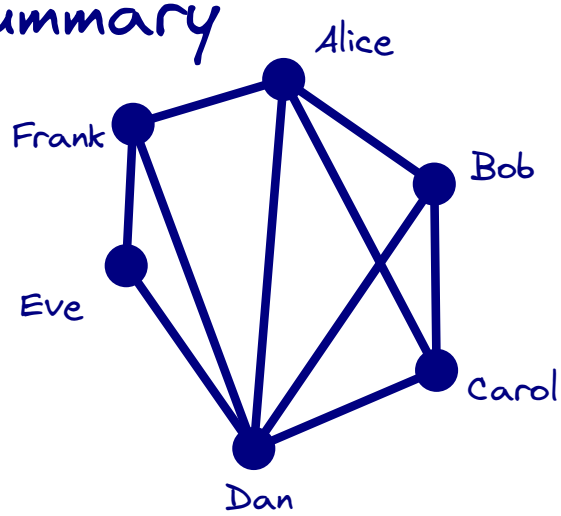
1. Edge List



List all neighbors of a vertex:

$\Theta(|E|)$ we have to go through all the edges.

Graph Representations: Summary



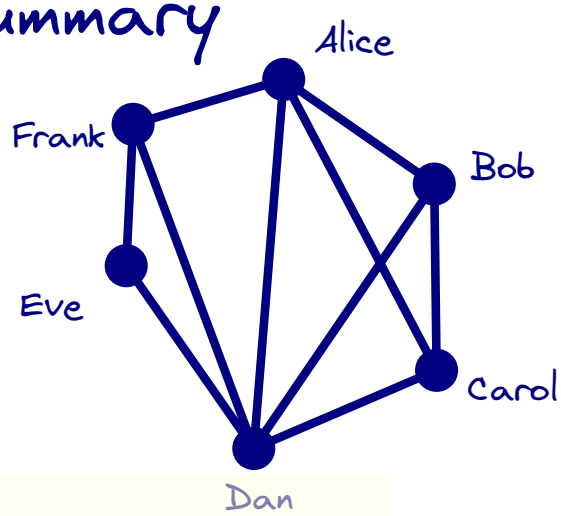
1. Edge List

List all neighbors of a vertex:

$\Theta(|E|)$ we have to go through all the edges.

2. Adjacency List

Graph Representations: Summary



1. Edge List

List all neighbors of a vertex:

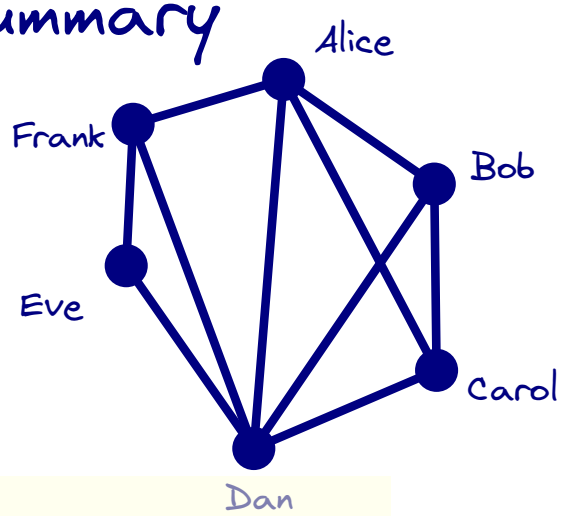
$\Theta(|E|)$ we have to go through all the edges.

2. Adjacency List

List all neighbors of a vertex:

$\Theta(|V|)$ Only need to check one of the lists.

Graph Representations: Summary



1. Edge List

List all neighbors of a vertex:

$\Theta(|E|)$ we have to go through all the edges.

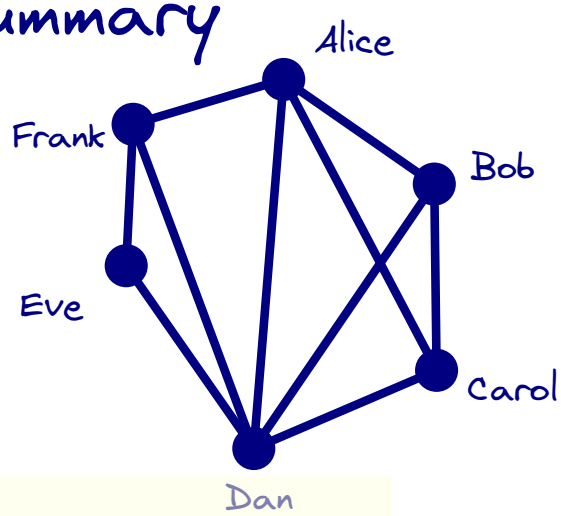
2. Adjacency List

List all neighbors of a vertex:

$\Theta(|V|)$ Only need to check one of the lists.

3. Adjacency Matrix

Graph Representations: Summary



1. Edge List

List all neighbors of a vertex:

$\Theta(|E|)$ we have to go through all the edges.

2. Adjacency List

List all neighbors of a vertex:

$\Theta(|V|)$ Only need to check one of the lists.

3. Adjacency Matrix

List all neighbors of a vertex:

$\Theta(|V|)$ Only need to check one row in the matrix.

Preferred Data Structures/Representations

- Traverse a graph?

Preferred Data Structures/Representations

- Traverse a graph?
- Adjacency List

Preferred Data Structures/Representations

- Traverse a graph?
- Adjacency List
- Delete an edge/Insert a new edge?

Preferred Data Structures/Representations

- Traverse a graph?
- Adjacency List
- Delete an edge/Insert a new edge?
- Adjacency Matrix

Preferred Data Structures/Representations

- Traverse a graph?
- Adjacency List
- Delete an edge/Insert a new edge?
- Adjacency Matrix
- Count Neighbors of a vertex?

Preferred Data Structures/Representations

- Traverse a graph?
- Adjacency List
- Delete an edge/Insert a new edge?
- Adjacency Matrix
- Count Neighbors of a vertex?
- Adjacency List

Preferred Data Structures/Representations

- Traverse a graph?
- Adjacency List
- Delete an edge/Insert a new edge?
- Adjacency Matrix
- Count Neighbors of a vertex?
- Adjacency List
- Operations on a graph with 10^6 vertices?

Preferred Data Structures/Representations

- Traverse a graph?
- Adjacency List
- Delete an edge/Insert a new edge?
- Adjacency Matrix
- Count Neighbors of a vertex?
- Adjacency List
- Operations on a graph with 10^6 vertices?
- Adjacency List

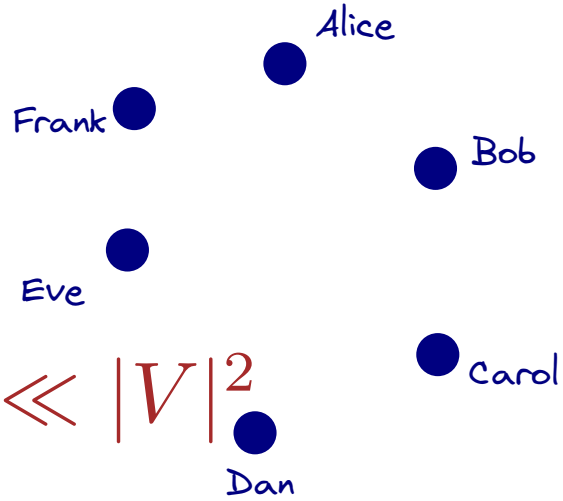
Preferred Data Structures/Representations

- Traverse a graph?
- Adjacency List
- Delete an edge/Insert a new edge?
- Adjacency Matrix
- Count Neighbors of a vertex?
- Adjacency List
- Operations on a graph with 10^6 vertices?
- Adjacency List
- Operations on a sparse graph vs dense graph?

Sparse vs Dense Graphs

Sparse vs Dense Graphs

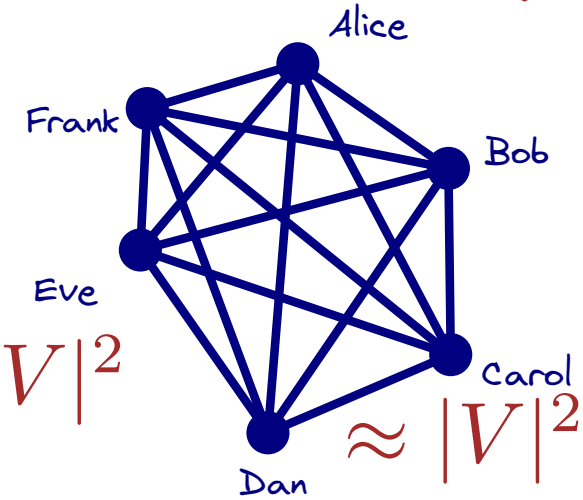
min. number of edges max. number of edges



0

$\ll |V|^2$

$\binom{|V|}{2} \approx |V|^2$



$\approx |V|^2$

SPARSE

twitter: 450 million vertices with 40-50 followers avg.

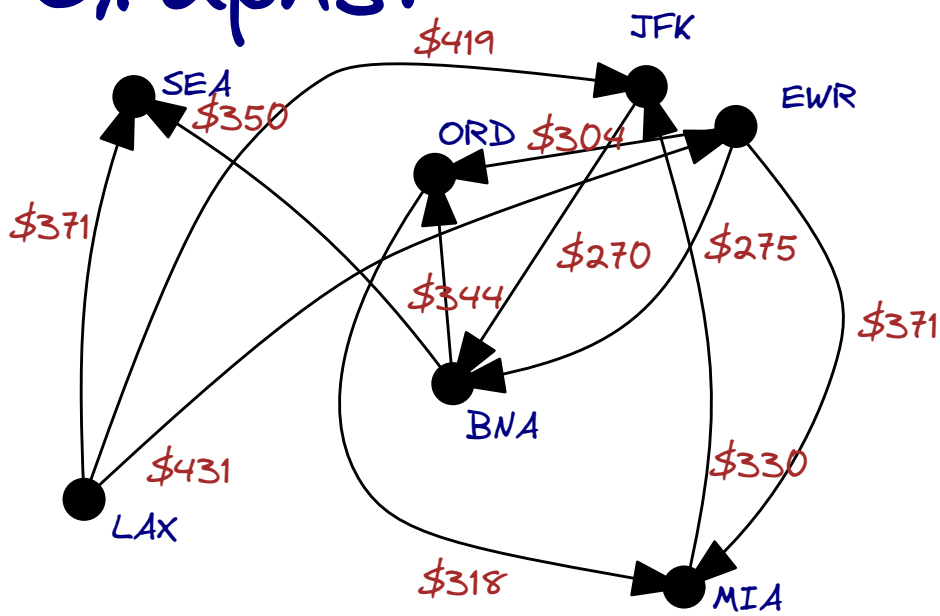
Facebook: 2.9 billion vertices with 190 friends avg.

DENSE

crypto-currency graph is complete.

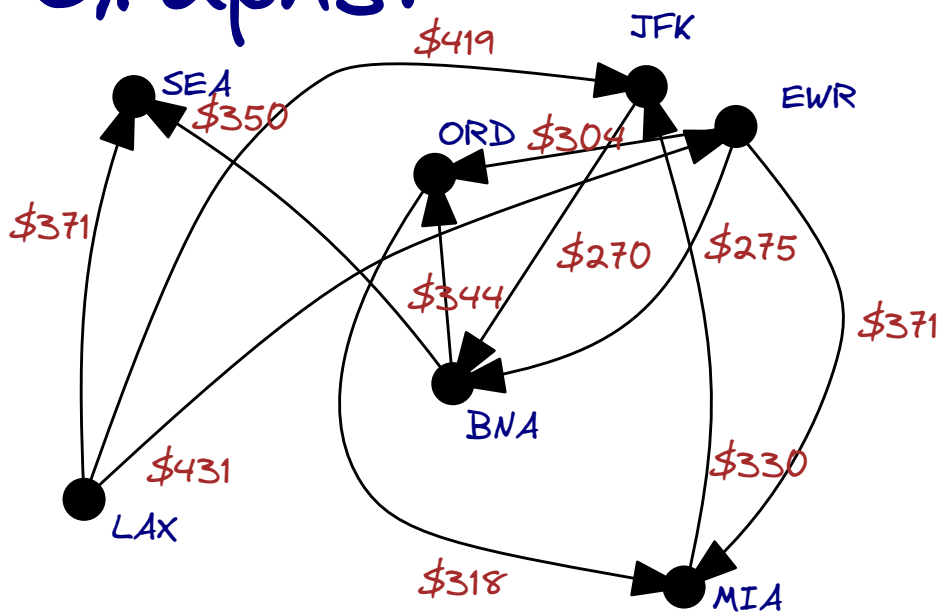
Weighted Graphs?

Graph can also represent weighted networks with weights on edges.



Weighted Graphs?

Graph can also represent weighted networks with weights on edges.

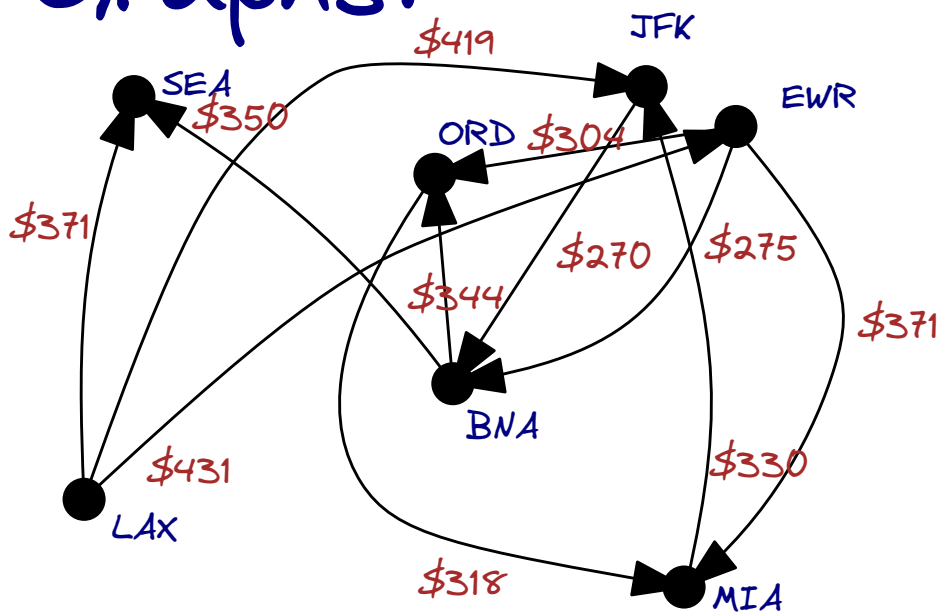



```
EWR -> MIA 371, BNA 275, ORD 304
JFK -> BNA 270
MIA -> JFK 330
BNA -> ORD 344, SEA 350
ORD -> MIA 318
SEA ->
LAX -> SEA 371, JFK 419, EWR 431
```

Weighted Adjacency List

Weighted Graphs?

Graph can also represent weighted networks with weights on edges.




```
EWR -> MIA 371, BNA 275, ORD 304
JFK -> BNA 270
MIA -> JFK 330
BNA -> ORD 344, SEA 350
ORD -> MIA 318
SEA ->
LAX -> SEA 371, JFK 419, EWR 431
```

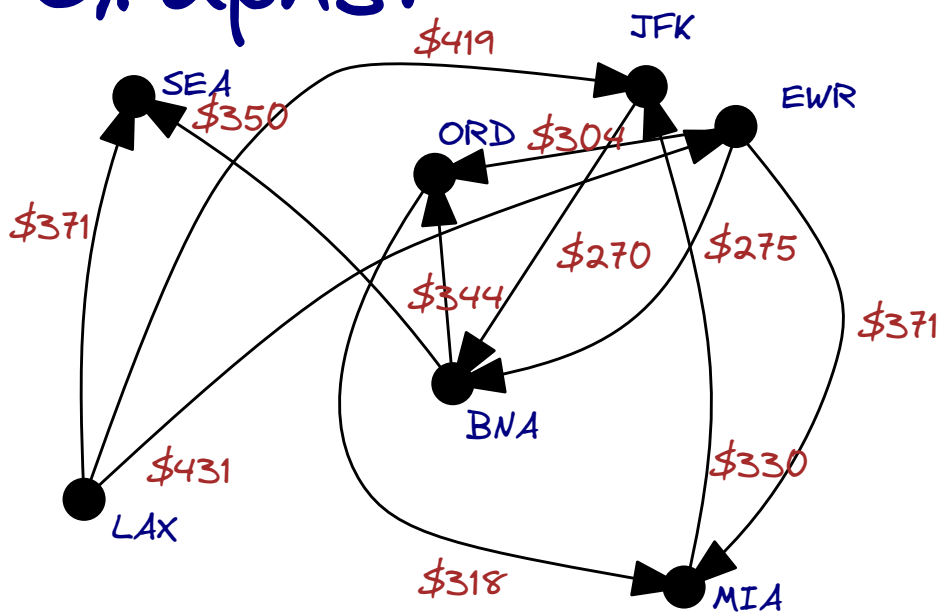
Weighted Adjacency List

Weighted Adjacency Matrix

	EWR	JFK	MIA	BNA	ORD	SEA	LAX
EWR	0	0	371	275	304	0	0
JFK	0	0	0	270	0	0	0
MIA	0	330	0	0	0	0	0
BNA	0	0	0	0	344	250	0
ORD	0	0	318	0	0	0	0
SEA	0	0	0	0	0	0	0
LAX	431	419	0	0	0	371	0

Weighted Graphs?

Graph can also represent weighted networks with weights on edges.



EWR -> MIA 371, BNA 275, ORD 304
JFK -> BNA 270
MIA -> JFK 330
BNA -> ORD 344, SEA 350
ORD -> MIA 318
SEA ->
LAX -> SEA 371, JFK 419, EWR 431

A weighted graph can be directed or undirected!

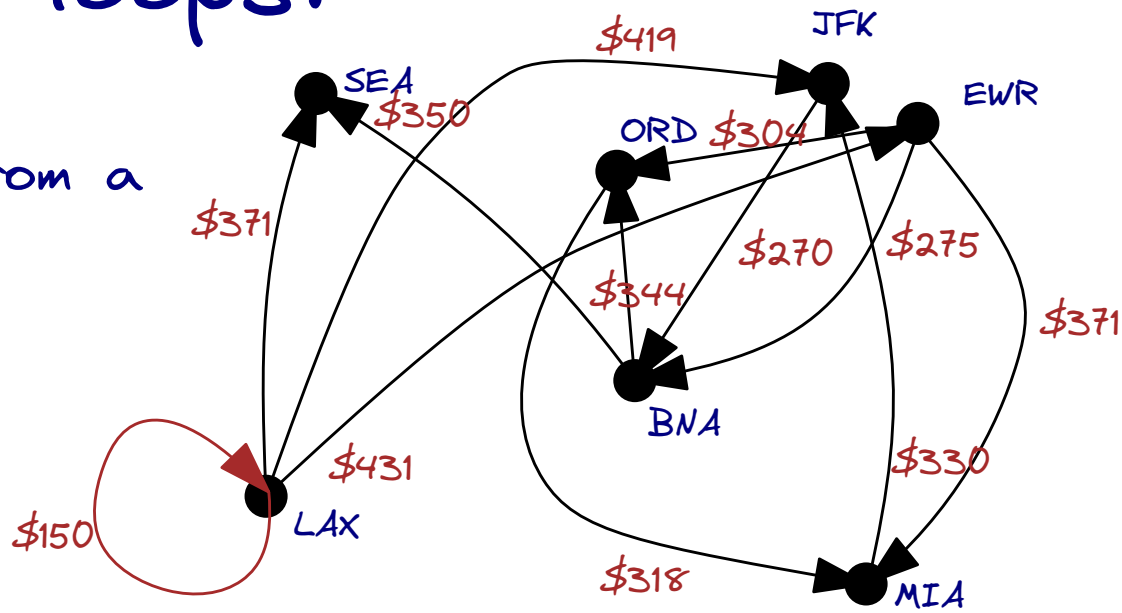
Weighted Adjacency Matrix

	EWR	JFK	MIA	BNA	ORD	SEA	LAX
EWR	0	0	371	275	304	0	0
JFK	0	0	0	270	0	0	0
MIA	0	330	0	0	0	0	0
BNA	0	0	0	0	344	250	0
ORD	0	0	318	0	0	0	0
SEA	0	0	0	0	0	0	0
LAX	431	419	0	0	0	371	0

Weighted Adjacency List

Self-loops?

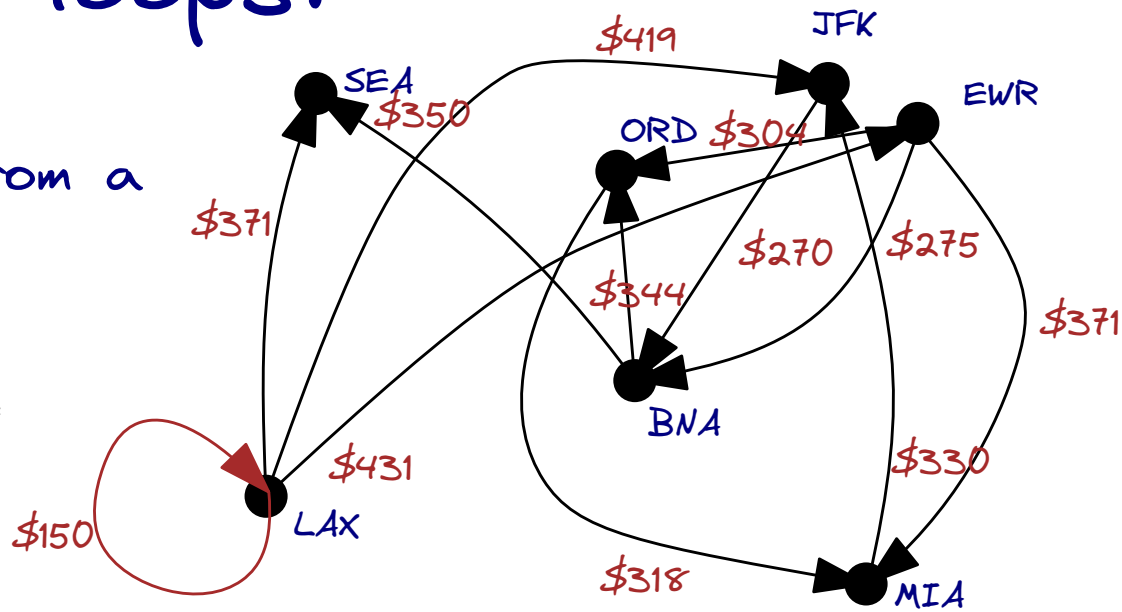
A self-loop is an edge from a vertex to itself.



Example: a plane that takes off from LAX, wanders around for a while and then lands back at LAX.

Self-loops?

A self-loop is an edge from a vertex to itself.



EWR -> MIA 371, BNA 275, ORD 304
JFK -> BNA 270
MIA -> JFK 330
BNA -> ORD 344, SEA 350
ORD -> MIA 318
SEA ->
LAX -> SEA 371, JFK 419, EWR 431, LAX 150

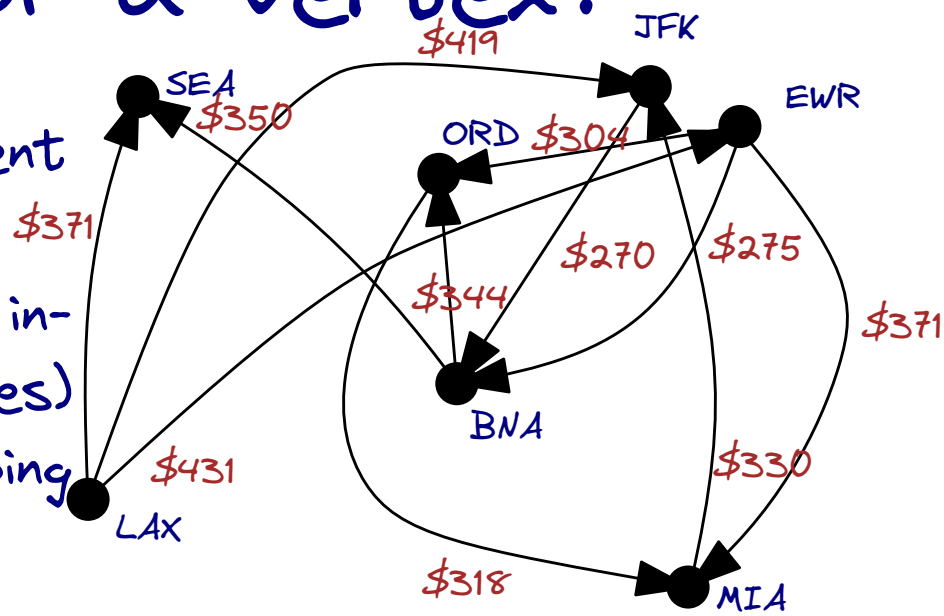
Example: a plane that takes off from LAX, wanders around for a while and then lands back at LAX.

	EWR	JFK	MIA	BNA	ORD	SEA	LAX
EWR	0	0	371	275	304	0	0
JFK	0	0	0	270	0	0	0
MIA	0	330	0	0	0	0	0
BNA	0	0	0	0	344	250	0
ORD	0	0	318	0	0	0	0
SEA	0	0	0	0	0	0	0
LAX	431	419	0	0	0	371	150

DEGREE of a vertex?

is the number of edges adjacent to it.

For directed graphs, we have in-degree (number of incoming edges) and out-degree (number of outgoing edges).



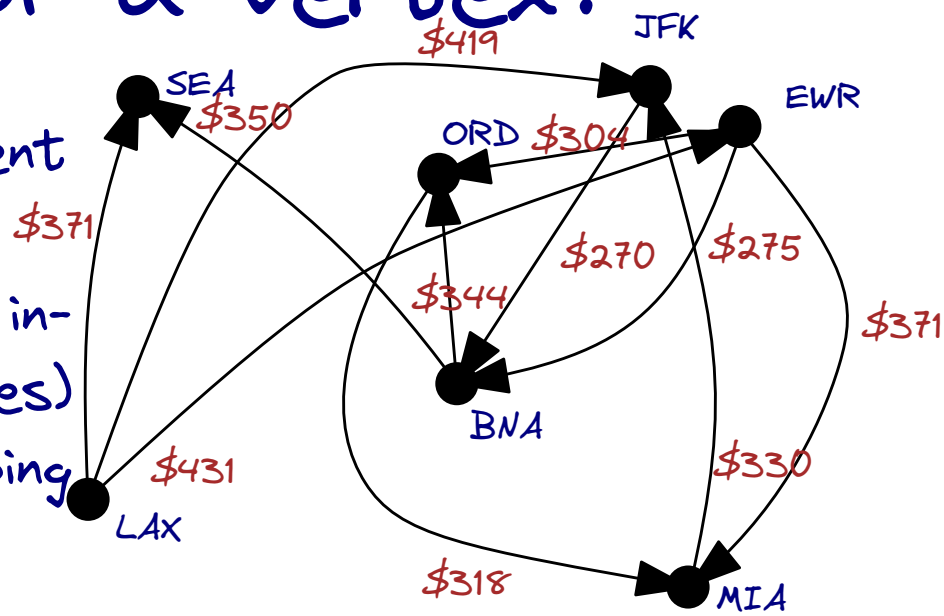
EWR -> MIA 371, BNA 275, ORD 304
JFK -> BNA 270
MIA -> JFK 330
BNA -> ORD 344, SEA 350
ORD -> MIA 318
SEA ->
LAX -> SEA 371, JFK 419, EWR 431

Example: $deg^-(SEA) = 2, deg^+(SEA) = 0$

DEGREE of a vertex?

is the number of edges adjacent to it.

For directed graphs, we have in-degree (number of incoming edges) and out-degree (number of outgoing edges).



EWR -> MIA 371, BNA 275, ORD 304
JFK -> BNA 270
MIA -> JFK 330
BNA -> ORD 344, SEA 350
ORD -> MIA 318
SEA ->
LAX -> SEA 371, JFK 419, EWR 431

Example: $deg^-(SEA) = 2, deg^+(SEA) = 0$

ORD is an OUT-neighbor of BNA

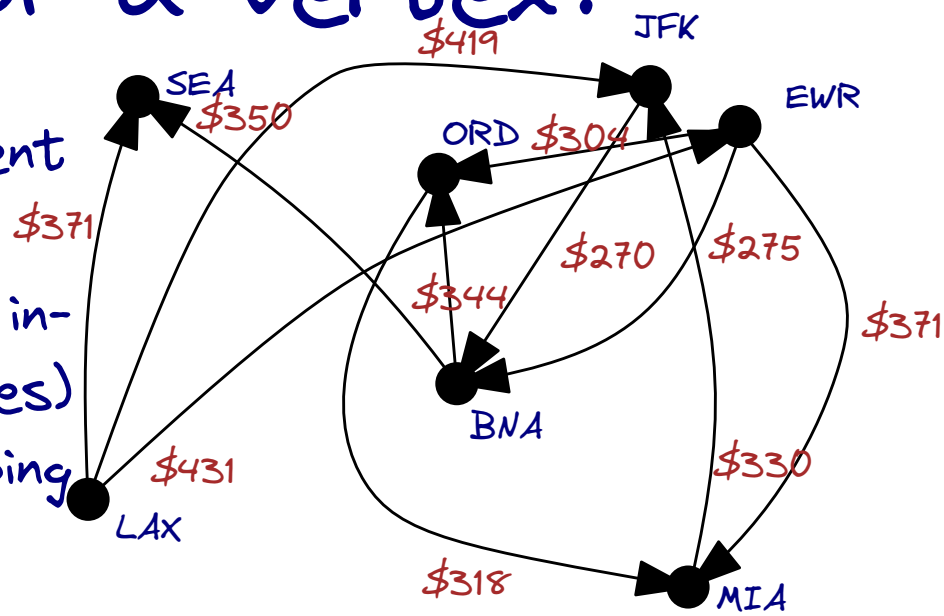
BNA is an IN-neighbor of ORD

ORD is adjacent to BNA.

DEGREE of a vertex?

is the number of edges adjacent to it.

For directed graphs, we have in-degree (number of incoming edges) and out-degree (number of outgoing edges).



EWR -> MIA 371, BNA 275, ORD 304
JFK -> BNA 270
MIA -> JFK 330
BNA -> ORD 344, SEA 350
ORD -> MIA 318
SEA ->
LAX -> SEA 371, JFK 419, EWR 431

Example: $deg^-(SEA) = 2, deg^+(SEA) = 0$

ORD is an OUT-neighbor of BNA

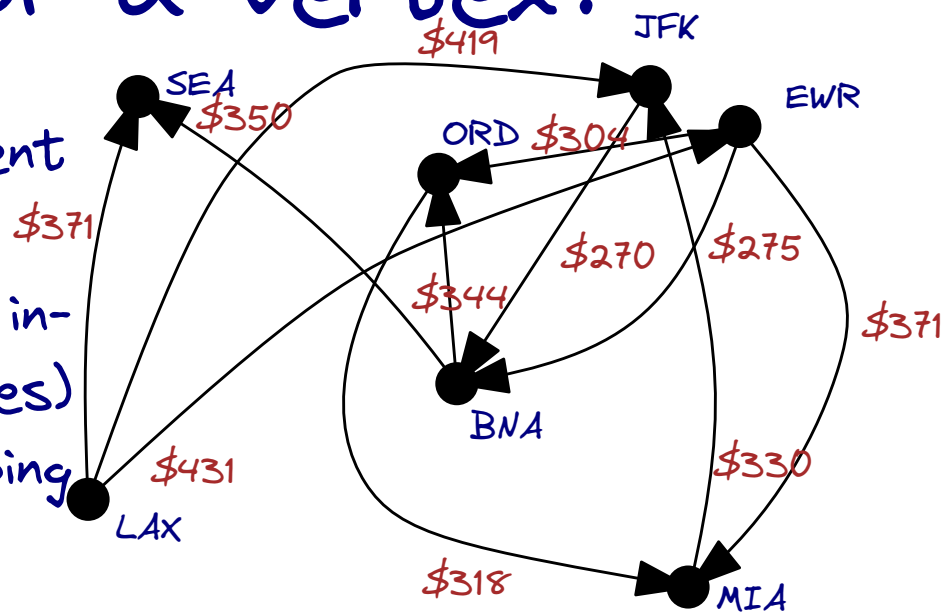
BNA is an IN-neighbor of ORD

ORD is adjacent to BNA.

DEGREE of a vertex?

is the number of edges adjacent to it.

For directed graphs, we have in-degree (number of incoming edges) and out-degree (number of outgoing edges).



EWR -> MIA 371, BNA 275, ORD 304
JFK -> BNA 270
MIA -> JFK 330
BNA -> ORD 344, SEA 350
ORD -> MIA 318
SEA ->
LAX -> SEA 371, JFK 419, EWR 431

Example: $deg^-(SEA) = 2, deg^+(SEA) = 0$

ORD is an OUT-neighbor of BNA

BNA is an IN-neighbor of ORD

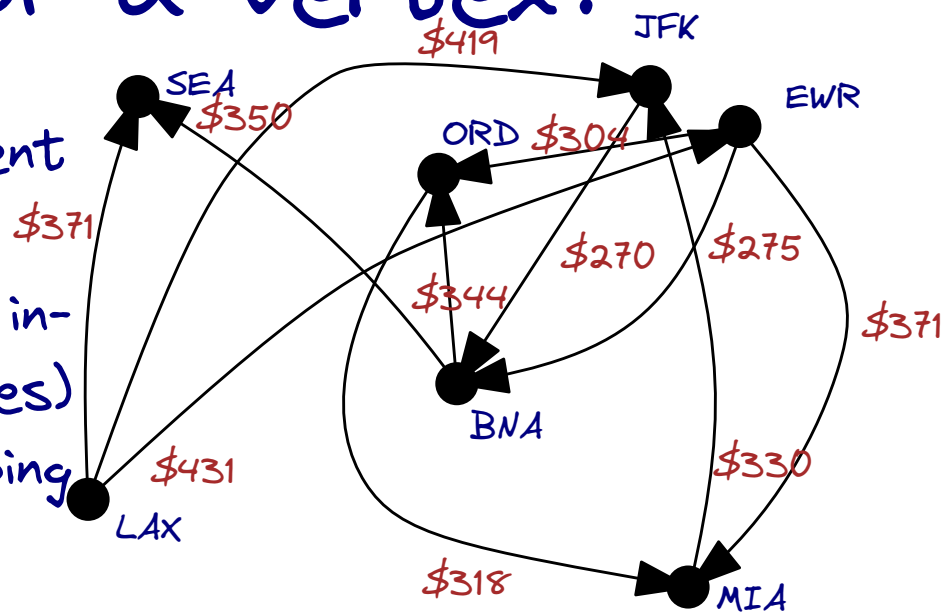
ORD is adjacent to BNA.

Question: Create a graph with the following degrees: $2, 2, 1, 3, 1$.

DEGREE of a vertex?

is the number of edges adjacent to it.

For directed graphs, we have in-degree (number of incoming edges) and out-degree (number of outgoing edges).



EWR -> MIA 371, BNA 275, ORD 304
JFK -> BNA 270
MIA -> JFK 330
BNA -> ORD 344, SEA 350
ORD -> MIA 318
SEA ->
LAX -> SEA 371, JFK 419, EWR 431

Example: $deg^-(SEA) = 2, deg^+(SEA) = 0$

ORD is an OUT-neighbor of BNA

BNA is an IN-neighbor of ORD

ORD is adjacent to BNA.

Question: Create a graph with the following degrees: $2, 2, 1, 3, 1$.

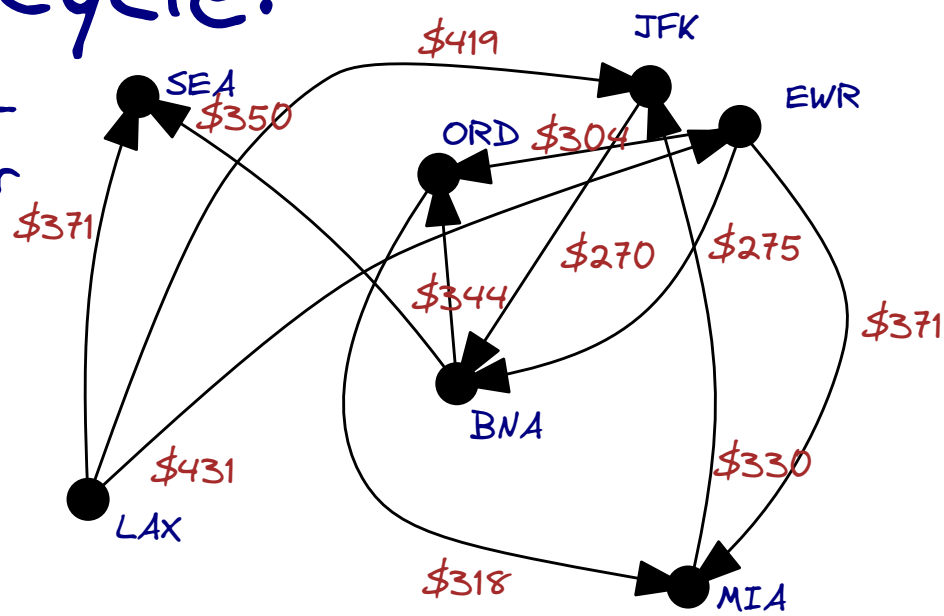
You can not! We will see why in a bit.

Path and Cycle?

A path is a sequence of vertices where each consecutive pair is adjacent.

Example: {BNA, ORD, MIA} is a path.

Example: {BNA, ORD, EWR, MIA} is NOT a path.



EWR -> MIA 371, BNA 275, ORD 304
JFK -> BNA 270
MIA -> JFK 330
BNA -> ORD 344, SEA 350
ORD -> MIA 318
SEA ->
LAX -> SEA 371, JFK 419, EWR 431

Path and Cycle?

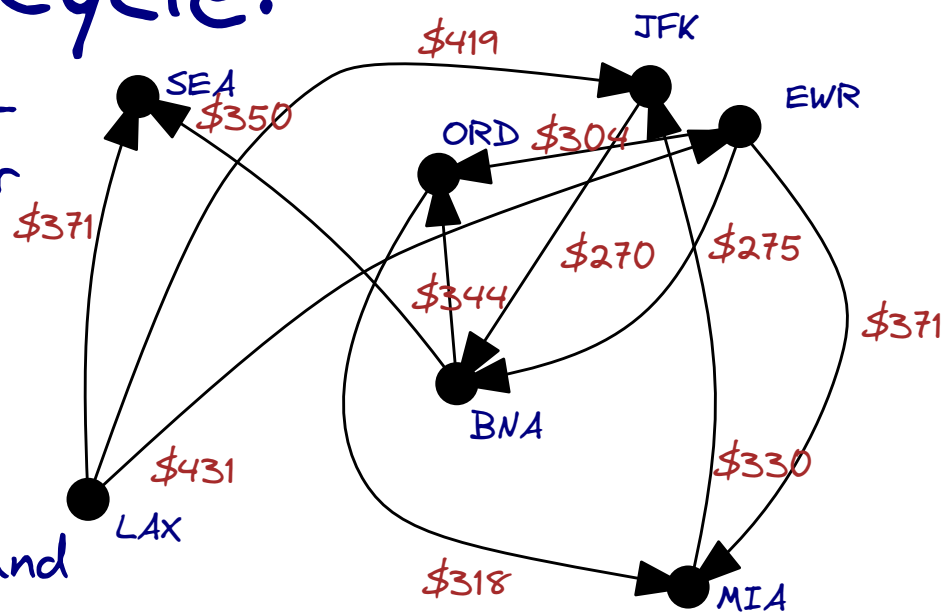
A path is a sequence of vertices where each consecutive pair is adjacent.

Example: {BNA, ORD, MIA} is a path.

Example: {BNA, ORD, EWR, MIA} is NOT a path.

A cycle is a path that starts and ends at the same vertex.

Example: {BNA, ORD, MIA, EWR, BNA} is a cycle.



EWR -> MIA 371, BNA 275, ORD 304
JFK -> BNA 270
MIA -> JFK 330
BNA -> ORD 344, SEA 350
ORD -> MIA 318
SEA ->
LAX -> SEA 371, JFK 419, EWR 431

Path and Cycle?

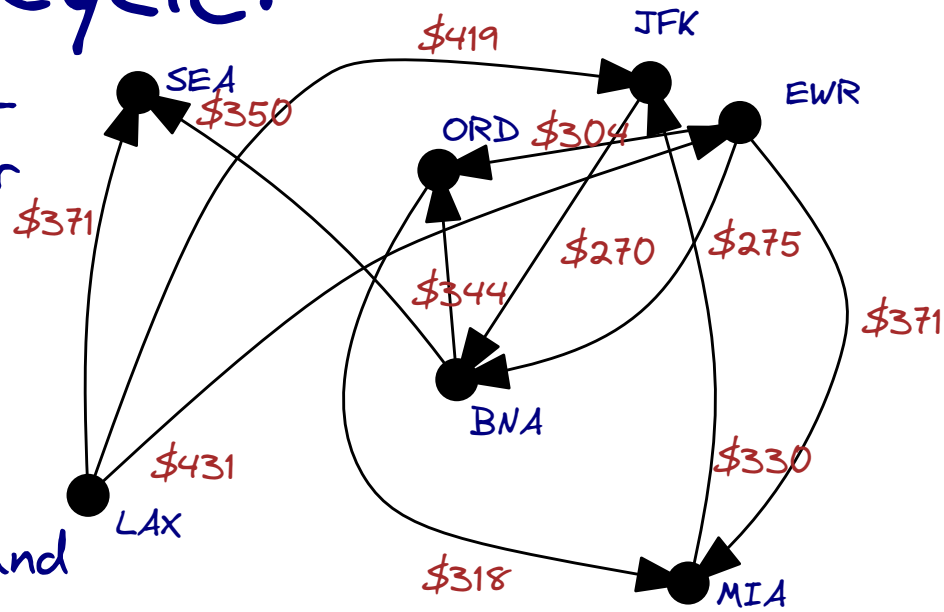
A path is a sequence of vertices where each consecutive pair is adjacent.

Example: {BNA, ORD, MIA} is a path.

Example: {BNA, ORD, EWR, MIA} is NOT a path.

A cycle is a path that starts and ends at the same vertex.

Example: {BNA, ORD, MIA, EWR, BNA} is a cycle.



EWR -> MIA 371, BNA 275, ORD 304
JFK -> BNA 270
MIA -> JFK 330
BNA -> ORD 344, SEA 350
ORD -> MIA 318
SEA ->
LAX -> SEA 371, JFK 419, EWR 431

A Direct Acyclic Graph (DAG) is directed graph that has no cycles.

Handshaking Lemma

- **Lemma:** The sum of degrees in an undirected graph is twice the number of edges. In particular, this must be even.

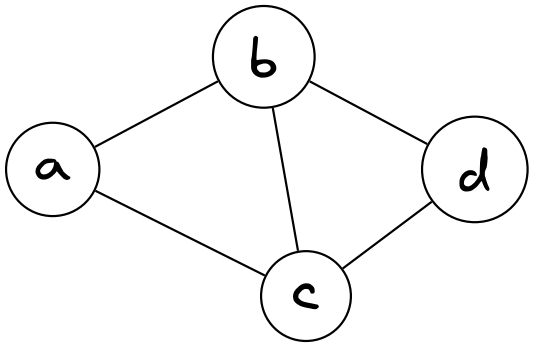
$$\sum_{v \in V} deg(v) = 2 \times |E|$$

Connectivity, Components, Trees, Forests

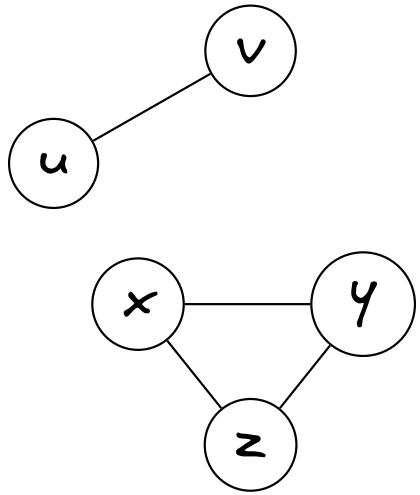
Connected vs Disconnected

- Connected (undirected): For every pair of vertices, there exists a path.

Connected



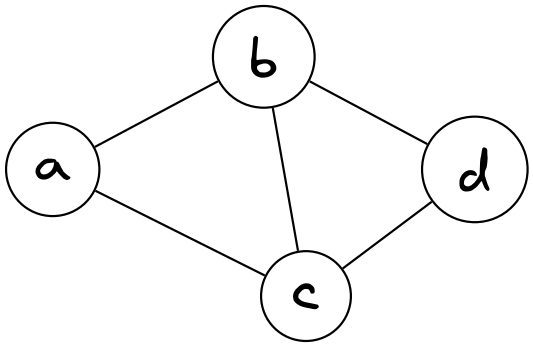
Disconnected



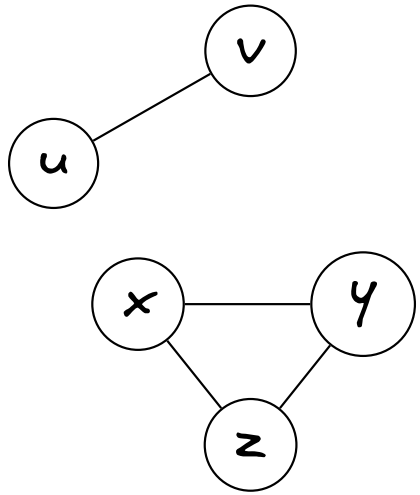
Connected vs Disconnected

- Connected (undirected): For every pair of vertices, there exists a path.
- Disconnected: There exist vertices with no path between them (multiple components).

Connected



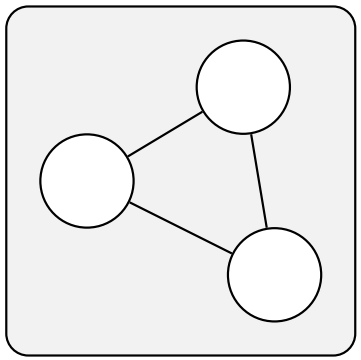
Disconnected



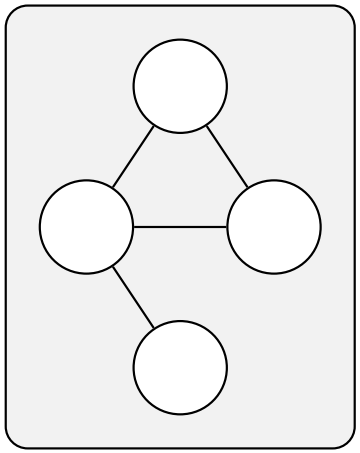
Connected Components

- Component: A maximal connected subgraph of an undirected graph.

C1



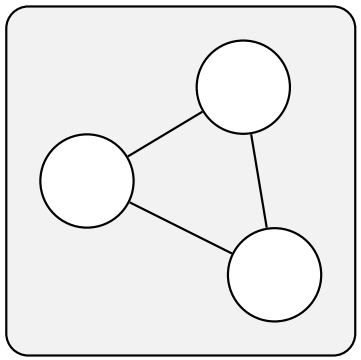
C2



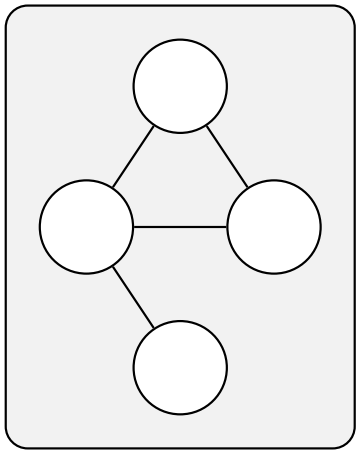
Connected Components

- Component: A maximal connected subgraph of an undirected graph.
- Components partition the vertex set; every vertex belongs to exactly one component.

C1

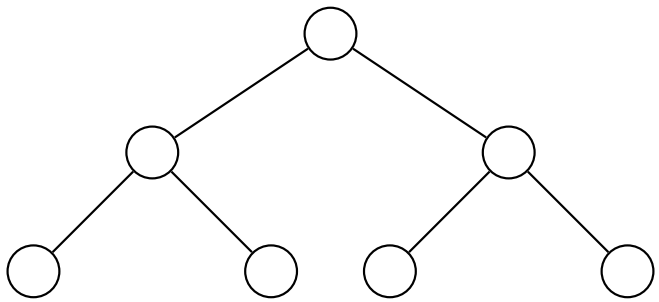


C2



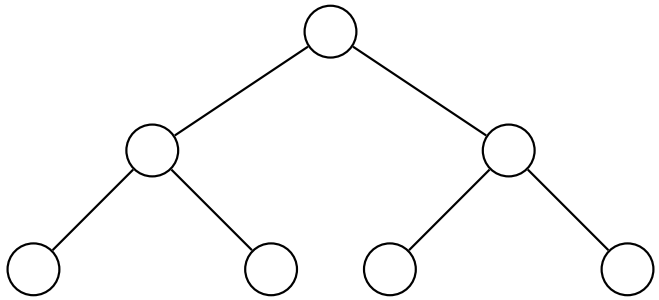
Trees

- Tree: A connected, acyclic undirected graph.



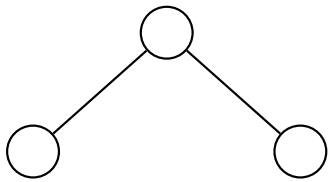
Trees

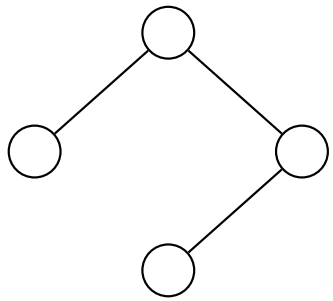
- Tree: A connected, acyclic undirected graph.
- Properties: With n vertices, a tree has $n-1$ edges; unique simple path between any two vertices.



Forests

- Forest: *A* disjoint union of trees (acyclic components).





Forests

- Forest: A disjoint union of trees (acyclic components).
- Every forest can be turned into a tree by adding edges between components.