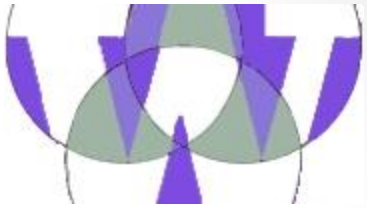
Introduction to Data Science

Lecture 9; December 7th, 2016

Ernst Henle
ErnstHe@UW.edu
Skype: ernst-henle

Agenda



- Announcements
  - Social Interactions on LinkedIn: Continue to discuss statistics, job prospects, etc.
- HIVE Lab with Hue, Impala, and File Browser
- MapReduce
- Break
- MapReduce Lab
- Break
- Graph Data Intro
- Quiz Graph Data (Measures)
- Graph Data Popularity
- Quiz Graph Data (Popularity)
- Google Matrix
- Assignment. See assignment slides at the end of the deck. Complete all assignments items from all assignment slides. Submit by this Saturday at 11:57 PM.

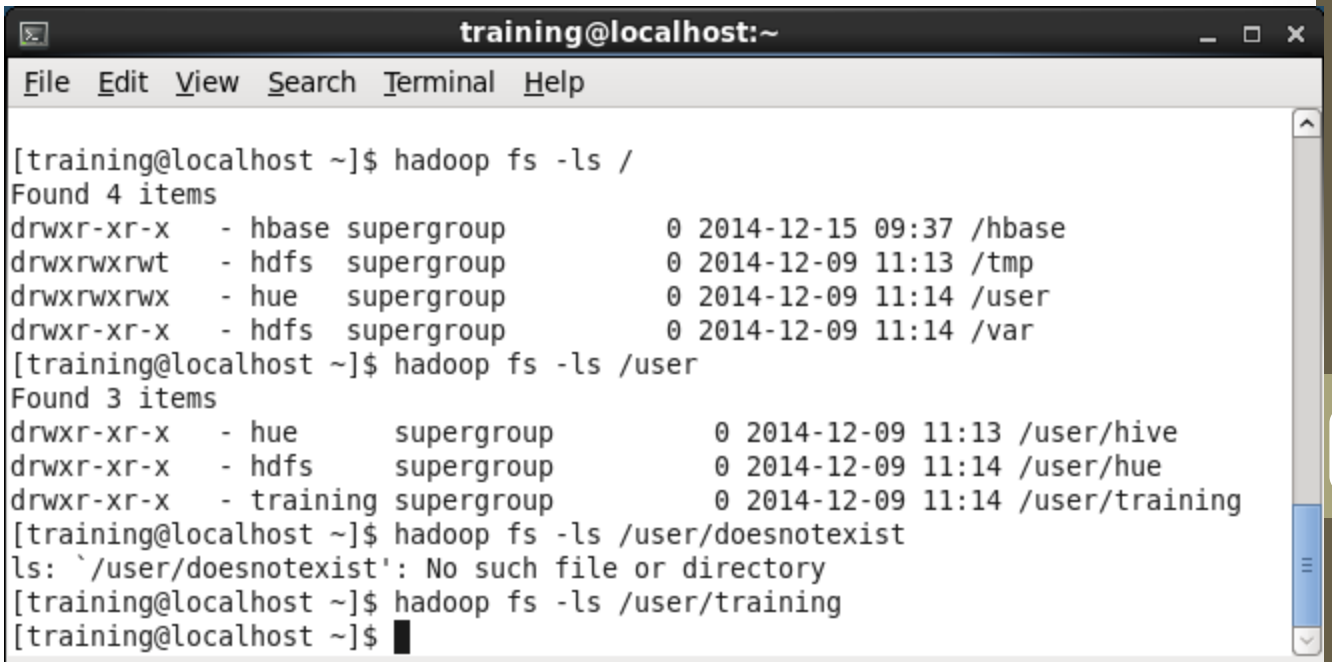
Hadoop-2 Labs

HDFS Lab (0)

"So you want to hire me as a Data Scientist for Intelligent Virtualized Deep Machine Learning Real-time Big Data in the Cloud for Social Networks? Ok, but if you also want Hadoop, increase my salary by 50%."

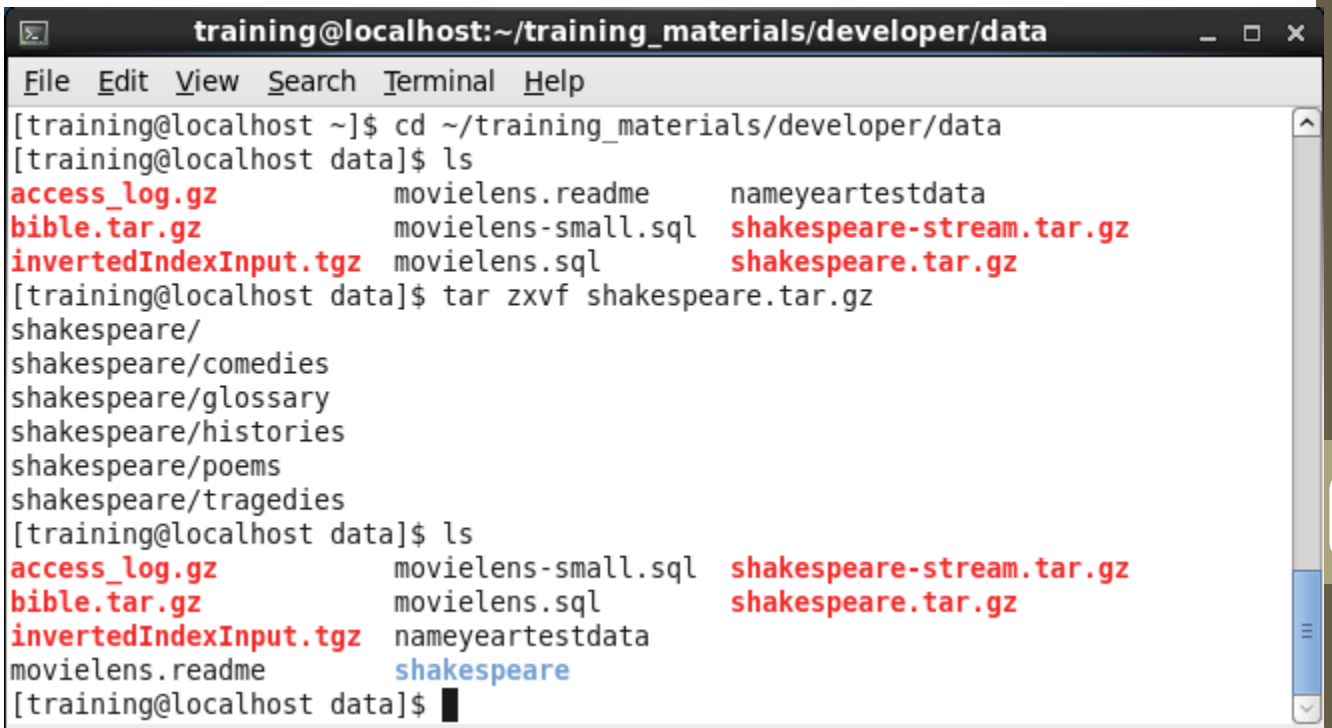
HDFS Lab (1)

- Enter these commands into the console to list some folders and files inside HDFS:
  - `$ hadoop fs -ls /`
  - `$ hadoop fs -ls /user`
  - `$ hadoop fs -ls /user/doesnotexist`
  - `$ hadoop fs -ls /user/training`
- "fs" stands for file system (HDFS). "ls" is an argument to list HDFS files.



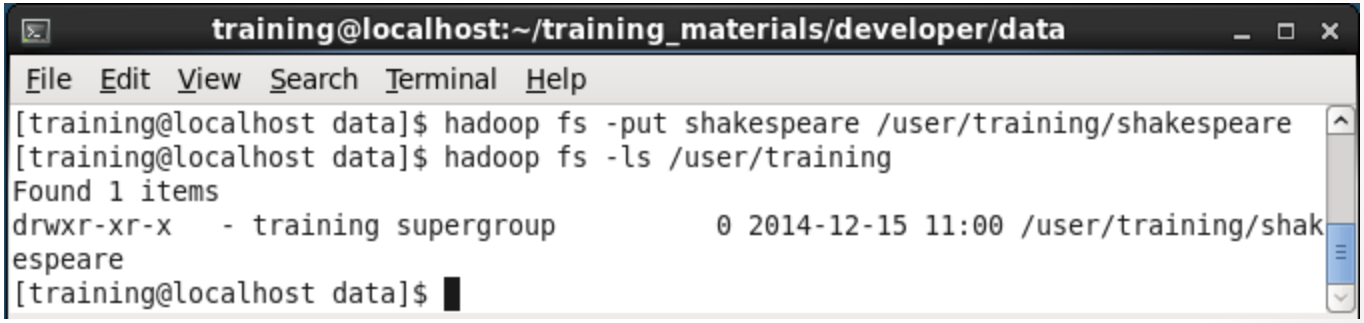
HDFS Lab (2)

- Enter these commands into the console to find and expand the shakespeare tar on the Linux OS :
  - $ **`cd ~/training_materials/developer/data`**
  - $ **`ls`**
  - $ **`tar zxvf shakespeare.tar.gz`**
  - $ **`ls`**



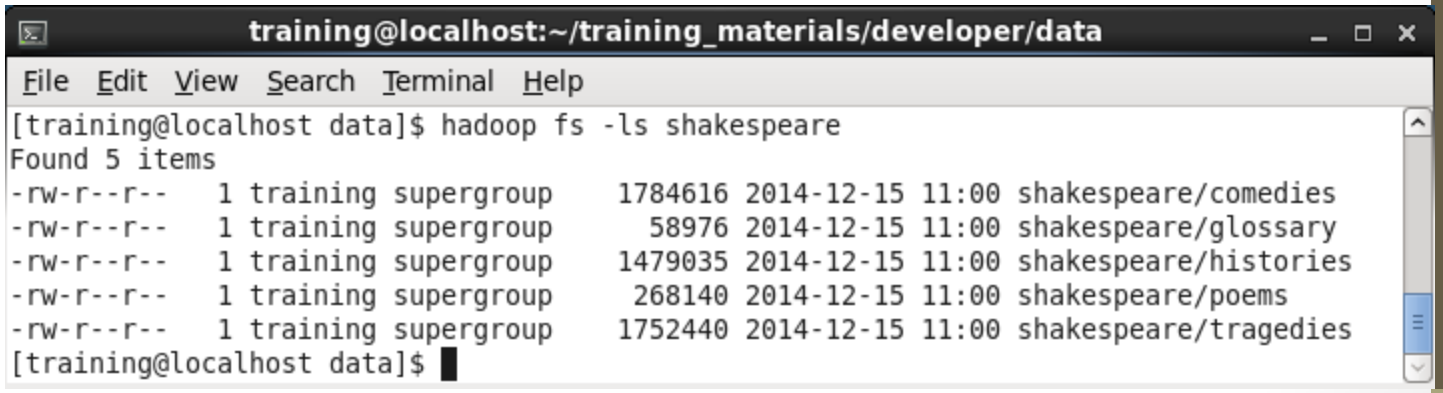
HDFS Lab (3)

- Enter these commands into the console to place the shakespeare directory into HDFS under training and then verify that the directory is in HDFS:
  - `$ hadoop fs -put shakespeare /user/training/shakespeare`
  - `$ hadoop fs -ls /user/training`



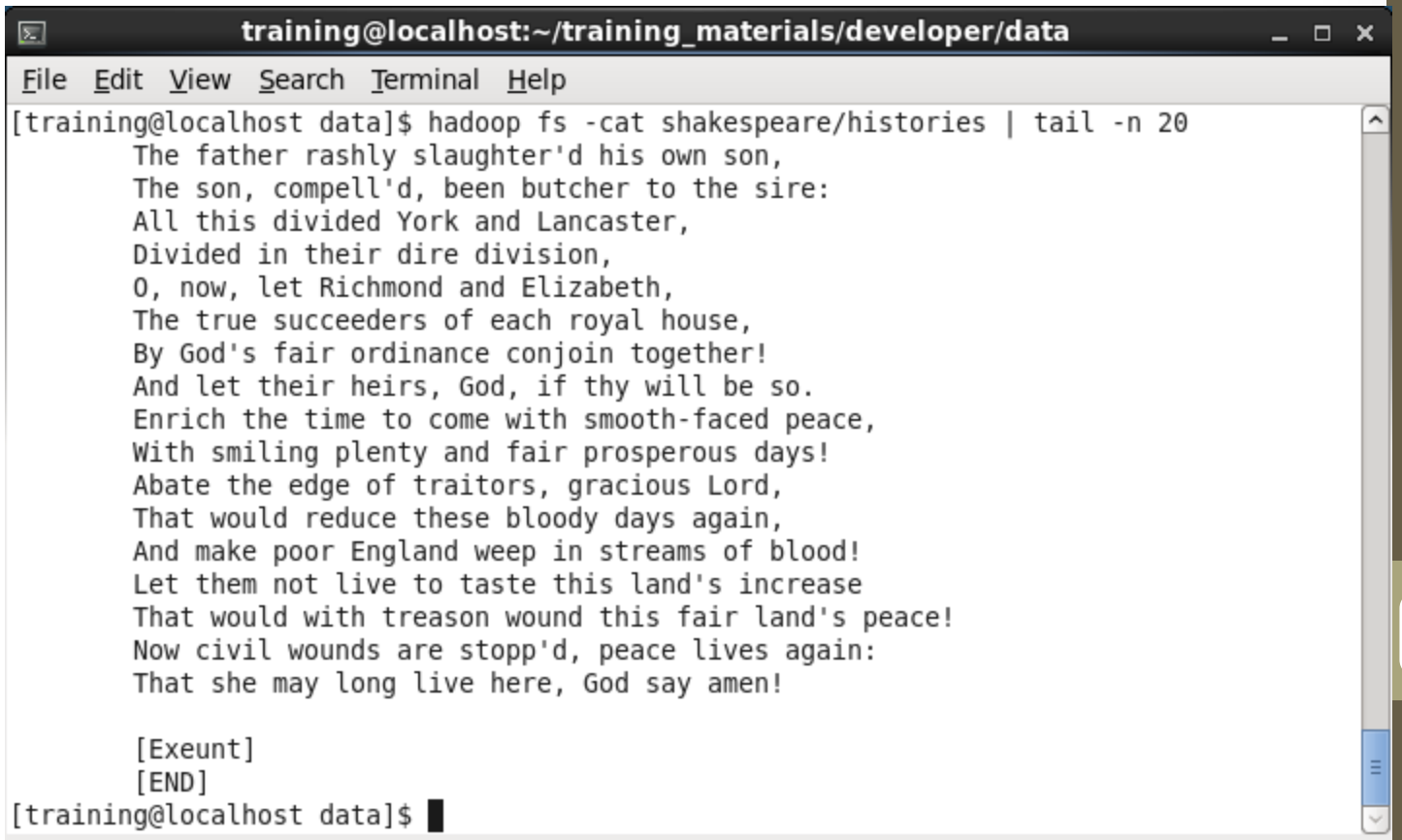
HDFS Lab (4)

- Enter this command into the console to list the 5 files in the shakespeare directory of HDFS:
  - `$ hadoop fs -ls /user/training/shakespeare`



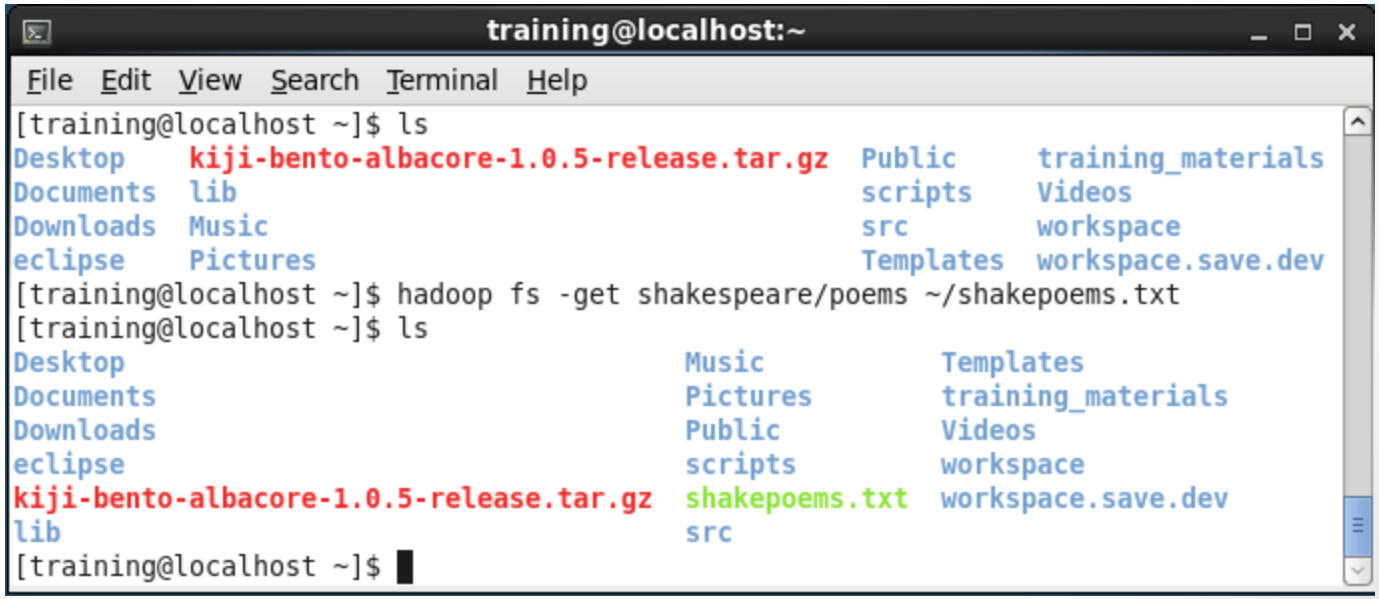
HDFS Lab (5)

- Enter this command into the console to read the last 20 lines of the histories file in HDFS:
  - `$ hadoop fs -cat shakespeare/histories | tail -n 20`



HDFS Lab (6)

- Enter the following commands to retrieve files from HDFS and list them in the Linux OS:
  - $ cd ~
  - $ **ls**
  - $ **hadoop fs -get shakespeare/poems ~/shakepoems.txt**
  - $ **ls**



Sqoop Lab (0)

- **Sq**l + Had**oop** → **Sqoop**
- Use Sqoop to share between a RDBMS and Hadoop

Sqoop Lab (1)

- Use this Sqoop command to familiarize yourself with Sqoop:
  - `$ sqoop help`

- Use these sqoop commands to list mysql databases on localhost and then tables in one of those databases:
  - `$ sqoop list-databases --connect jdbc:mysql://localhost --username training --password training`
  - `$ sqoop list-tables --connect jdbc:mysql://localhost/movielens --username training --password training`

Sqoop Lab (2)

- Use this Sqoop command to import the movie table from mysql to an HDFS file. The fields of a table record (row) will be separated by tabs in the corresponding record of the HDFS file:
  - $ **sqoop import --connect jdbc:mysql://localhost/movielens --username training --password training --fields-terminated-by '\t' --table movie**

- Note that Sqoop constructs some SQL statements for MySQL and that it converts the SQL statement into a MapReduce job that only has a Map component:
  - **Test SQL: SELECT t.* FROM `movie` AS t LIMIT 1**
  - **SELECT MIN(`id`), MAX(`id`) FROM `movie`**

- Use this HDFS command to list the newly imported file(s) and then view the last part of one of the files:
  - $ **hadoop fs -ls movie**
  - $ **hadoop fs -tail movie/part-m-00000**

To run this command a second time you will need to avoid a naming collision:
sqoop import --connect jdbc:mysql://localhost/movielens --username training --password training --fields-terminated-by '\t' --table movie --target-dir /movie2

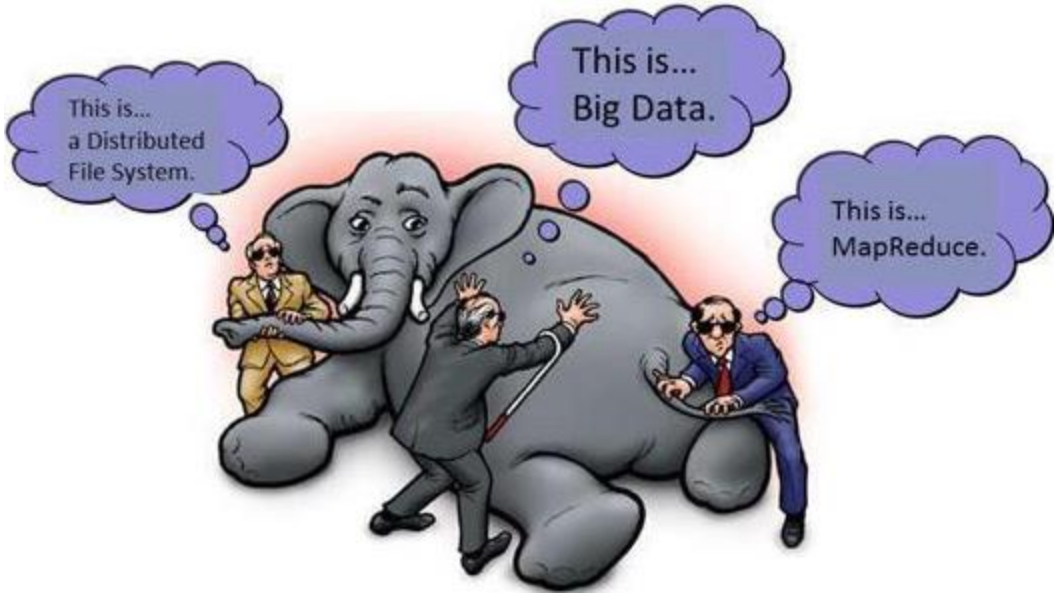
Sqoop Lab (3)

- Use this Sqoop command to import the movierating table from mysql to an HDFS file. The fields of a table record (row) will be separated by tabs in the corresponding record of the HDFS file:
  - $ **sqoop import --connect jdbc:mysql://localhost/movielens --username training --password training --fields-terminated-by '\t' --table movierating**

- Use these HDFS commands to list the newly imported file(s) and then view the last part of one of the files:
  - $ **hadoop fs -ls movierating**
  - $ **hadoop fs -tail movierating/part-m-00000**

To run this command a second time you will need to avoid a naming collision:
sqoop import --connect jdbc:mysql://localhost/movielens --username training --password training --fields-terminated-by '\t' --table movierating --target-dir /movierating2

Break


This is...
a Distributed
File System.
This is...
Big Data.
This is...
MapReduce.

Hive Lab (0)

"It does look similar—but this one is powered by Hadoop"

Hive Lab (1)

- Check that we have movie and movierating in HDFS
  - `$ hadoop fs -cat /movie/part-m-00000 | head`
  - `$ hadoop fs -cat /movierating/part-m-00000 | head`
- Start Hive
  - `$ hive`

- Tell Hive that the HDFS files, movie and movierating, are tables:
  - `hive> CREATE EXTERNAL TABLE movie (id INT, name STRING, year INT) ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' LOCATION '/user/training/movie';`
  - `hive> CREATE EXTERNAL TABLE movierating (userid INT, movieid INT, rating INT) ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' LOCATION '/user/training/movierating';`

To run this command a second time you will need to avoid a naming collision:
hive> DROP TABLE **movie**
hive> DROP TABLE **movie**rating

Hive Lab (2)

- Ask Hive to show tables and provide metadata on these tables
  - `hive> SHOW TABLES;`
  - `hive> DESCRIBE movie;`
  - `hive> DESCRIBE movierating;`
- Use Hive to find information from your tables
  - `hive> SELECT * FROM movie WHERE year < 1925;`
  - `hive> SELECT * FROM movie WHERE year < 1925 AND year != 0 ORDER BY name;`
  - `hive> SELECT * FROM movierating WHERE userid=149;`
  - `hive> SELECT AVG(rating) FROM movierating WHERE userid=149;`
  - `hive> SELECT userid, COUNT(userid),AVG(rating) FROM movierating where userid < 10 GROUP BY userid;`

Hive Lab (3)

- Use Hive to create a new table
  - `hive> CREATE TABLE MovieRated (MovieID INT, NumRatings INT, AvgRating FLOAT);`
  - `hive> insert overwrite table MovieRated SELECT movieid,COUNT(movieid),AVG(rating) FROM movierating GROUP BY movieid;`
- Query the new table
  - `hive> SELECT COUNT(*) FROM MovieRated;`
  - `hive> SELECT * FROM MovieRated WHERE AvgRating > 4.4 and NumRatings > 1000;`

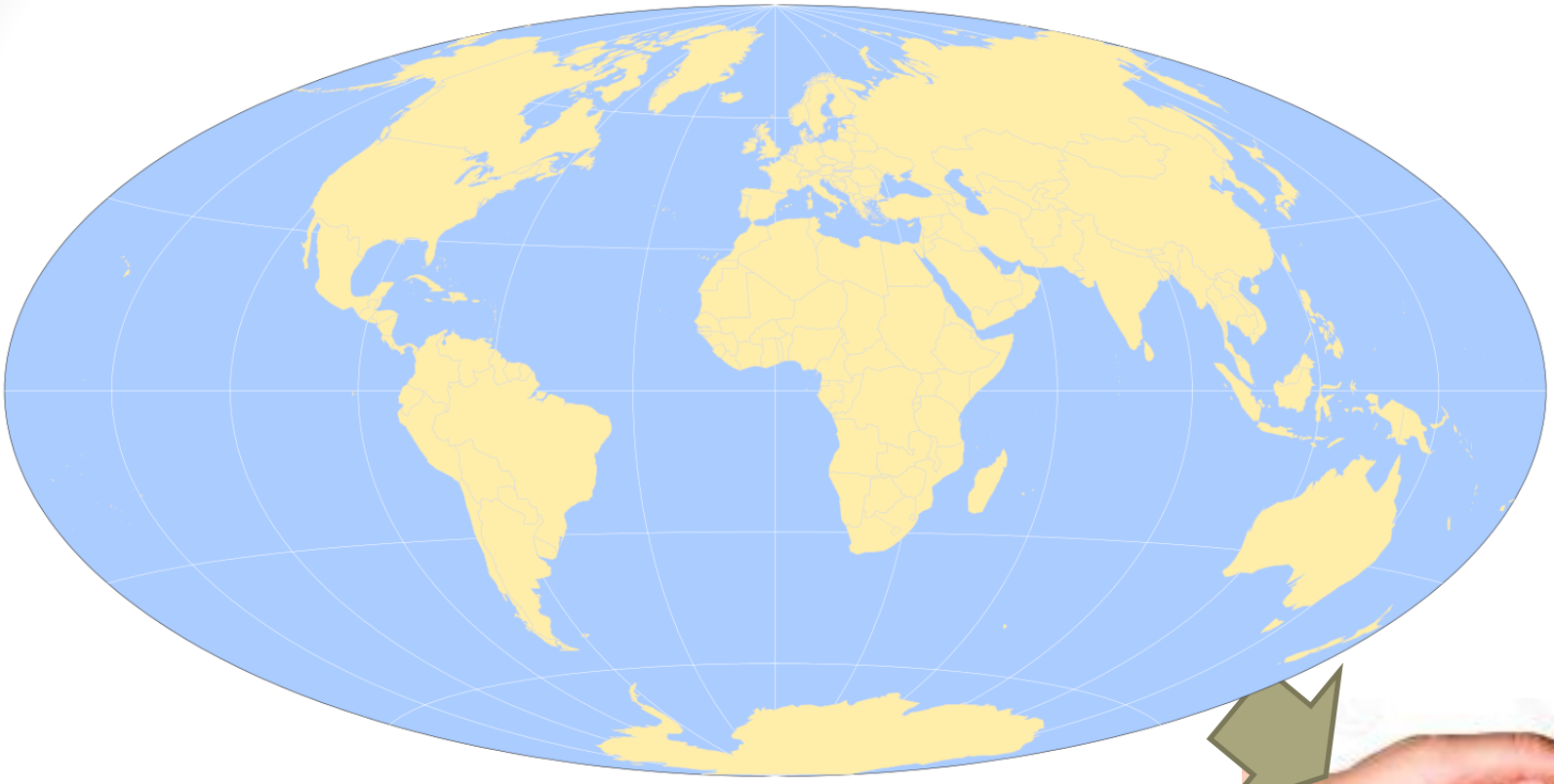
Hive Lab (4)

- Query with Join
  - **`hive> select name, NumRatings , AvgRating from MovieRated join movie on MovieRated.movieid=movie.id where AvgRating > 4.4 and NumRatings > 500 ORDER BY AvgRating;`**
- Quit Hive
  - **`hive> QUIT;`**
- Take-home Exercise
  - What is the name of the oldest movie in the database that has a top rating?

Hive Lab (5) Impala

- Use Impala to execute “HiveQL”
  1. Open Firefox
  2. Start Hue by clicking on the Hue link in the Bookmarks toolbar (Username: training, Password: training)
  3. Select Impala and Query Editor Tab
  4. Enter Query into Query text box for Impala:
     - `select name, NumRatings , AvgRating from MovieRated join movie on MovieRated.movieid=movie.id where NumRatings > 500 ORDER BY AvgRating DESC limit 20;`
  5. Click on Execute

MapReduce (0)



MapReduce (1)

- The purpose of Hadoop is to facilitate scale-out computing.
- MapReduce is Hadoop's answer to scale-out processing (Like HDFS is Hadoop's answer to scale-out persistence).
- The ideas behind distributed processing:
  - Send the program to the data as opposed to the data to the program
  - Make use of distributed data where each chunk of data already has its own CPUs
  - Allow independent, asynchronous processing of data
- MapReduce is a pattern (programming model) designed to make coding of distributed processes easy.

MapReduce (2)

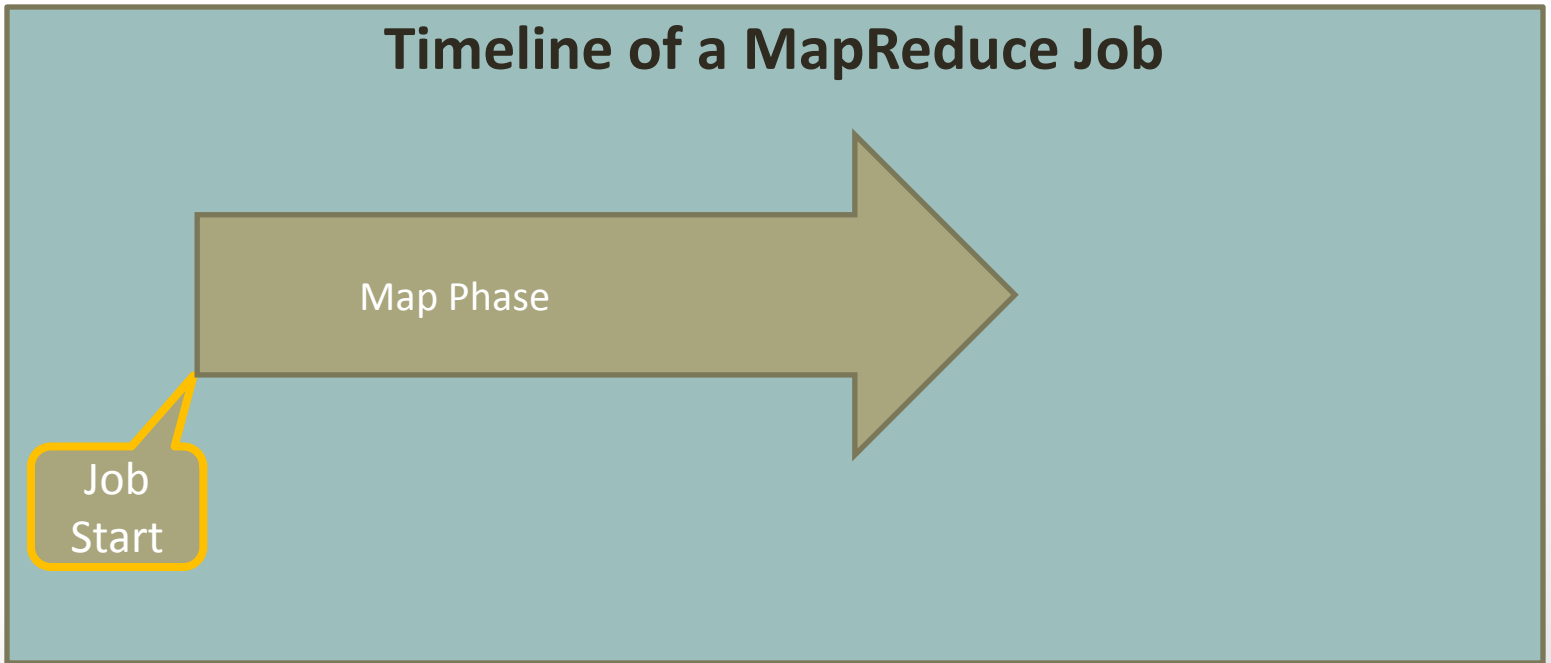
- The word MapReduce is made from **Map** and **Reduce**
- When a MapReduce job is started, then Hadoop sends **Map** and **Reduce** jobs to the data nodes. Hadoop manages all the details of data passing among data nodes.
  - The **Map** portion of the computation occurs on the individual data nodes where the data already reside. There is no need for the data to travel.
  - The **Reduce** portion of the computation occurs on some of the data nodes (not necessarily where the data reside). In some cases part of the **Reduce** operation can occur on the data node where the data already reside.

MapReduce (3)

- MapReduce uses three stages:
  1. Map
  2. Shuffle
  3. Reduce
- Then, why don't we call MapReduce: MapShuffleReduce?
  - Because: In most cases, the programmer need only be concerned with Map and Reduce. The Shuffle and Sorting is taken care of by Hadoop.
  - To achieve scalable programs, Hadoop takes care off:
    - Creating tasks on the various data nodes
    - Tracking Tasks
    - Moving data among data nodes
    - File I/O, networking, process synchronization, recovery from failures, re-running jobs, splitting input, Moving Key-Value-Pairs from Map to Reduce
    - Outputs results

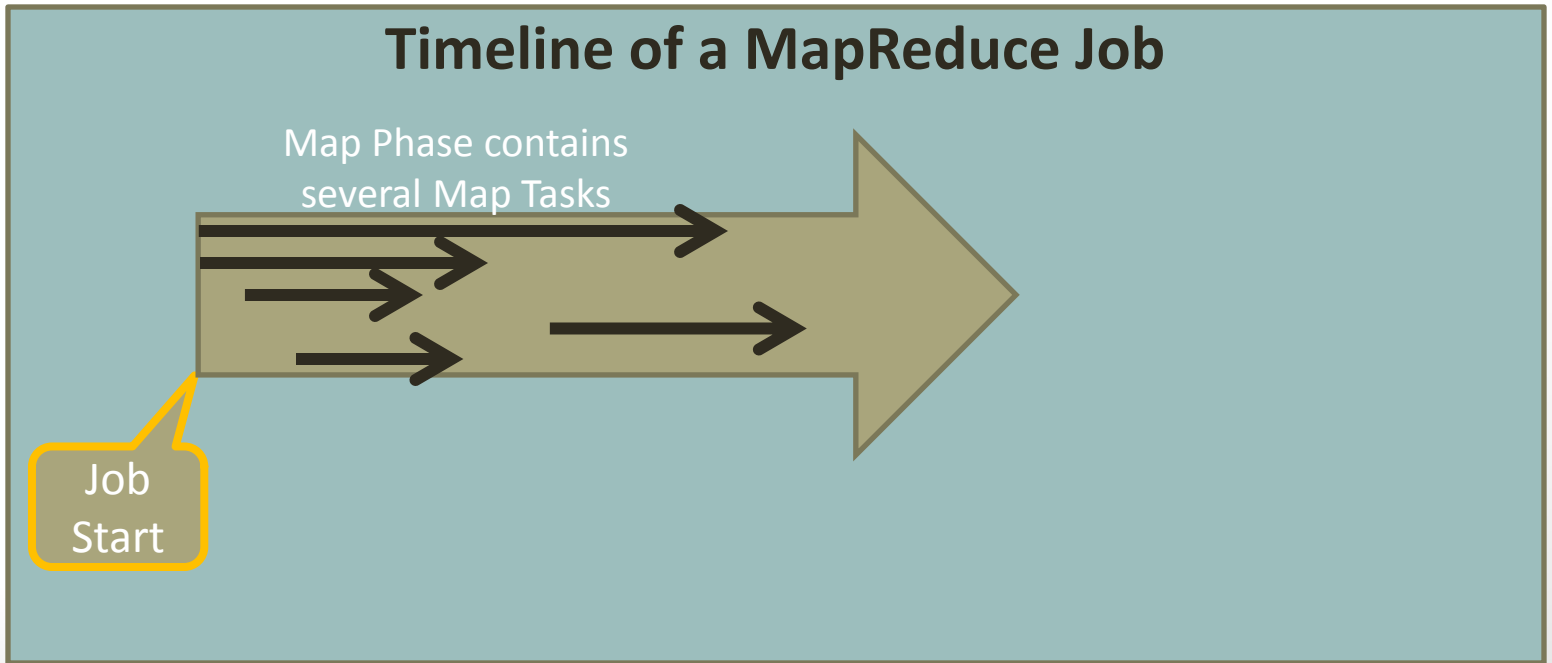
MapReduce (4)

- Parallelization Rules in MapReduce



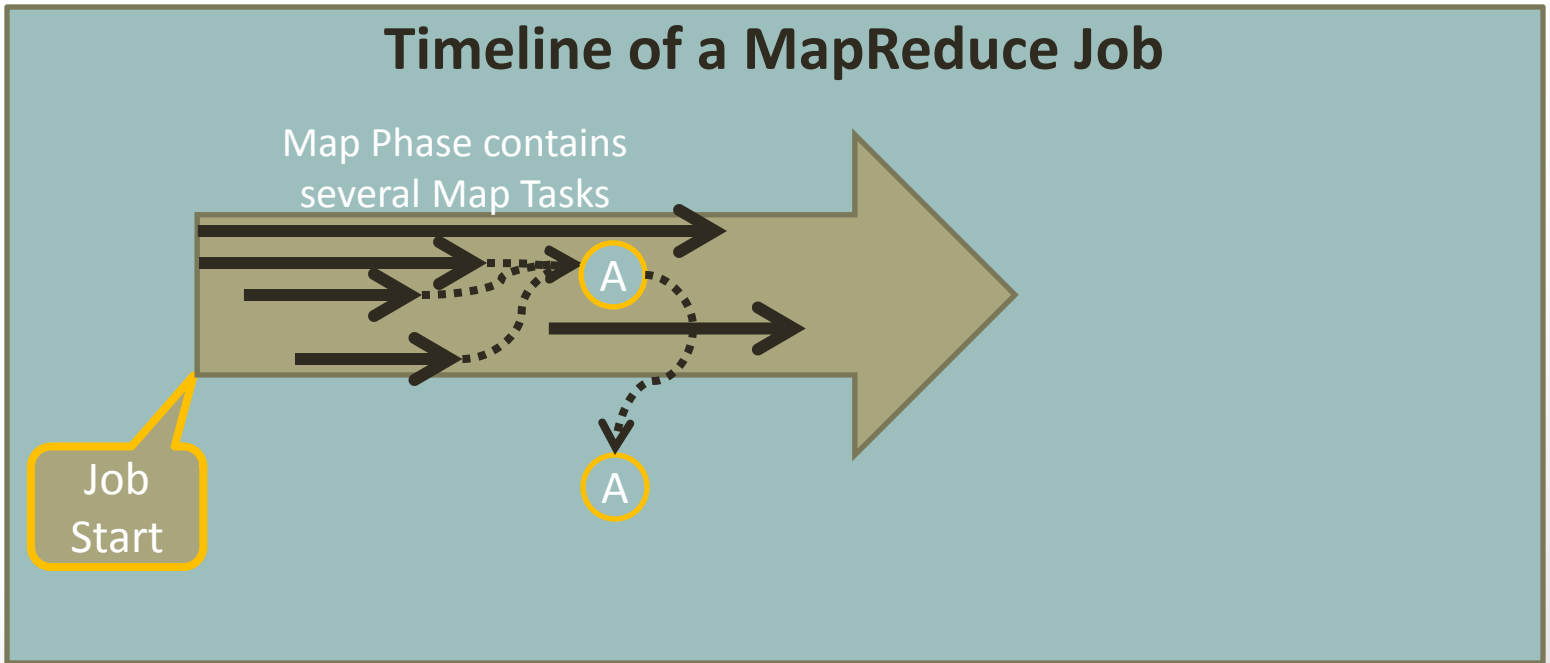
MapReduce (5)

- Parallelization Rules in MapReduce
  - Map tasks occur in parallel independent of each other



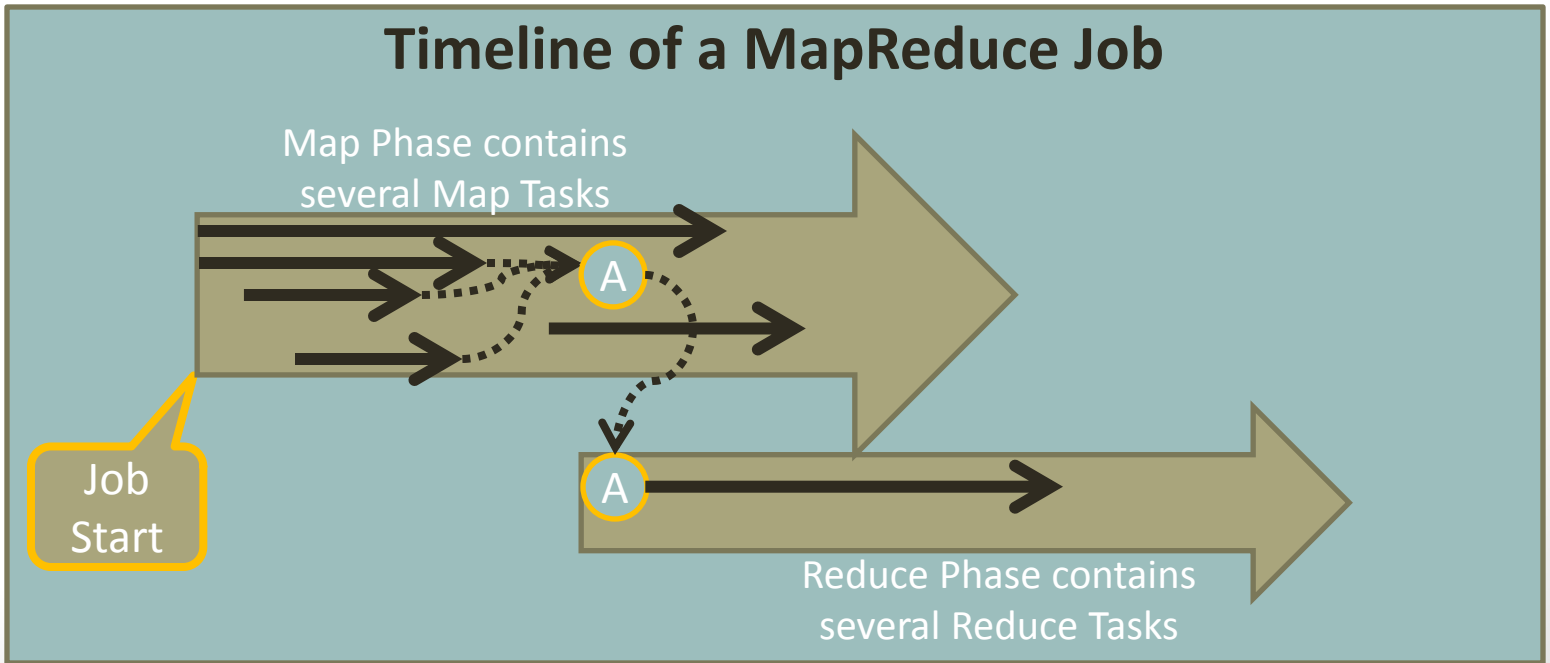
MapReduce (6)

- Parallelization Rules in MapReduce
  - Map tasks occur in parallel independent of each other
  - Map tasks terminate in “Shuffle and Sort”



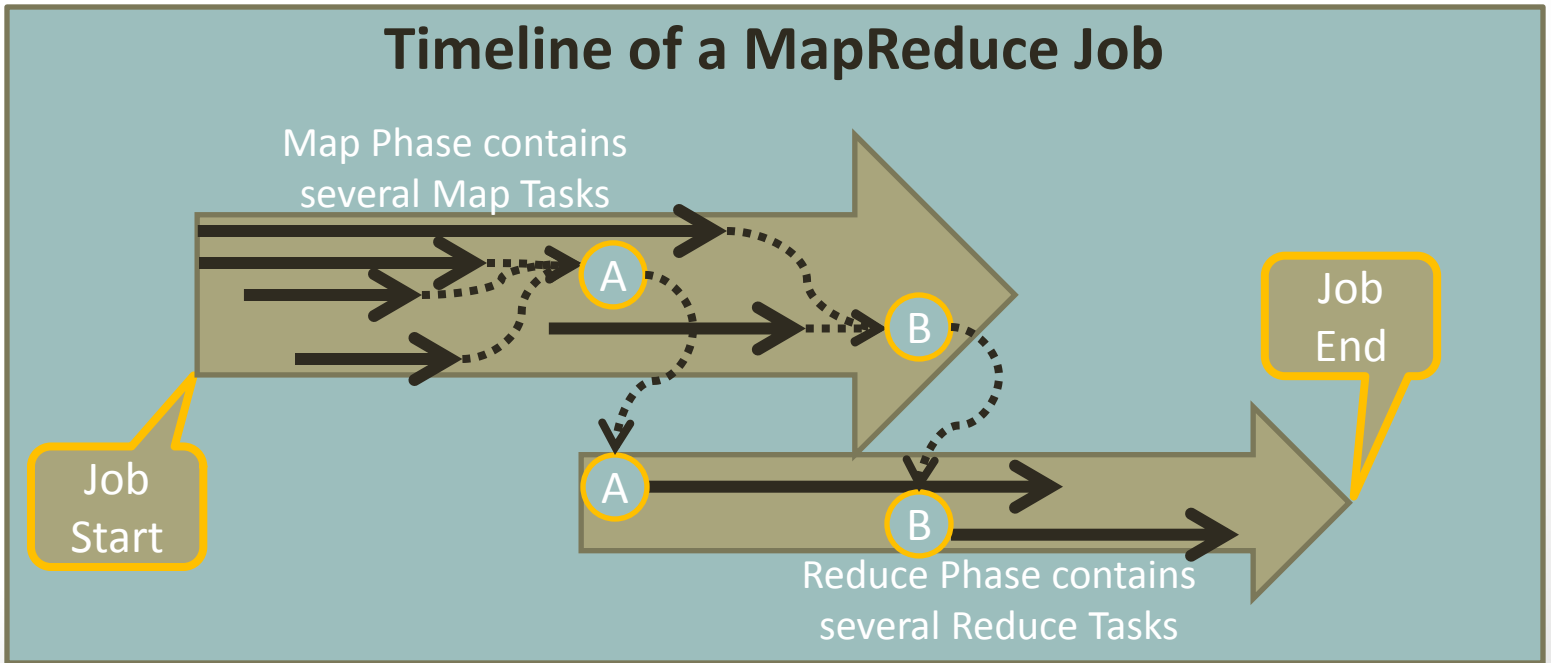
MapReduce (7)

- Parallelization Rules in MapReduce
  - Map tasks occur in parallel independent of each other
  - Map tasks terminate in “Shuffle and Sort”
  - Some Reduce tasks can start before all Mappers are done.



MapReduce (8)

- Parallelization Rules in MapReduce
  - Map tasks occur in parallel independent of each other
  - Map tasks terminate in “Shuffle and Sort”
  - Some Reduce tasks can start before all Mappers are done.
  - Reduce tasks occur in parallel independent of each other
  - Reducers cannot complete before all Mapping and Shuffle & Sort have completed.

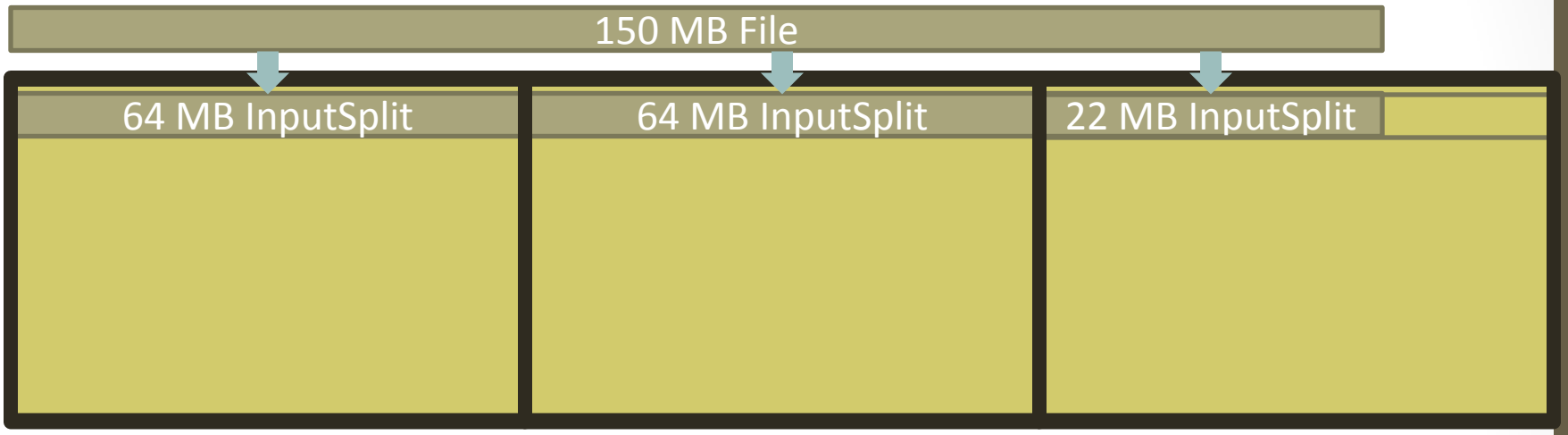


MapReduce (9)

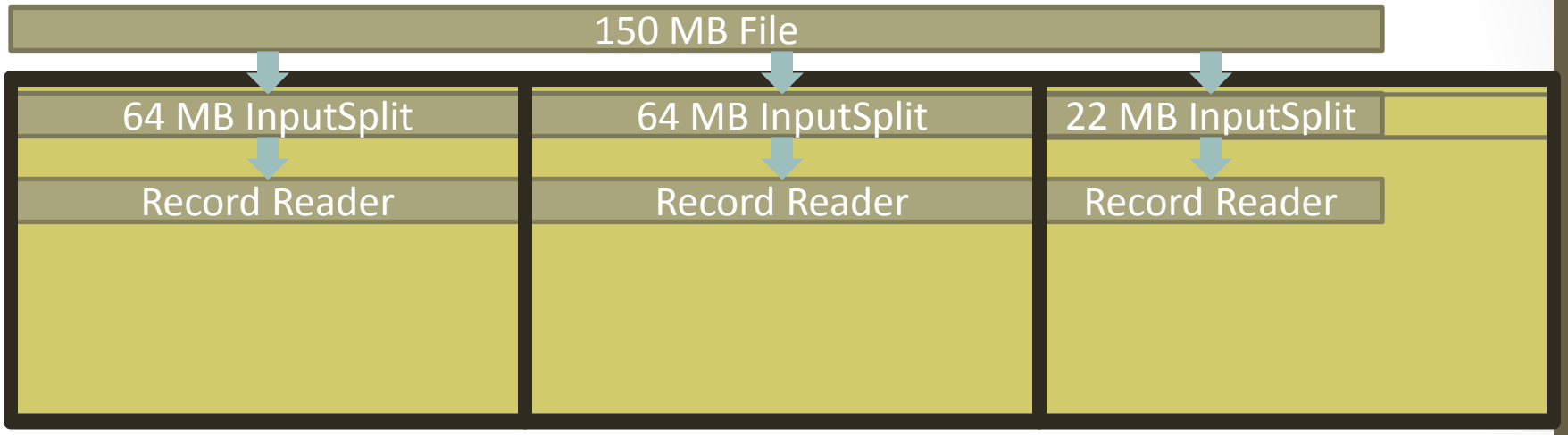
150 MB File

MapReduce (10)

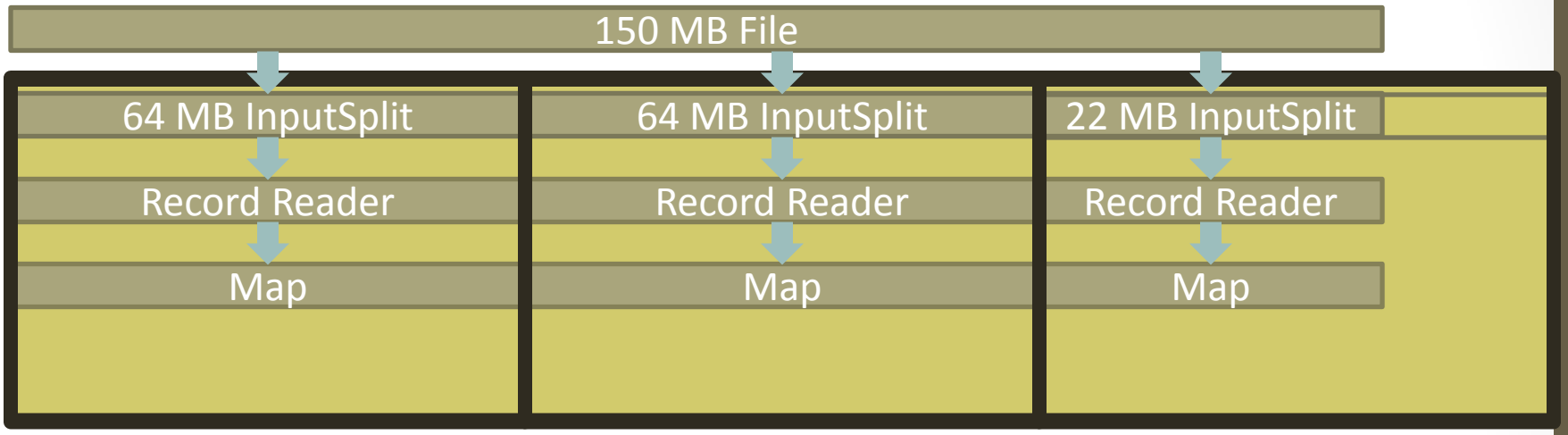
MapReduce (11)



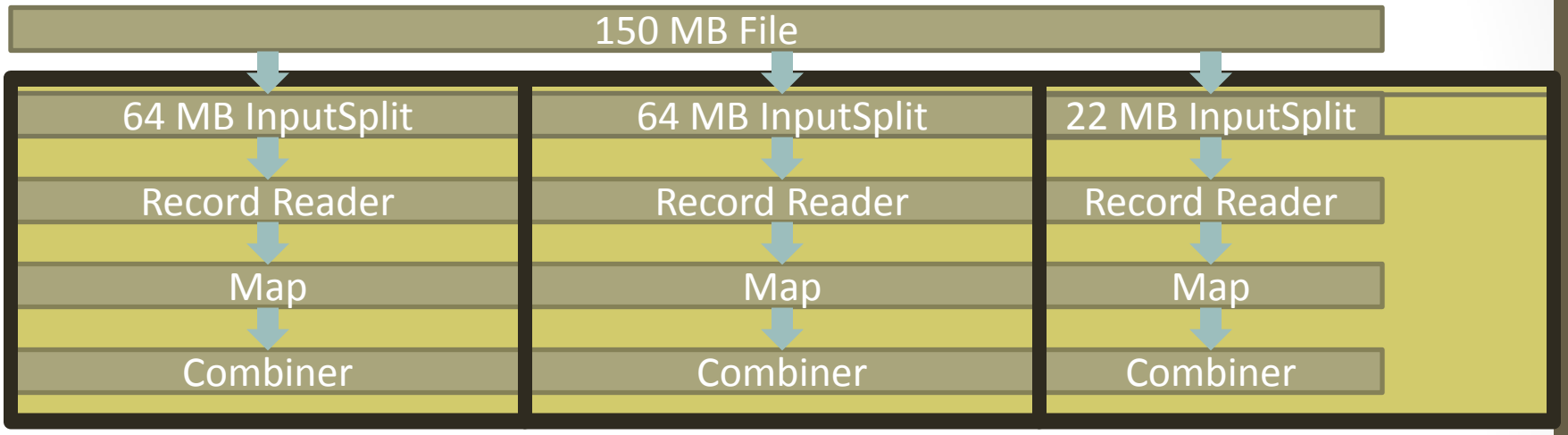
MapReduce (12)



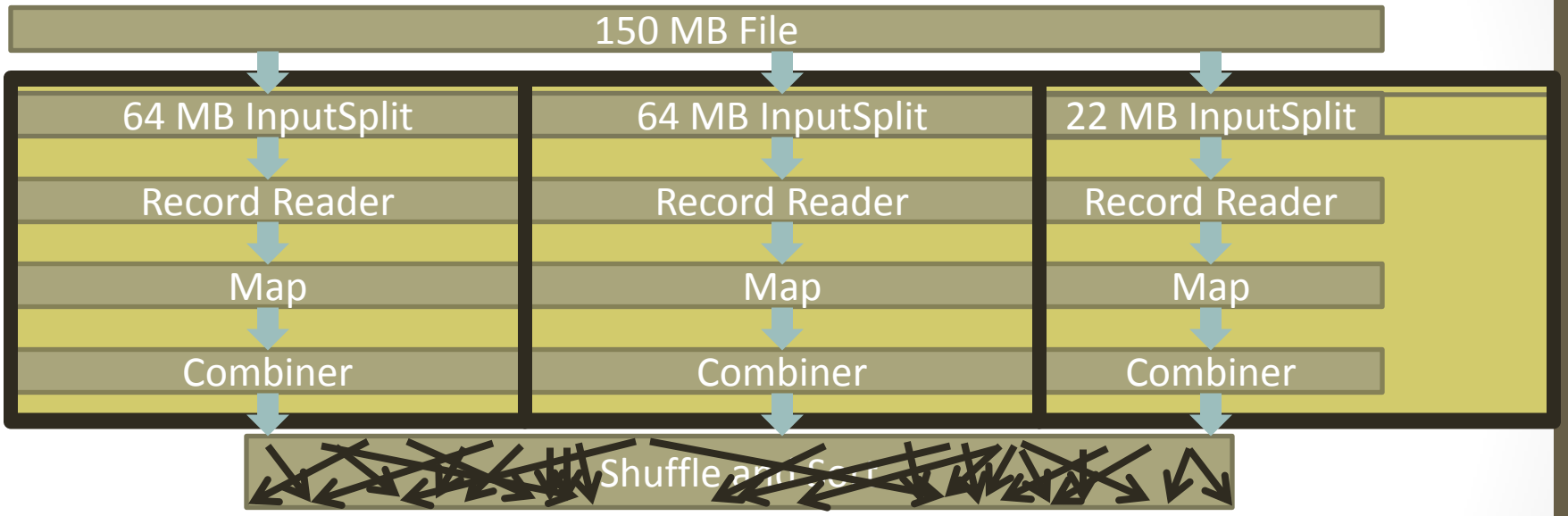
MapReduce (13)



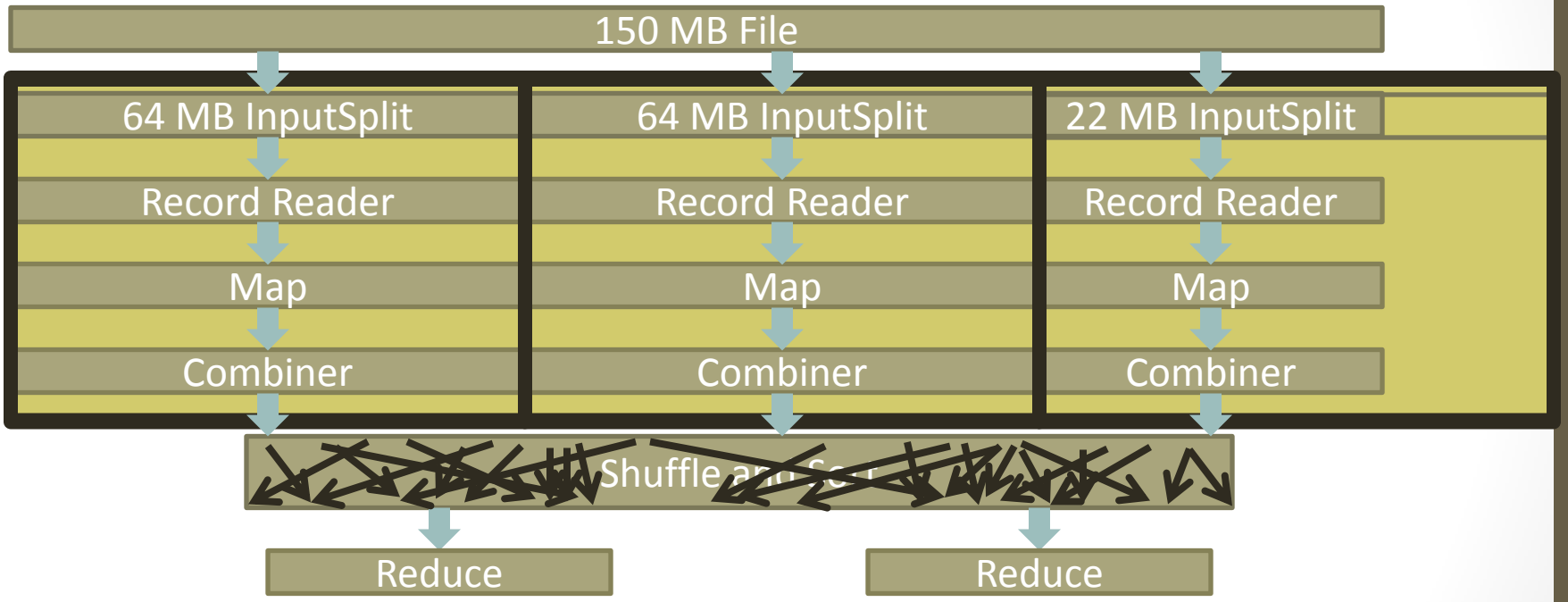
MapReduce (14)



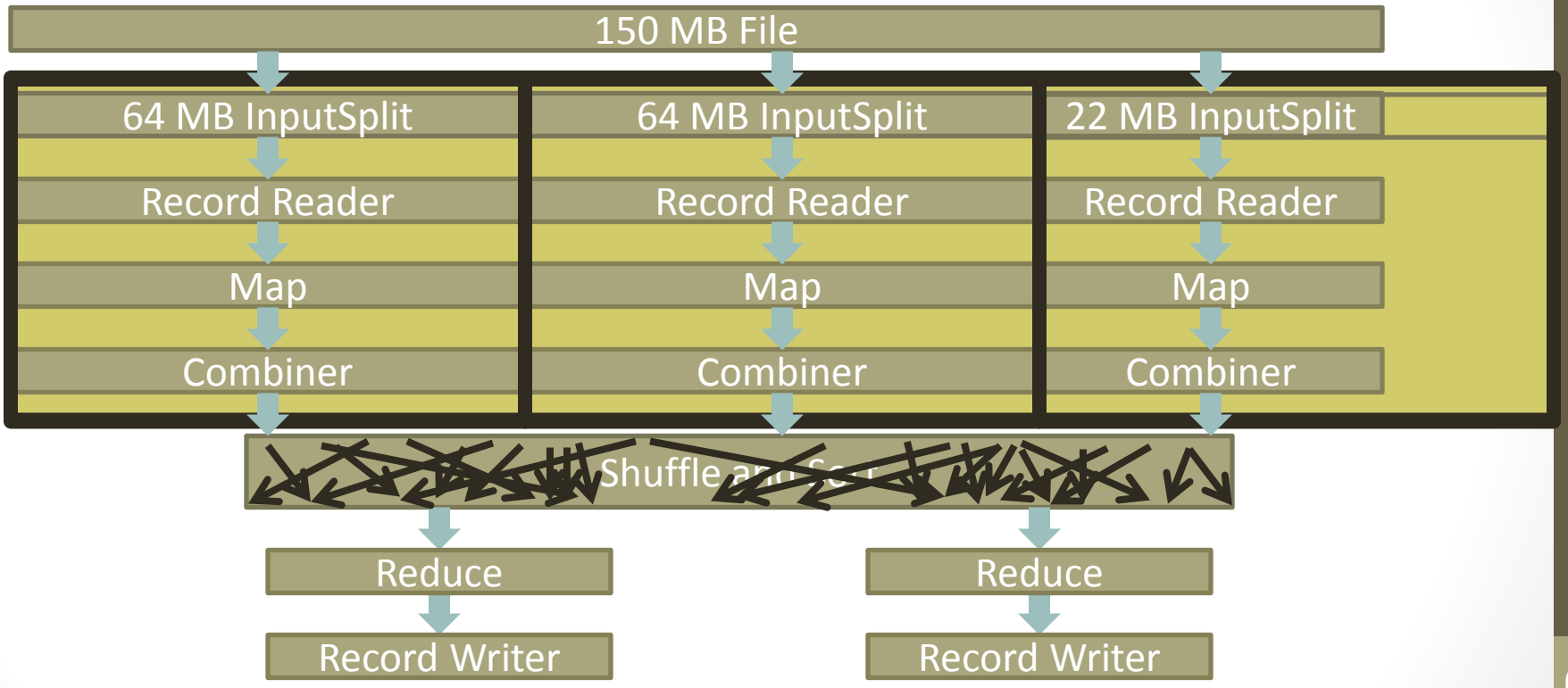
MapReduce (15)



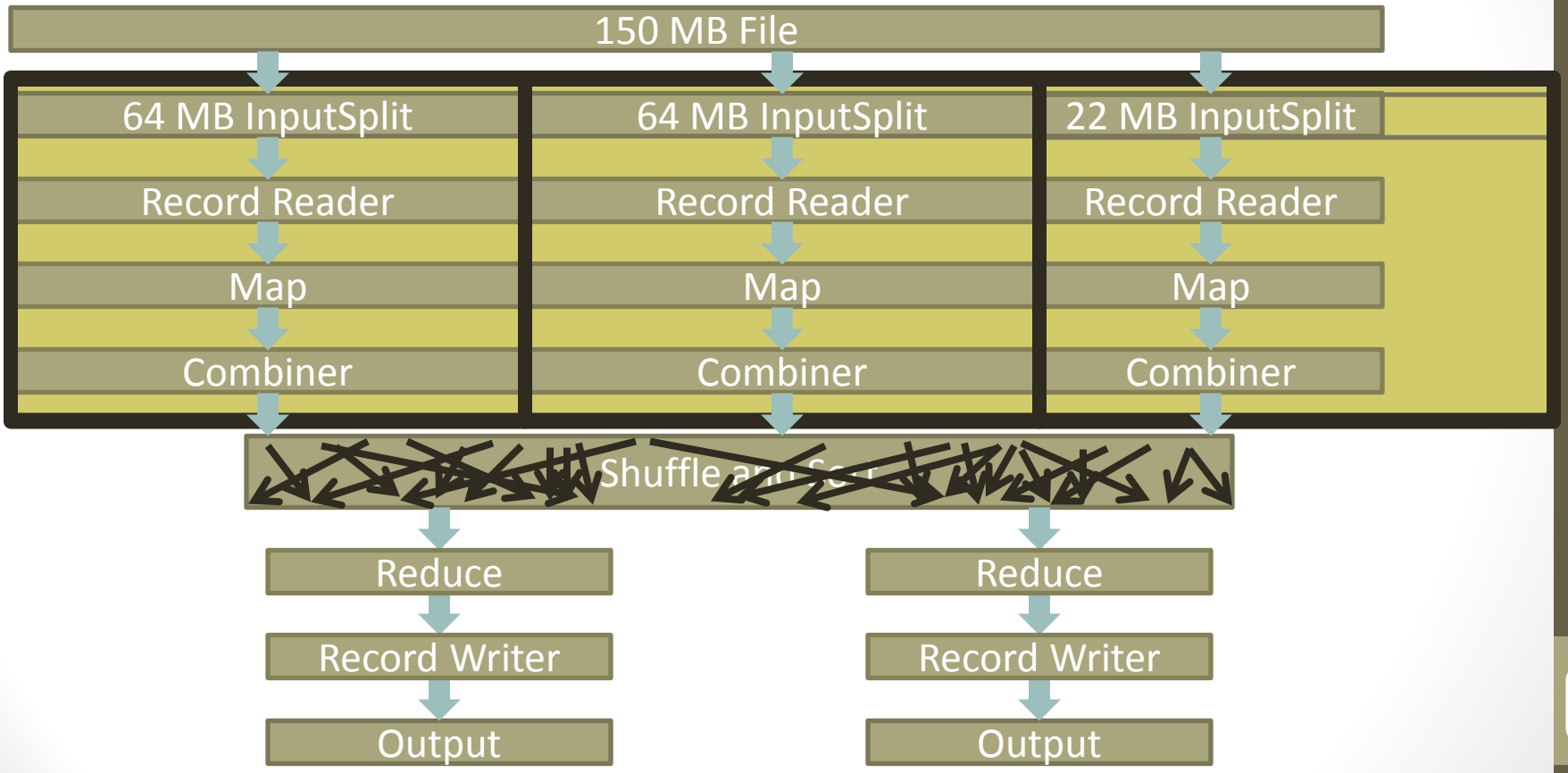
MapReduce (16)



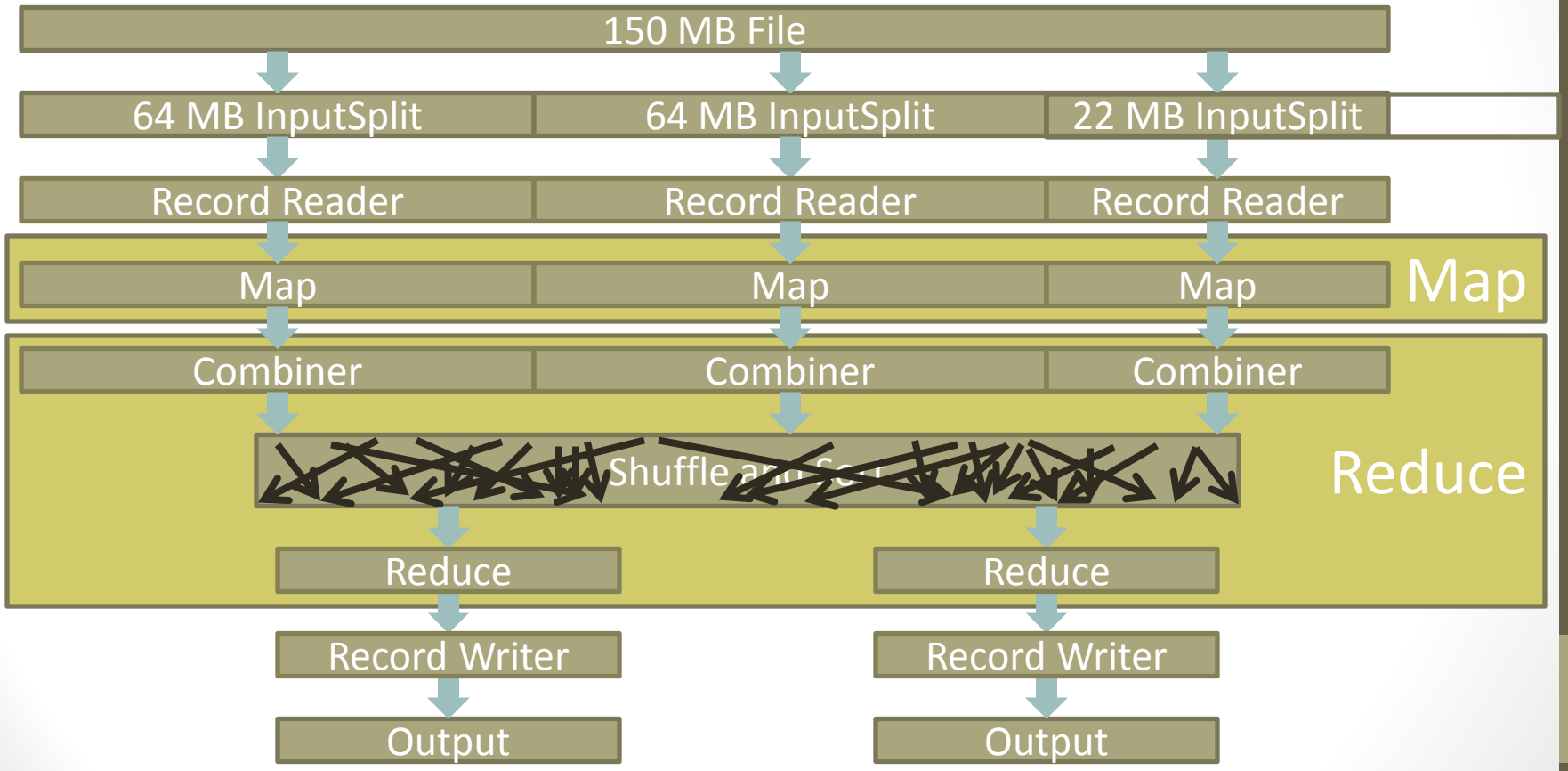
MapReduce (17)



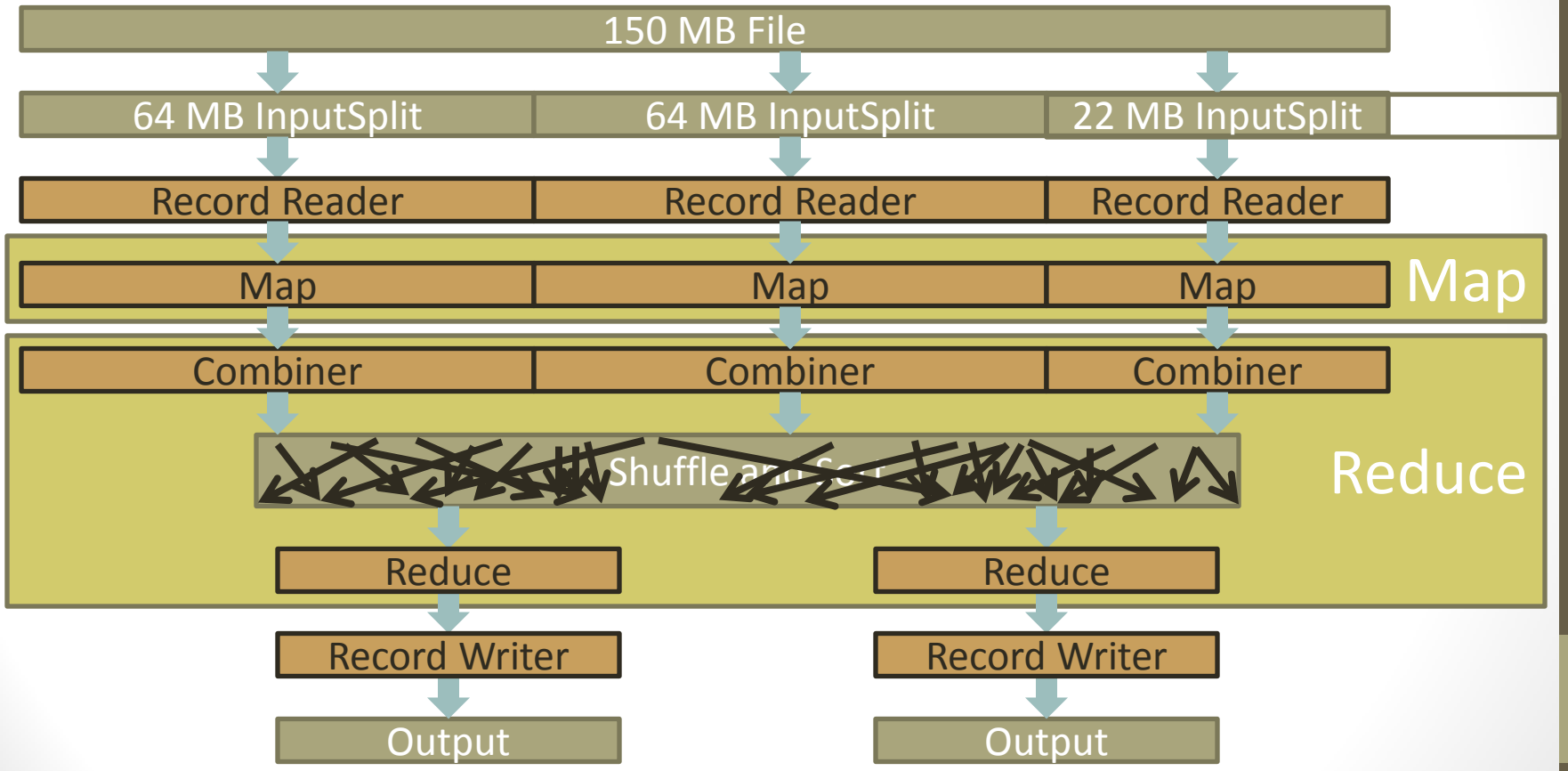
MapReduce (18)



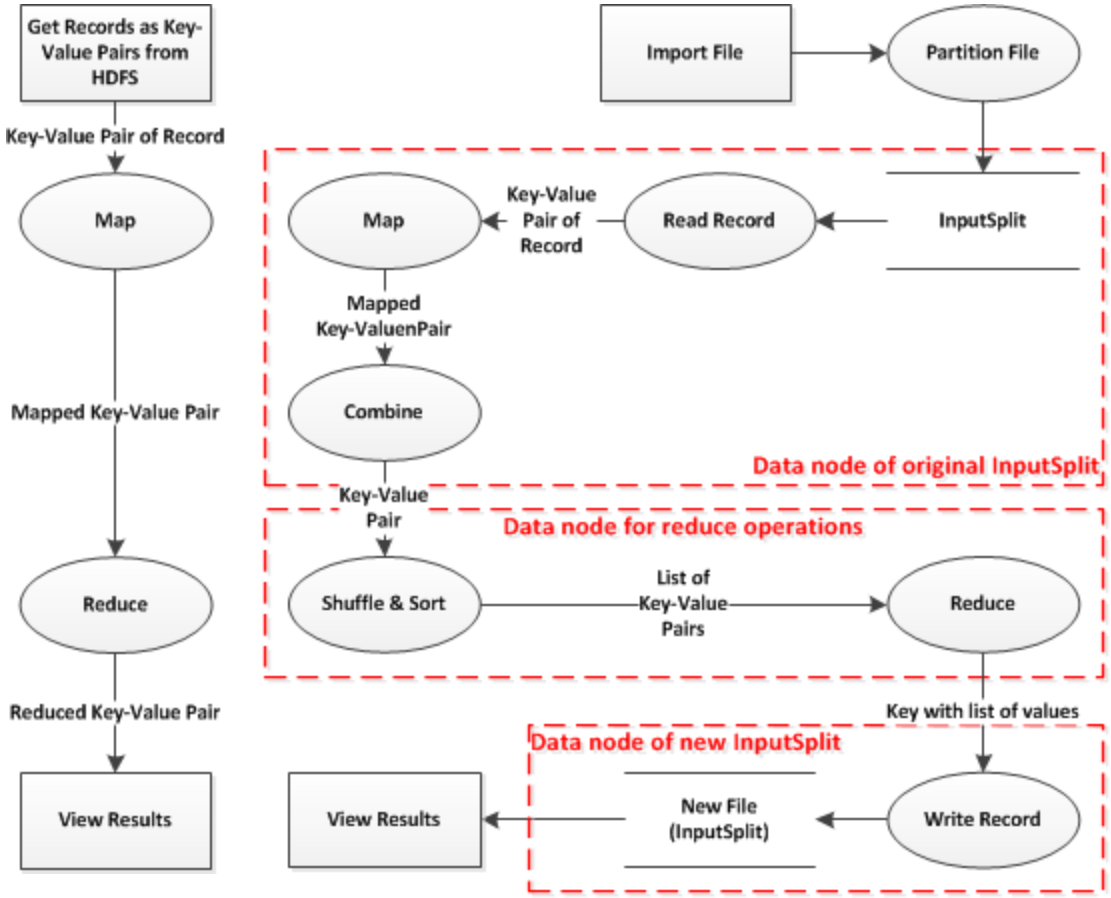
MapReduce(19)



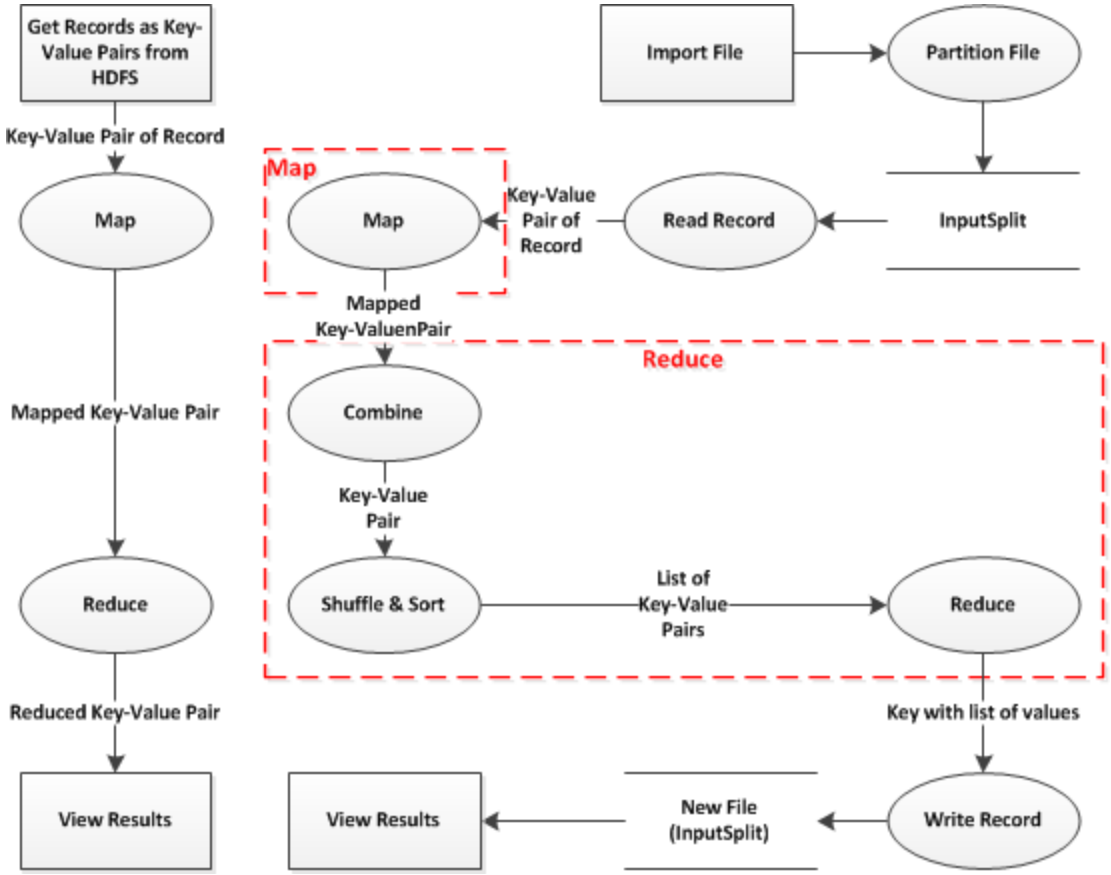
MapReduce (20)



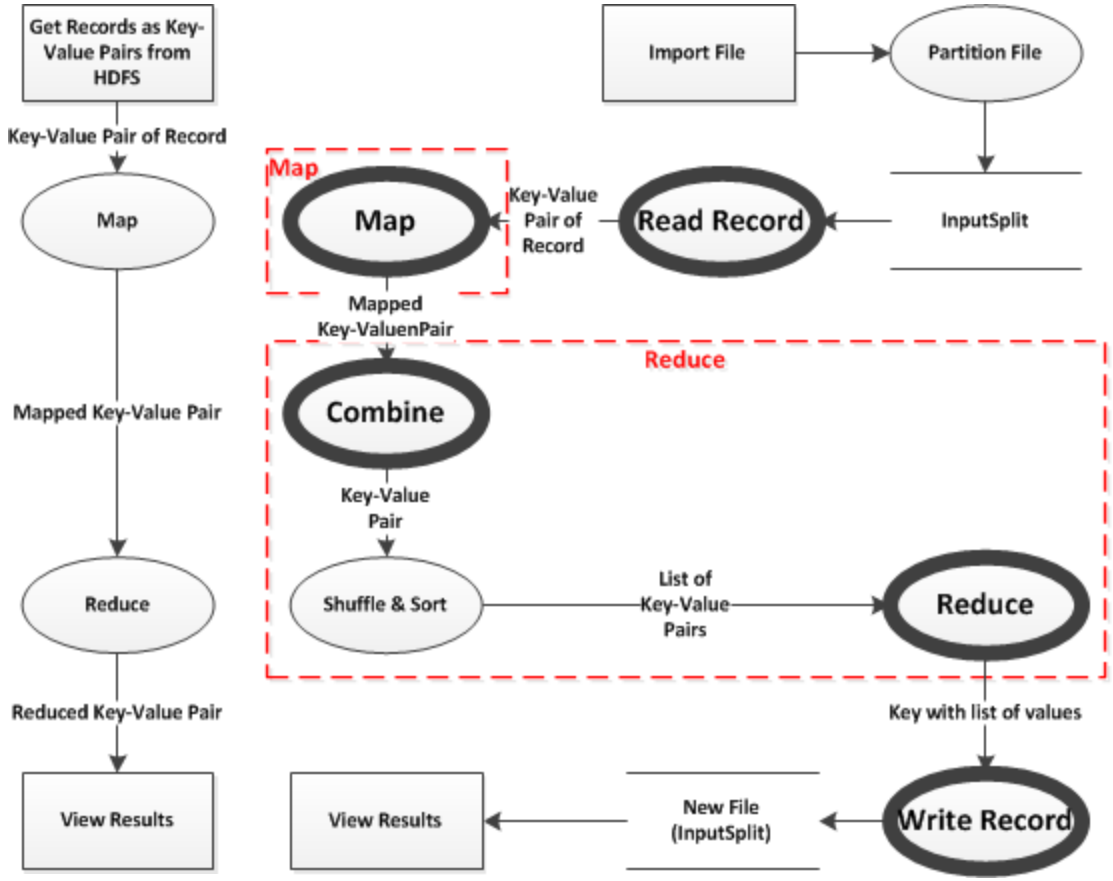
MapReduce (21)



MapReduce (22)



MapReduce (23)

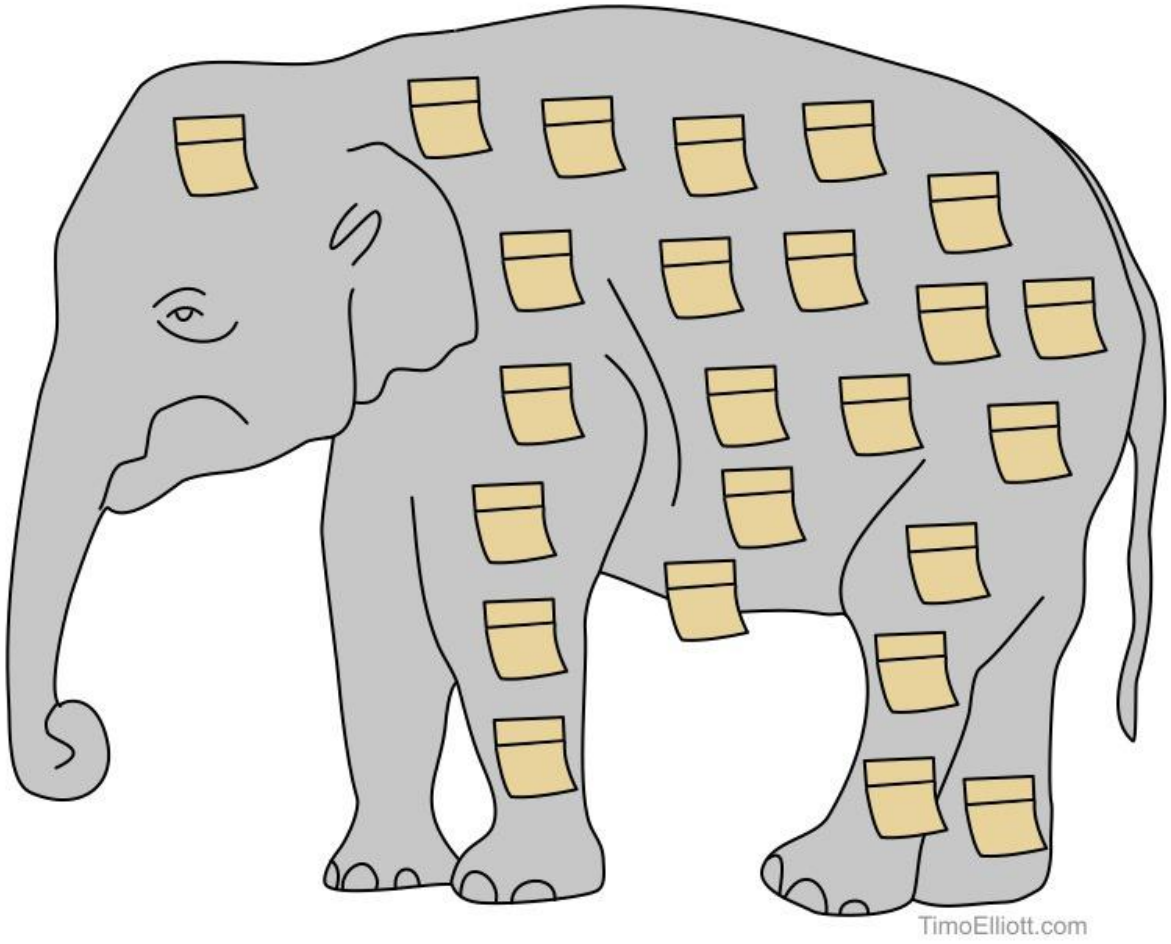


MapReduce (24)

- Record Read
  - Record → (K1,V1)
  - to be or not to be → (0,“to be or not to be”)
- Map
  - (K1,V1) → list(K2,V2)
  - (0,“to be or not to be”) → [(to,1),(be,1),(or,1),(not,1),(to,1),(be,1)]
- Shuffle and Sort
  - list(K2,V2) → (K2,list(V2))
  - [(to,1),(be,1),(or,1),(not,1),(to,1),(be,1)] → (to,[1,1]),(be,[1,1]),(or,[1]),(not,[1])
- Reduce
  - (K2,list(V2)) → list(K3,V3)
  - (to,[1,1]),(be,[1,1]),(or,[1]),(not,[1]) → [(to,2),(be,2),(or,1),(not,1)]
- Record Write
  - (K3,V3) → Records
  - [(to,2),(be,2),(or,1),(not,1)] →

to,2
be,2
or,1
not,1

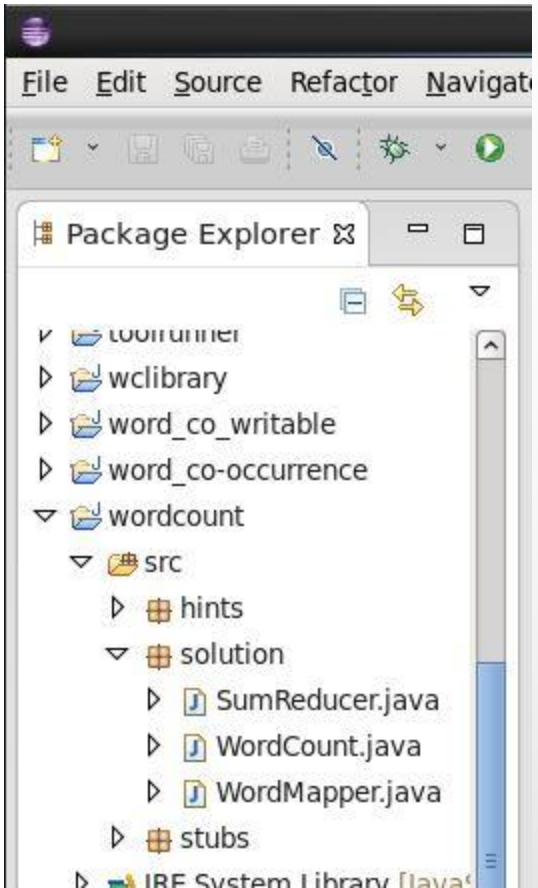
Break



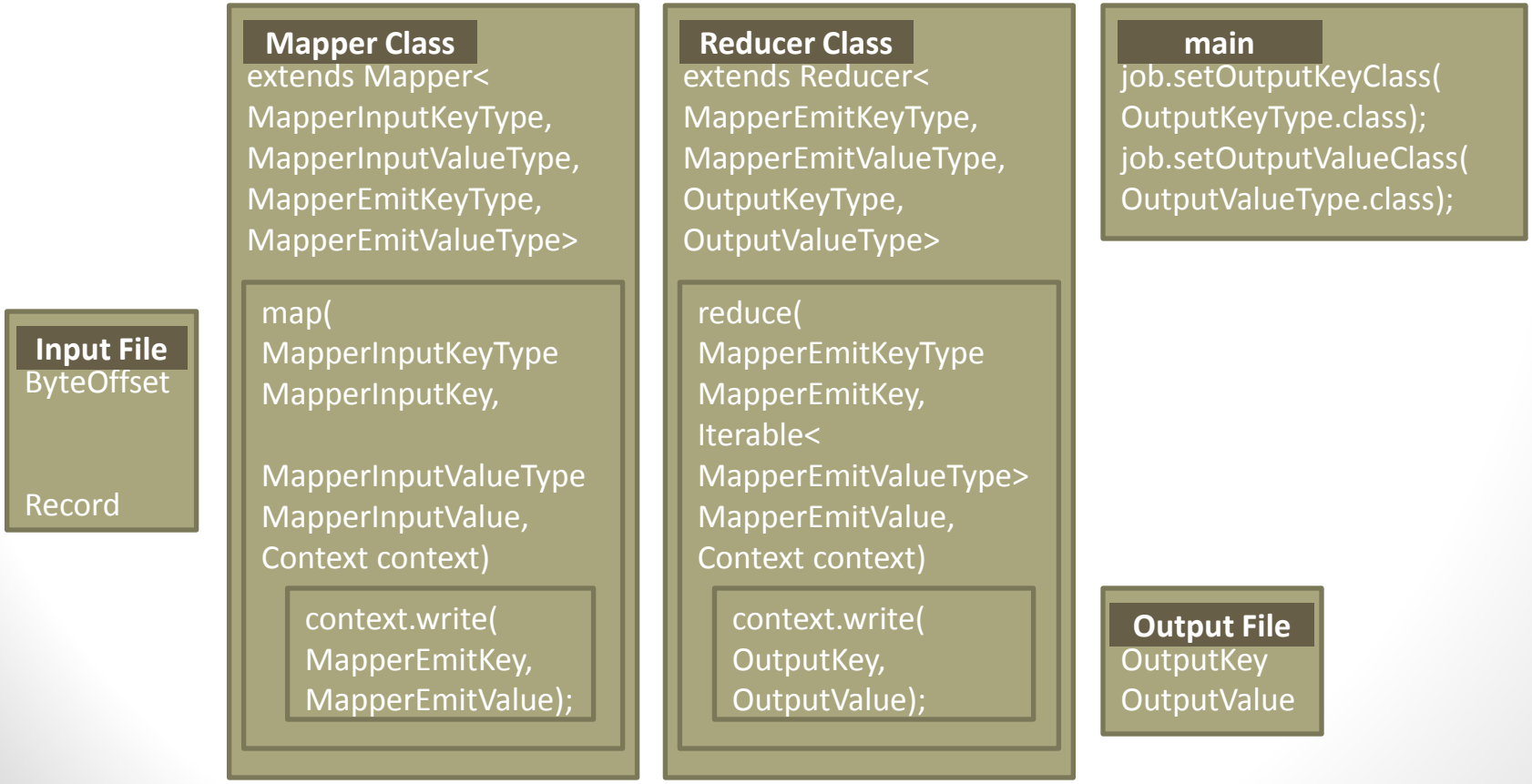
An early Hadoop prototype...

MapReduce Lab (1)

- Open Eclipse by clicking on the Eclipse icon. Then, use the Package Explorer to navigate to wordcount.
- See the java files in wordcount\src\solution\
  - WordCount.java
  - WordMapper.java
  - SumReducer.java
- Study these java files

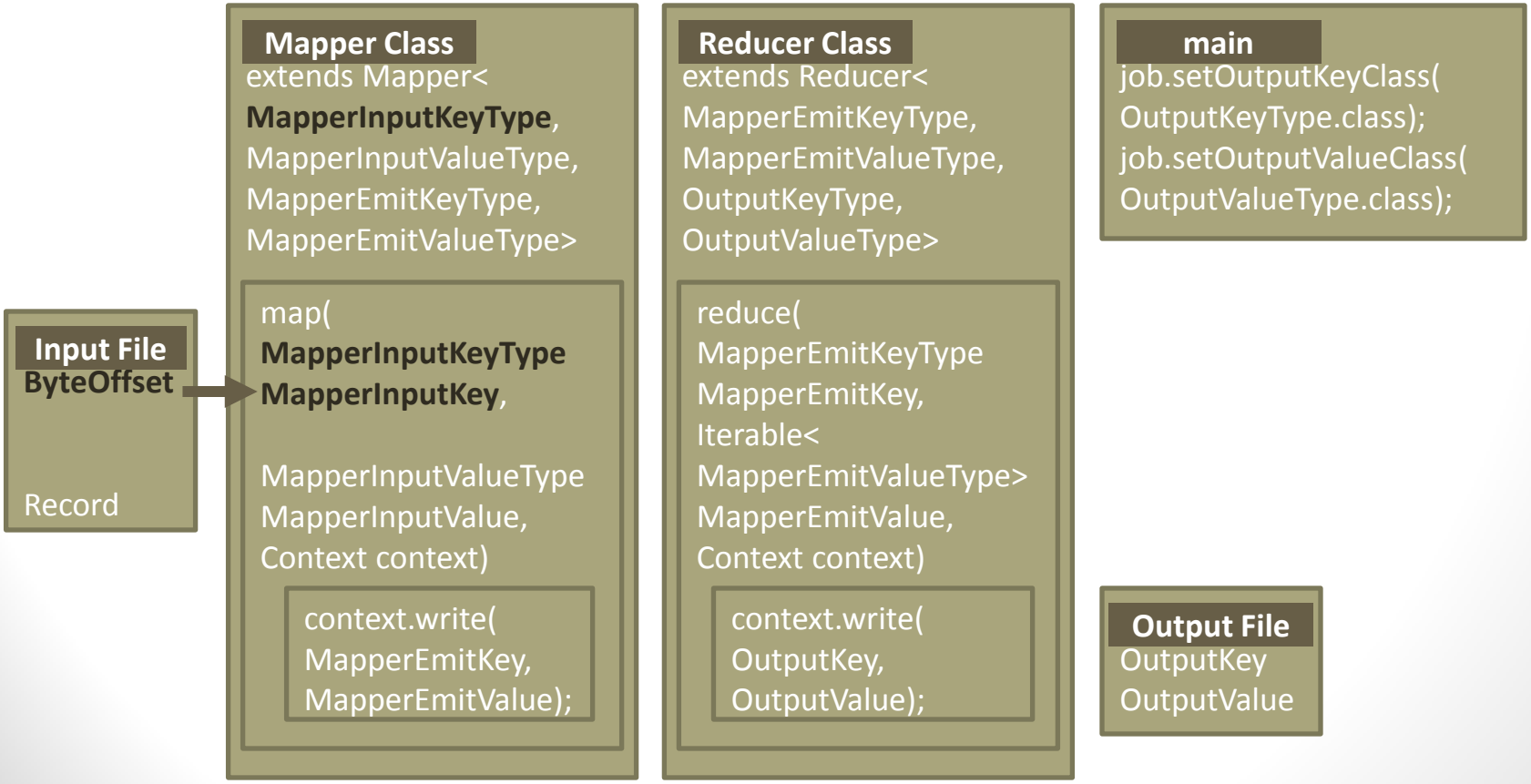


MapReduce Lab (2)



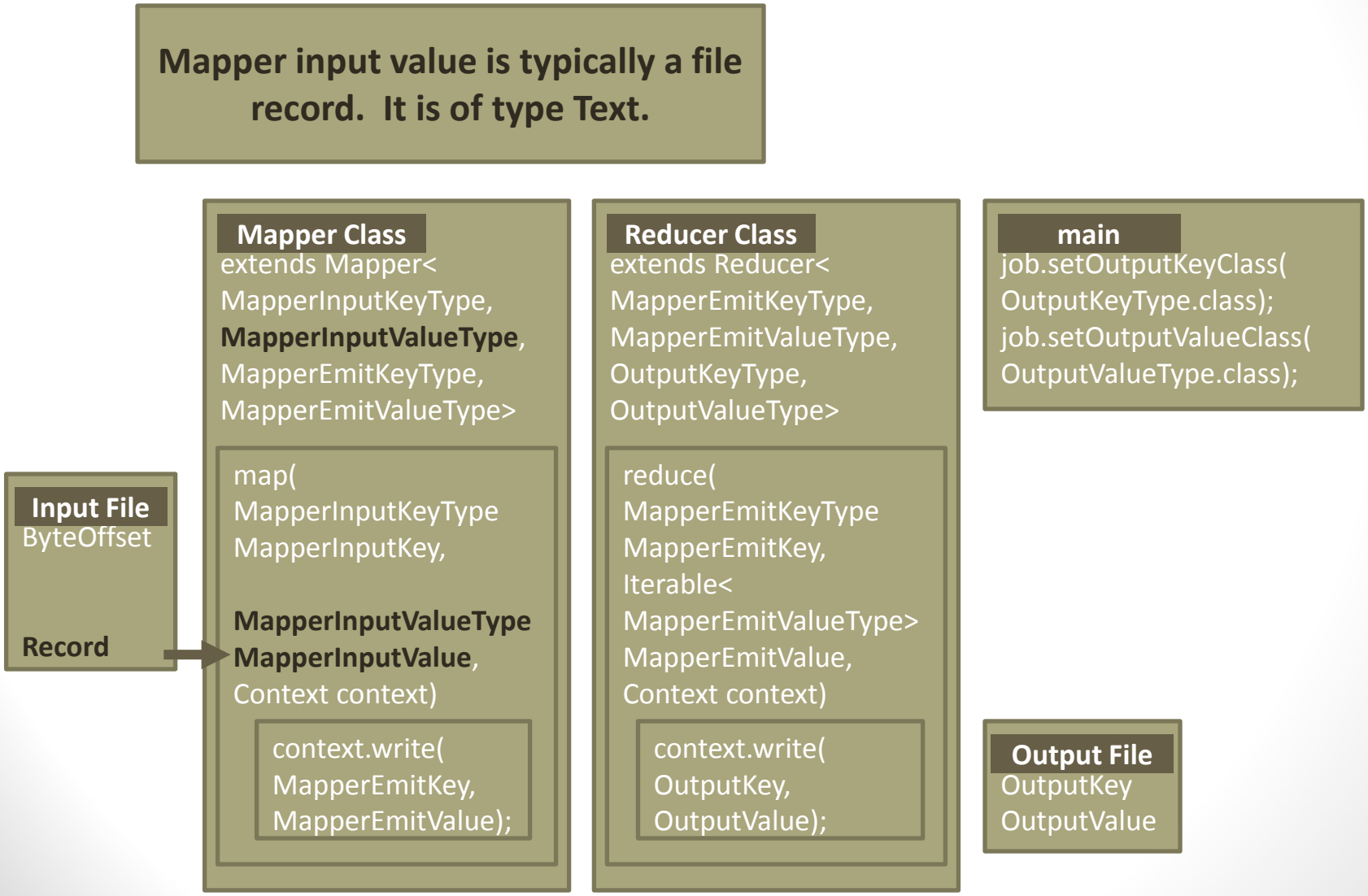
MapReduce Lab (3)

Mapper input key is typically just the file byte offset. Ignore it.



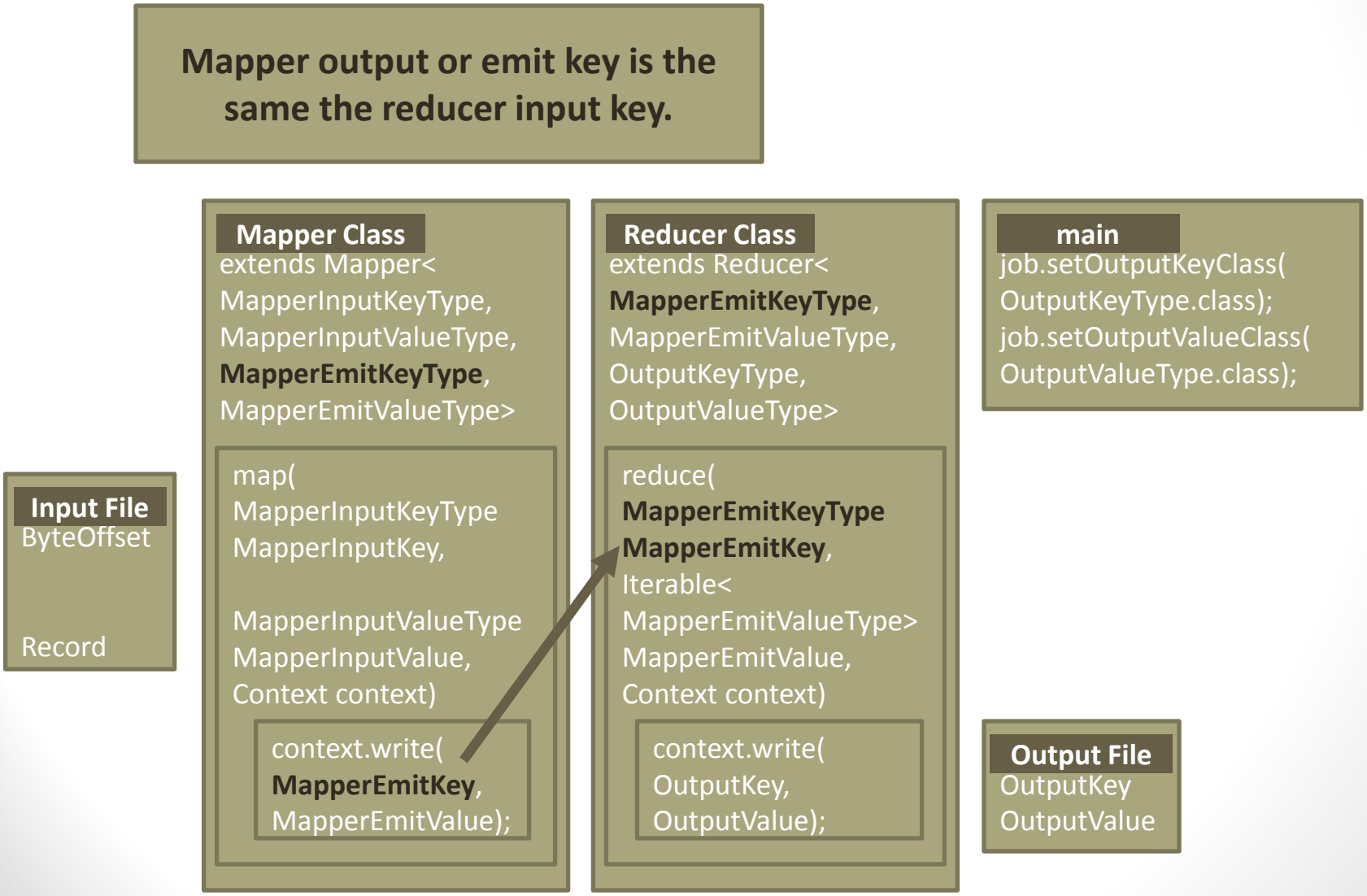
MapReduce Lab (4)

Mapper input value is typically a file record. It is of type Text.



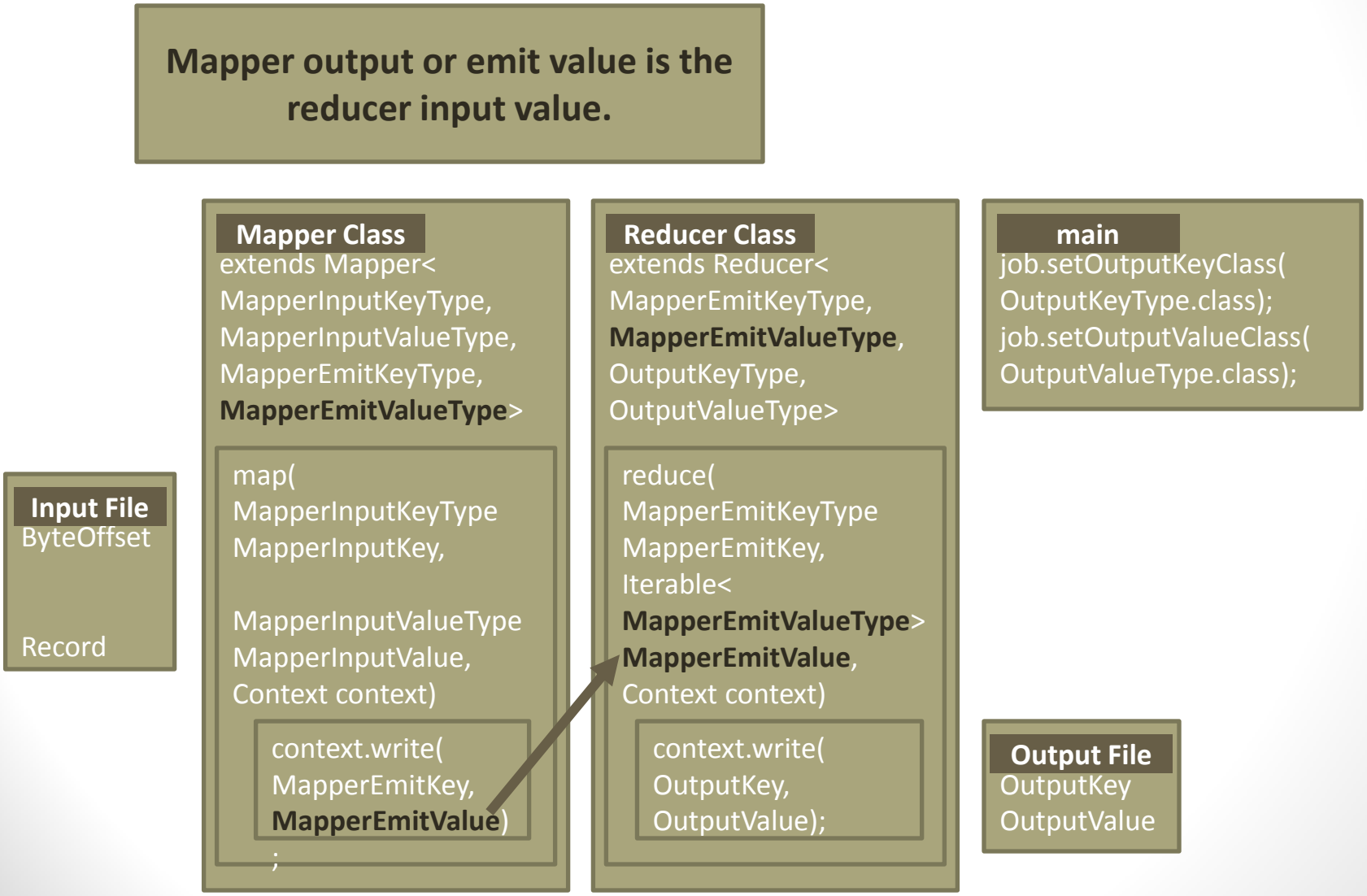
MapReduce Lab (5)

Mapper output or emit key is the same the reducer input key.



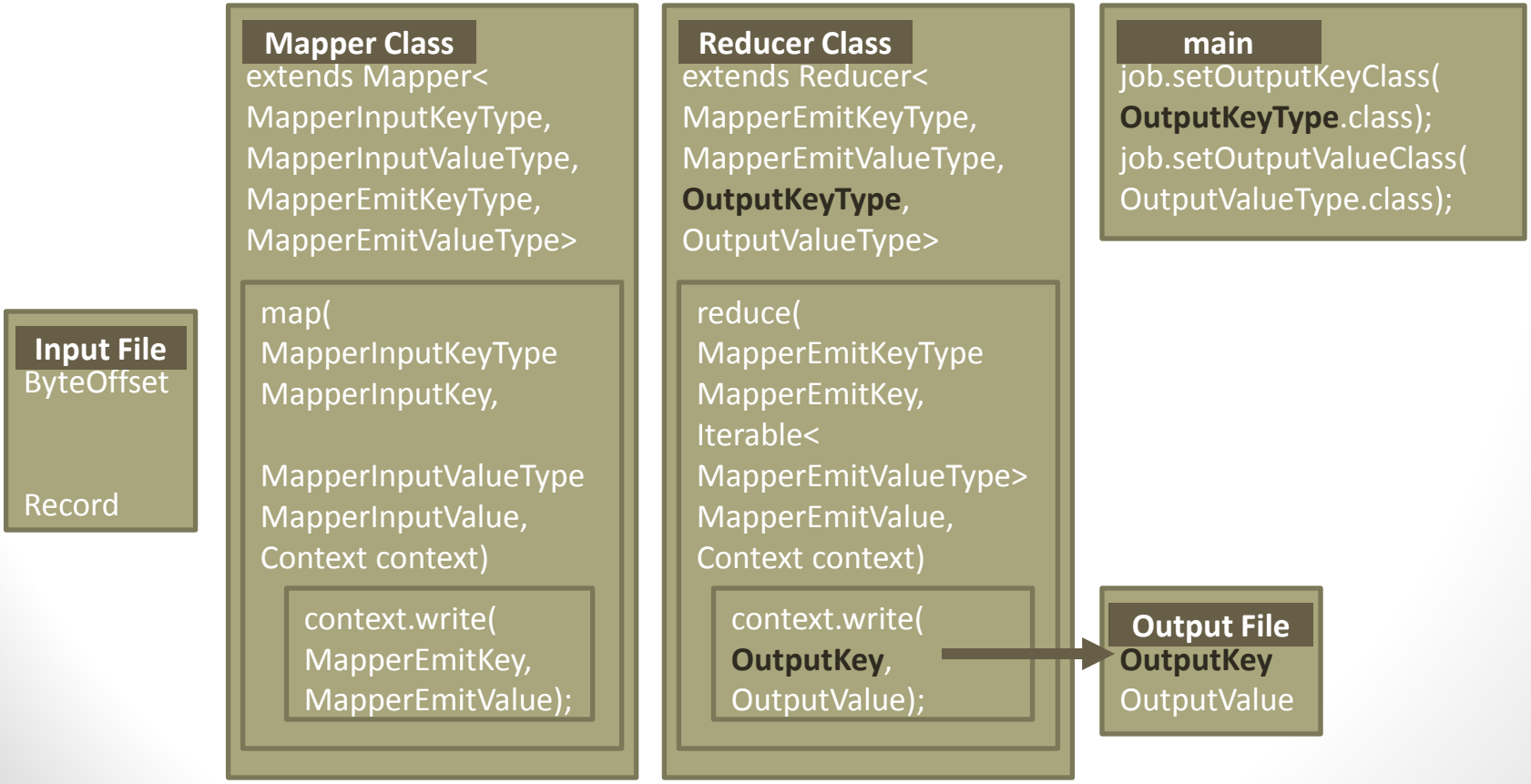
MapReduce Lab (6)

Mapper output or emit value is the reducer input value.



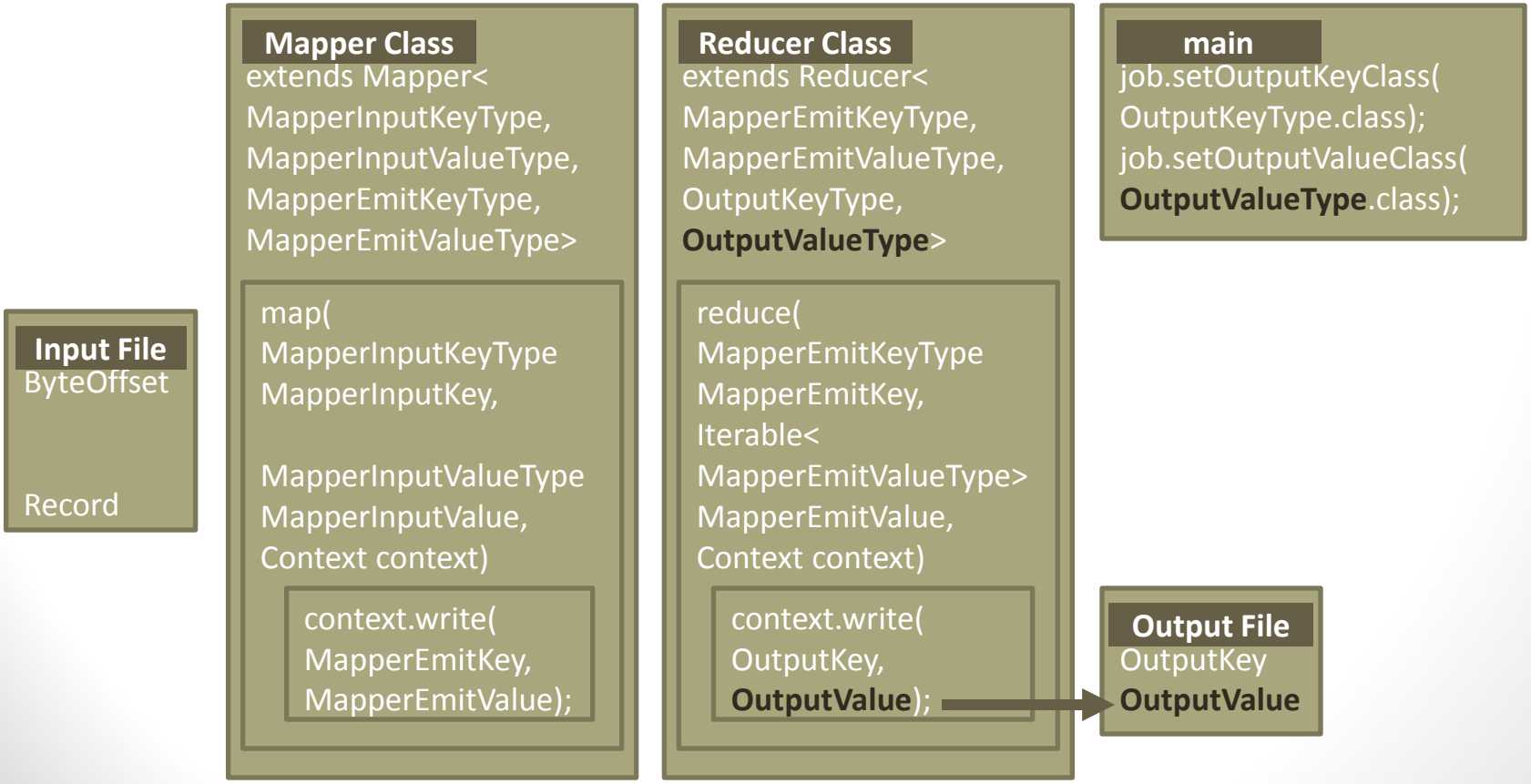
MapReduce Lab (7)

Reducer output or emit key is the output key from main and the key of the Output File

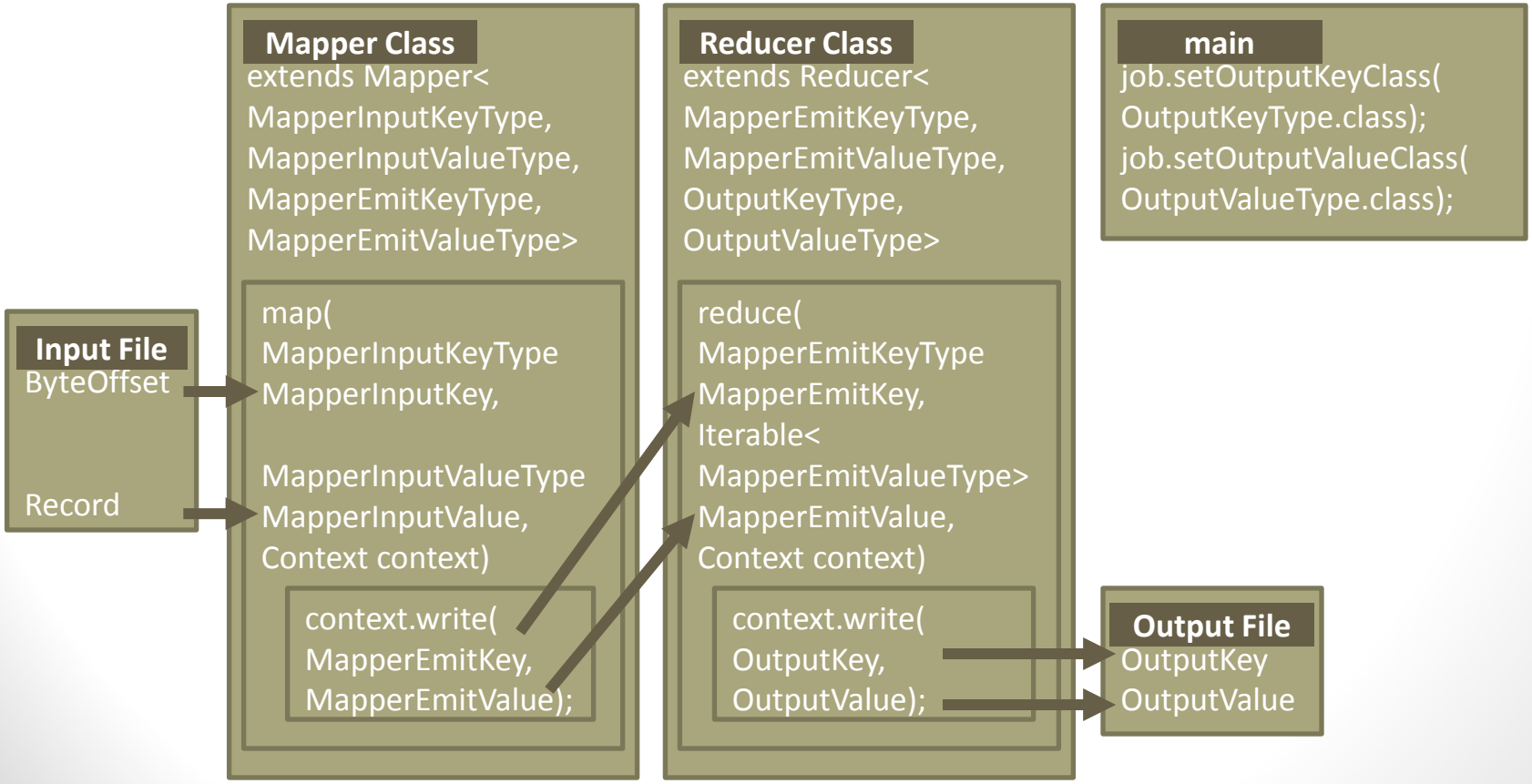


MapReduce Lab (8)

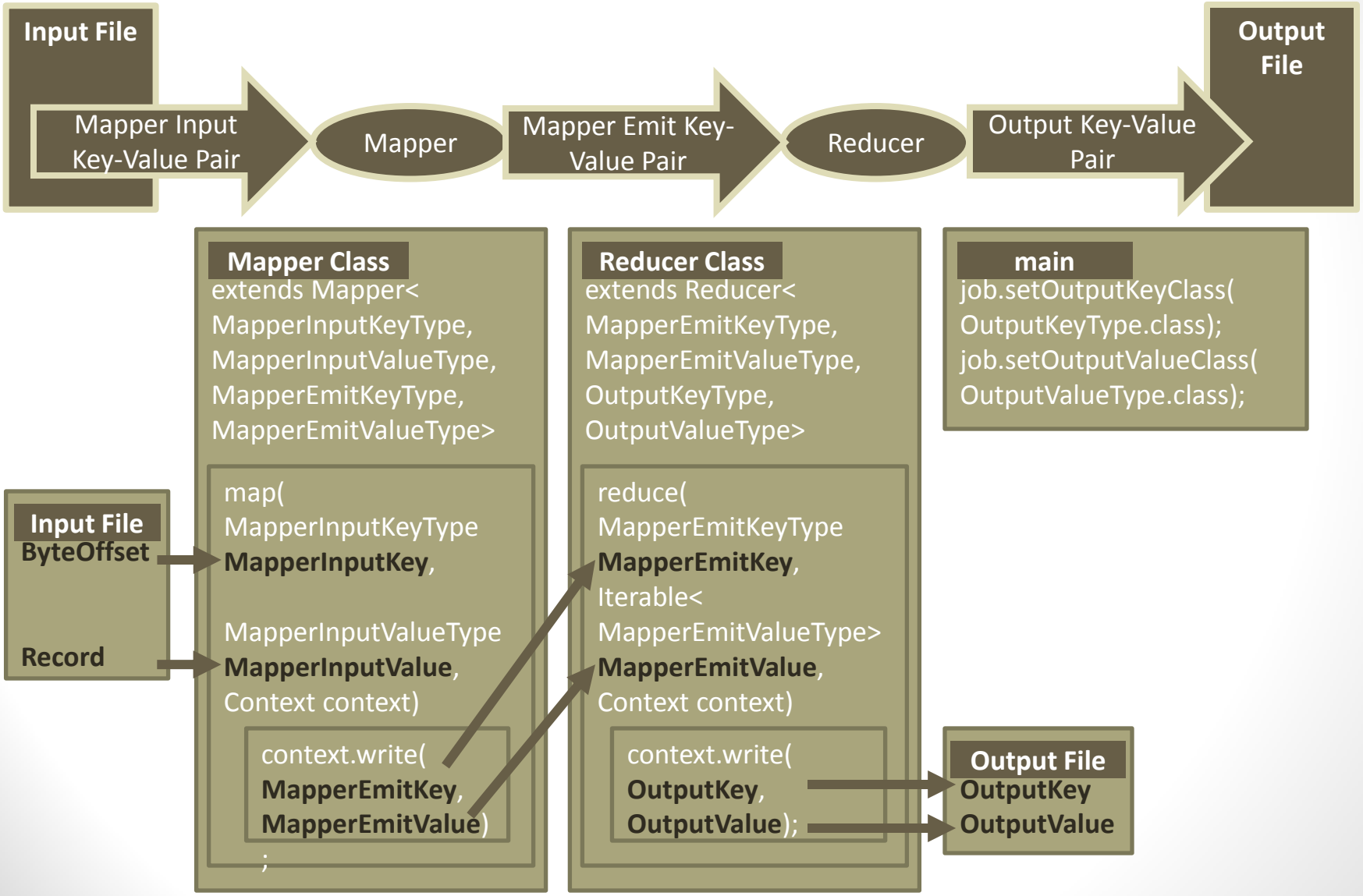
Reducer output or emit value is the output value from main and the value of the Output File



MapReduce Lab (9)



MapReduce Lab (10)



MapReduce Lab (11)

Open a terminal

In the terminal change the current directory:

- `$ cd ~/workspace/wordcount/src`

List the directories and files in the current directory:

- `$ ls`

List the files in the solution directory:

- `$ ls solution`

Show the classpath to get the Hadoop Jar and other required libraries:

- `$ hadoop classpath`

Compile the java files to class files:

- `$ javac -classpath `hadoop classpath` solution/*.java`

List the files in the solution directory (note the new class files):

- `$ ls solution`

Note: the apostrophe in this statement is not the straight or slanted quote (' or ‘) but rather this character (`).

MapReduce Lab (12)

In the terminal:

Create a JAR file containing the class files:

- `$ jar cvf wordcount.jar solution/*.class`

List the files in the current directory (note the new JAR file):

- `$ ls`

In Hadoop, run the wordcount program on all of Shakespeare's works:

- `$ hadoop jar wordcount.jar solution.WordCount shakespeare wordcounts`

In HDFS list the files that contain the results:

- `$ hadoop fs -ls wordcounts`

In HDFS list the results:

- `$ hadoop fs -cat wordcounts/part-r-00000 | less`

In Hadoop, run the wordcount program on Shakespeare's poems:

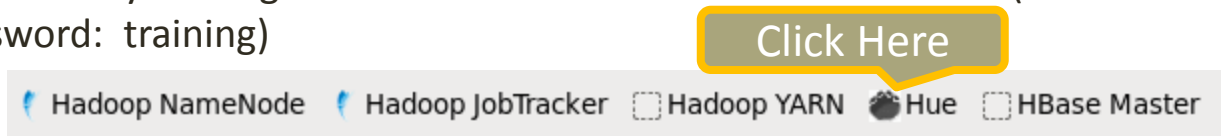
- `$ hadoop jar wc.jar solution.WordCount shakespeare/poems pwords`

hadoop fs -rm -r wordcounts pwords

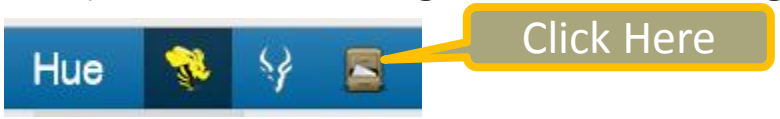
MapReduce Lab (13)

- Use Hue's HDFS file browser to browse and read the results
  1. Open Firefox

  2. Start Hue by clicking on the Hue link in the Bookmarks toolbar (Username: training, Password: training)



  3. Start File Browser (Username: training, Password: training)



  4. Find wordcounts/part-r-00000

  5. View Contents of part-r-00000

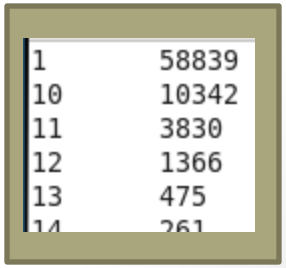
Select Shakespeare > histories > Viewing Bytes 1163653 – 1168000; Read: **King Richard II**

MapReduce Lab (14)

- The next two slides describe the assignment for this week.

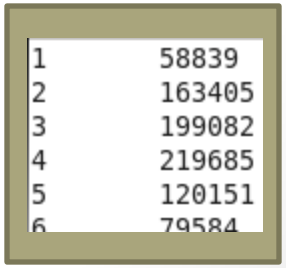
MapReduce Lab (15)

- Lab: Word Length Count 1
- This is part of today's homework assignment.
- Create a MapReduce program that lists word lengths and how often words of that length occur. Modify package wordcount.solution:
  - Calculate word length (word.length()) and cast to a Text.
  - Change map method to use the Text word length as the key.
  - Compile and run the program on shakespeare's works:
    - `hadoop jar wordcount.jar solution.WordCount shakespeare lengthcounts1`
  - Copy the modified java files (E.g. MyFile.java -> MyFile_1.java)
  - View the results using the file browser
    - `hadoop fs -cat lengthcounts1/part-r-00000 | less`
  - Take a screen shot of the results (approx. 18 rows). Similar to this:



MapReduce Lab (16)

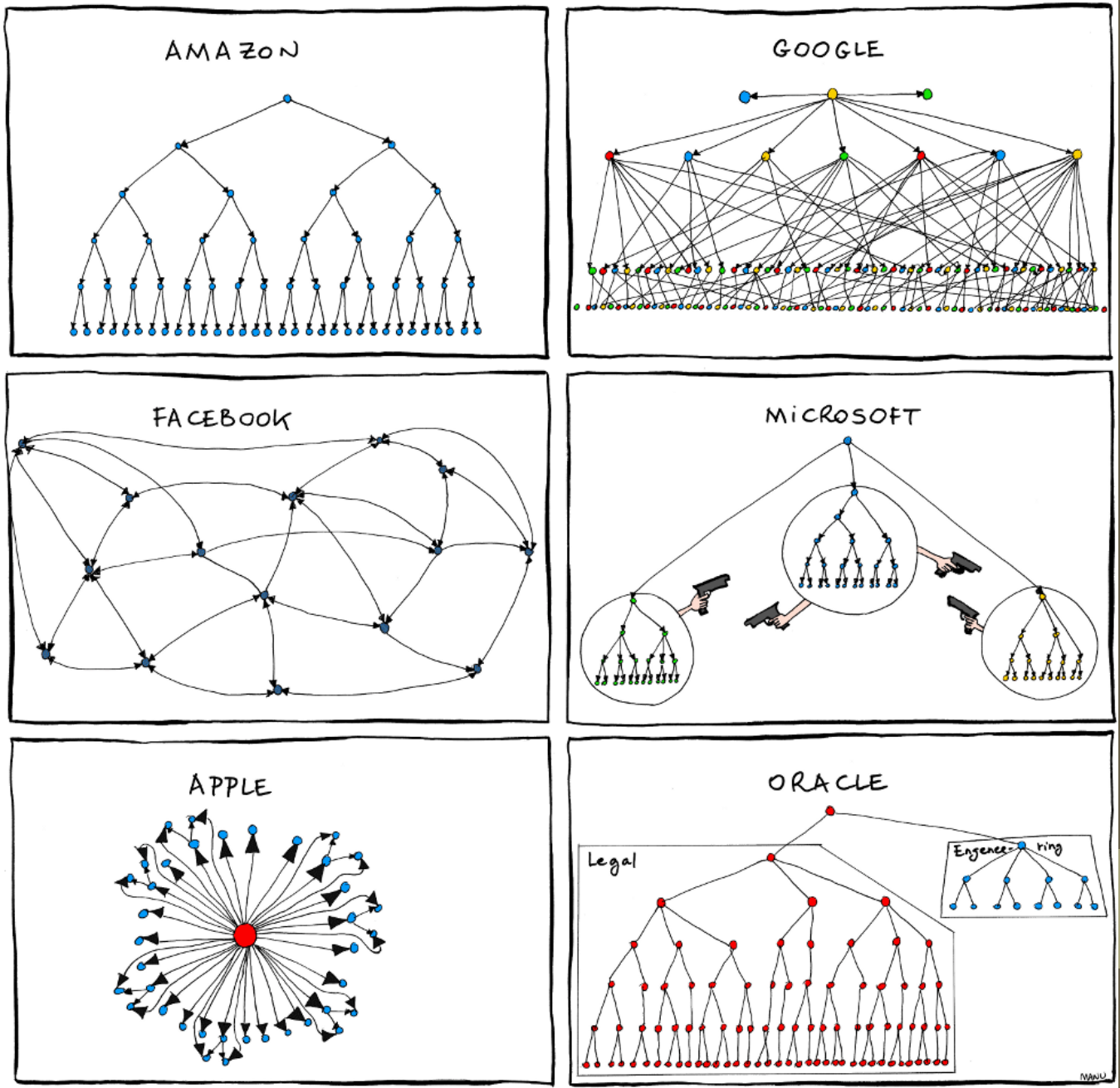
- Lab: Word Length Count 2
- This is part of today's homework assignment.
- Create a MapReduce program that lists word lengths and how often words of that length occur. Modify package wordcount.solution:
  - Calculate word length (word.length()) and cast to IntWritable
  - Change map method to use the word length as the key
  - Compile and run the program on shakespeare's works:
    - `hadoop jar wordcount.jar solution.WordCount shakespeare lengthcounts2`
  - Copy the modified java files (E.g. MyFile.java -> MyFile_2.java)
  - View the results using the file browser
    - `hadoop fs -cat lengthcounts2/part-r-00000 | less`
  - Take a screen shot of the results (approx. 18 rows). Similar to this:




C'mon, Hadoop — it's bedtime!

Hadoop-2 Labs

Break

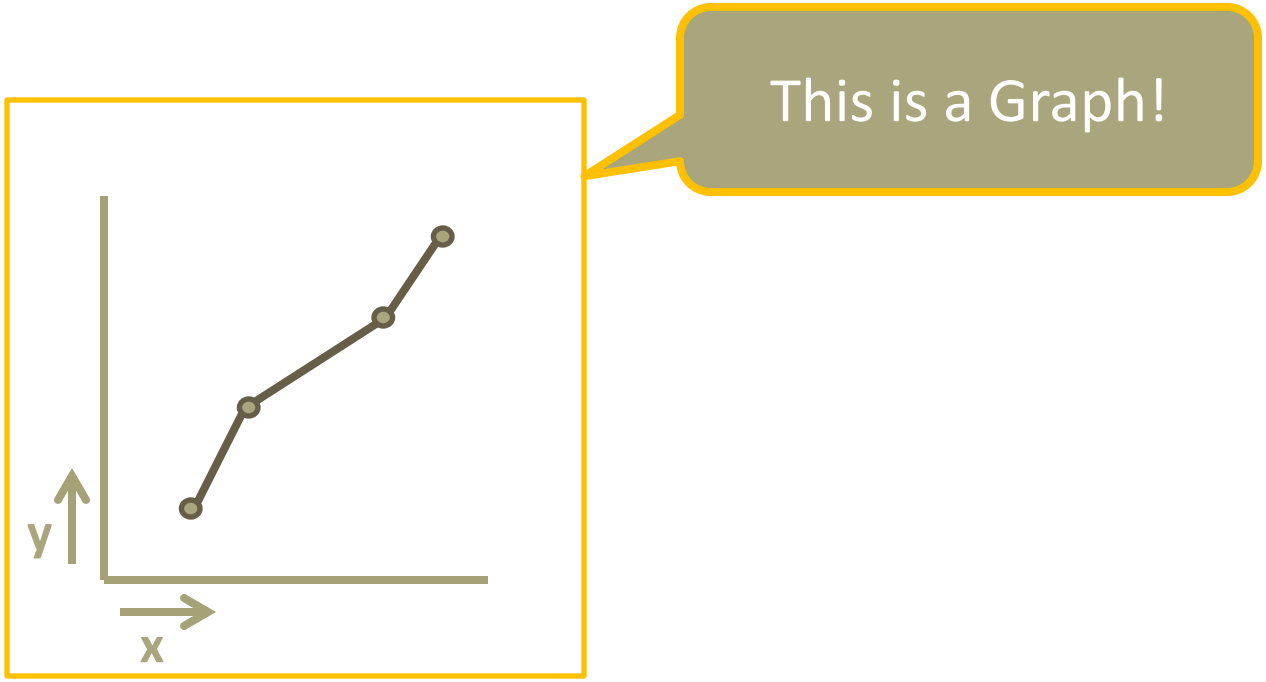

AMAZON
GOOGLE
FACEBOOK
MICROSOFT
APPLE
ORACLE
Legal
Engineering
MANU

Introduction to Graph Data

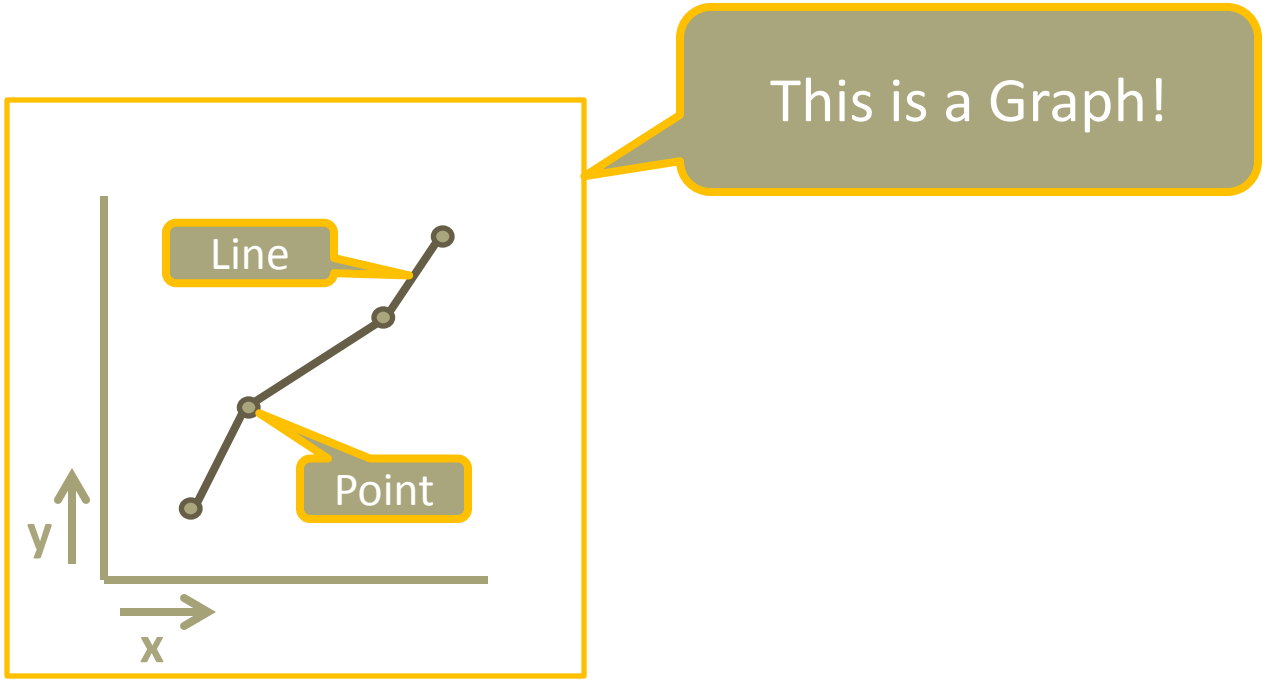
Graph Data: What is it? (0)

What is a Graph?

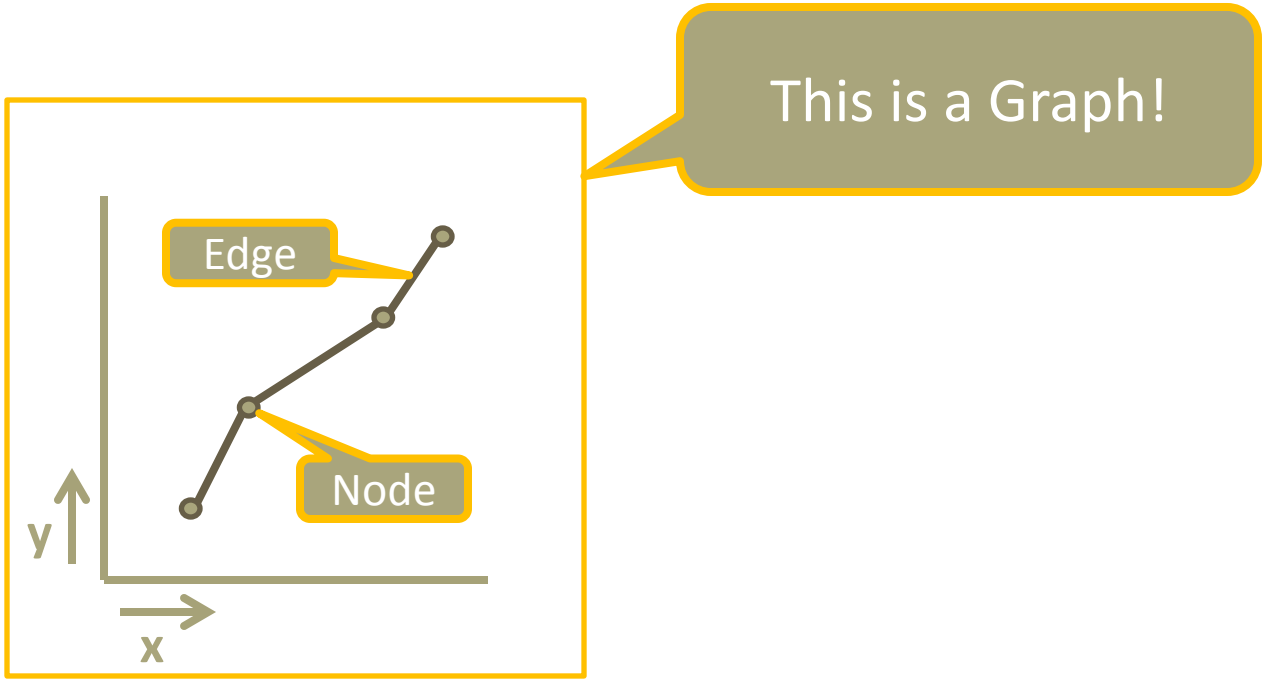
Graph Data: What is it? (1)



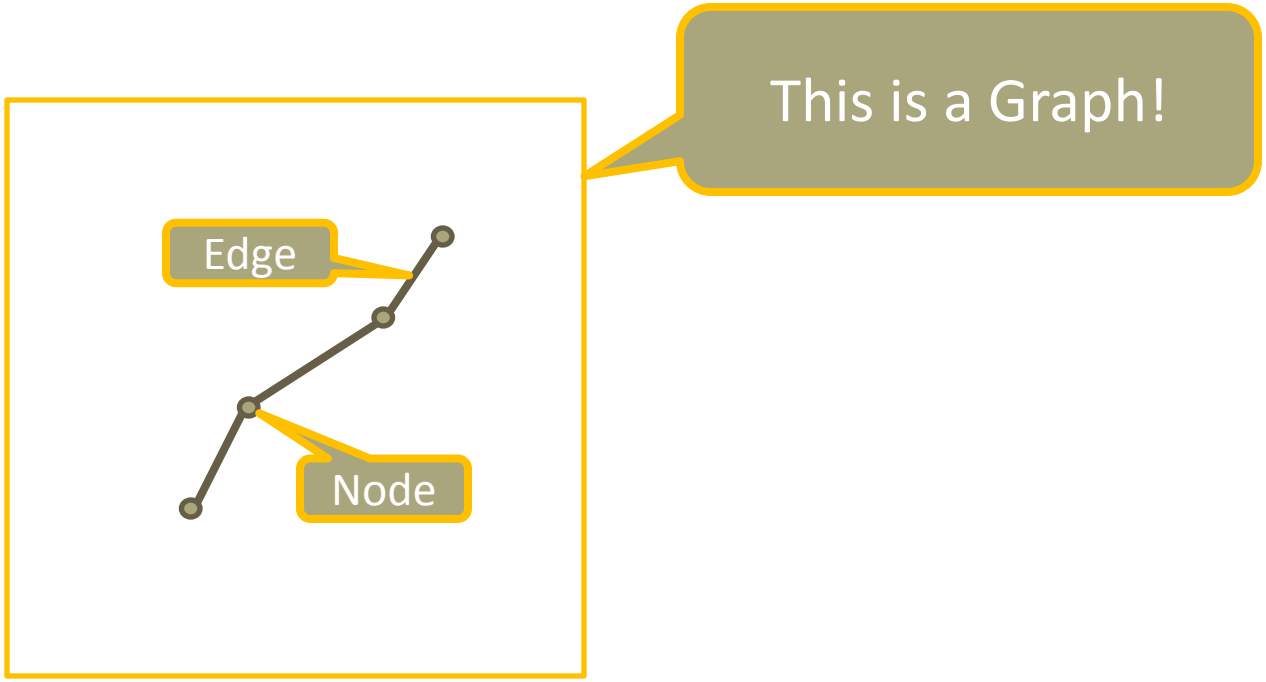
Graph Data: What is it? (2)



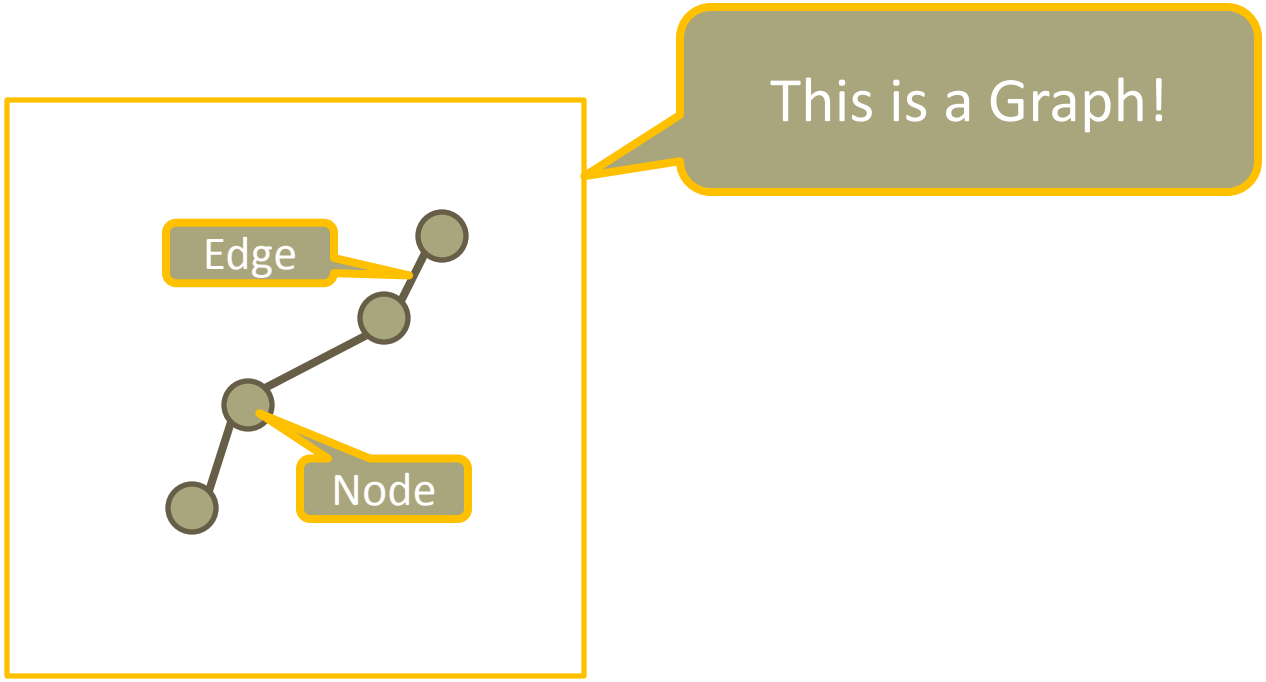
Graph Data: What is it? (3)



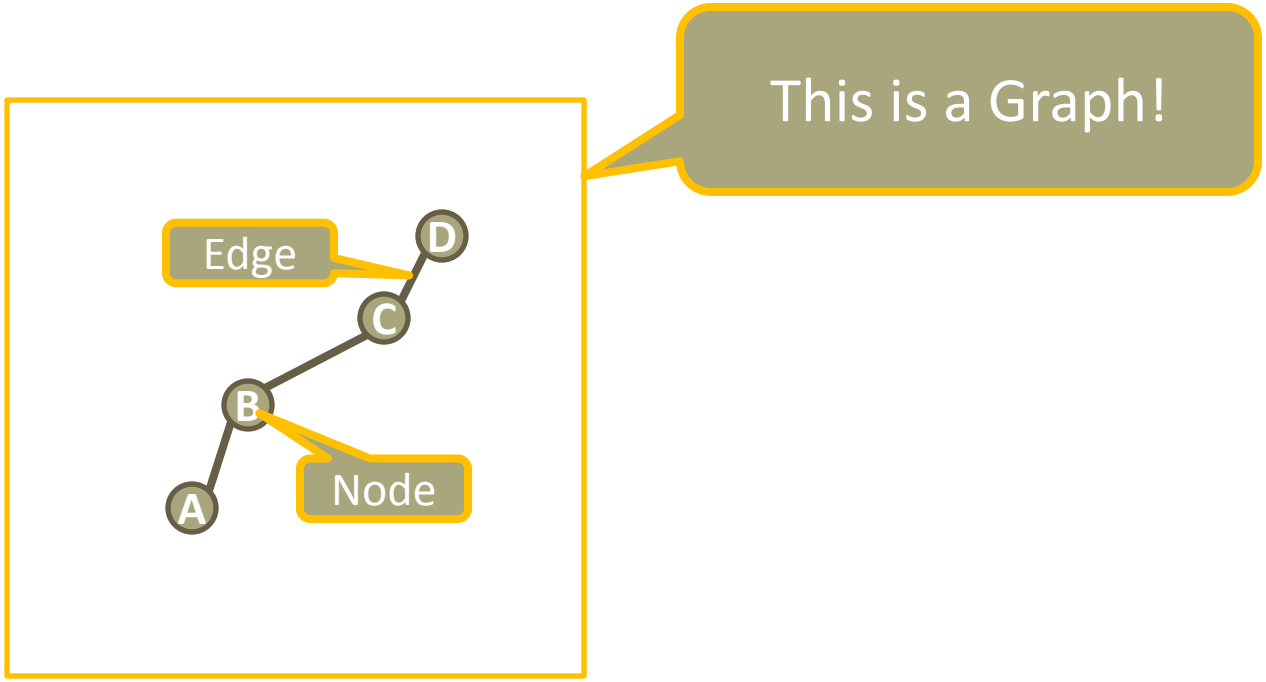
Graph Data: What is it? (4)



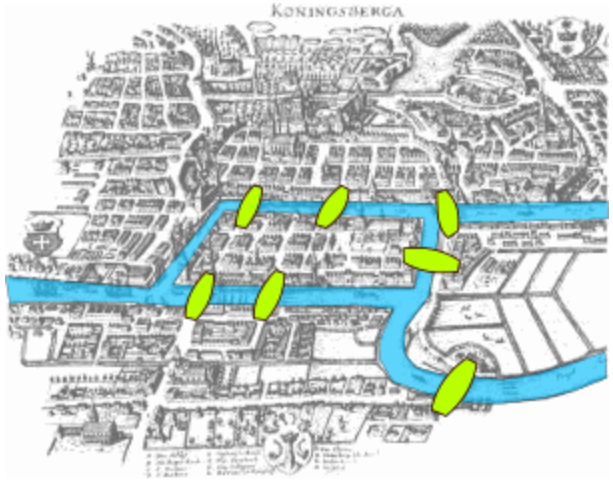
Graph Data: What is it? (5)



Graph Data: What is it? (6)



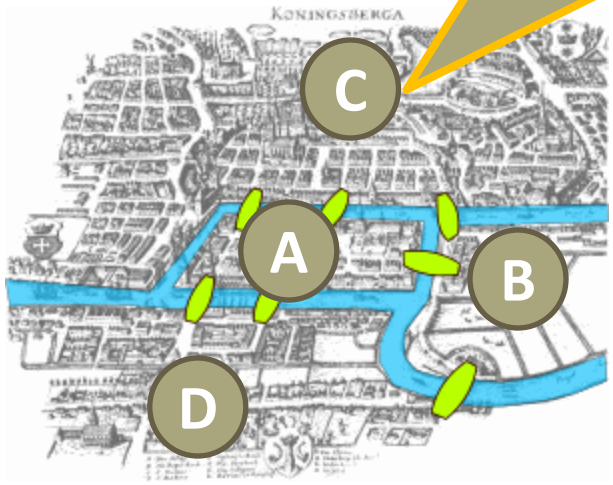
Graph Data: Euler's Seven Bridges of Königsberg (0)



"find a walk through the city that would cross each bridge once and only once"
http://en.wikipedia.org/wiki/Seven_Bridges_of_K%C3%B6nigsberg

Graph Data: Euler's Seven Bridges of Königsberg (1)

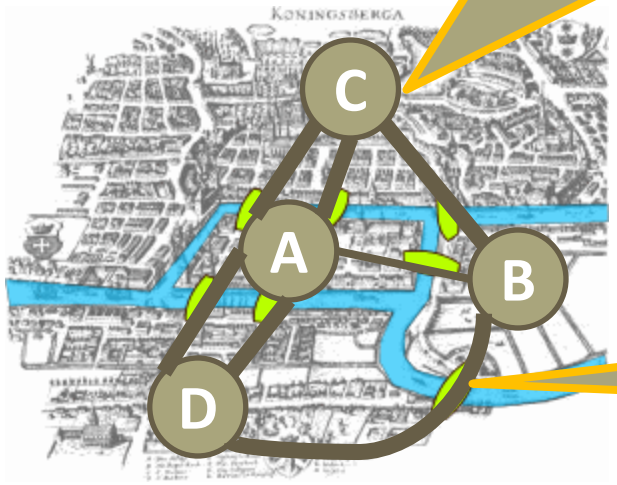
Each separated piece of land is a node



"find a walk through the city that would cross each bridge once and only once"
http://en.wikipedia.org/wiki/Seven_Bridges_of_K%C3%B6nigsberg

Graph Data: Euler's Seven Bridges of Königsberg (2)

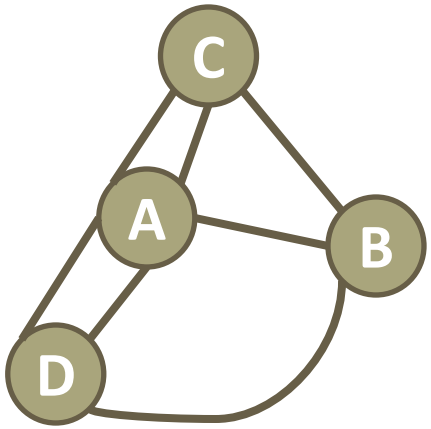
Each separated piece of land is a node



Each bridge is an edge

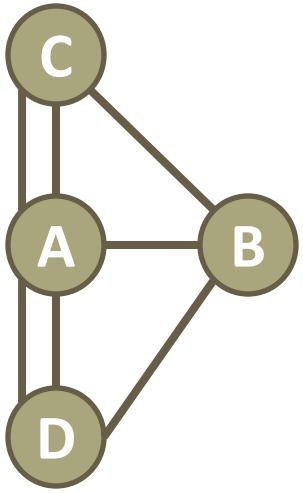
"find a walk through the city that would cross each bridge once and only once"
http://en.wikipedia.org/wiki/Seven_Bridges_of_K%C3%B6nigsberg

Graph Data: Euler's Seven Bridges of Königsberg (3)



"find a walk through the city that would cross each bridge once and only once"
http://en.wikipedia.org/wiki/Seven_Bridges_of_K%C3%B6nigsberg

Graph Data: Euler's Seven Bridges of Königsberg (4)

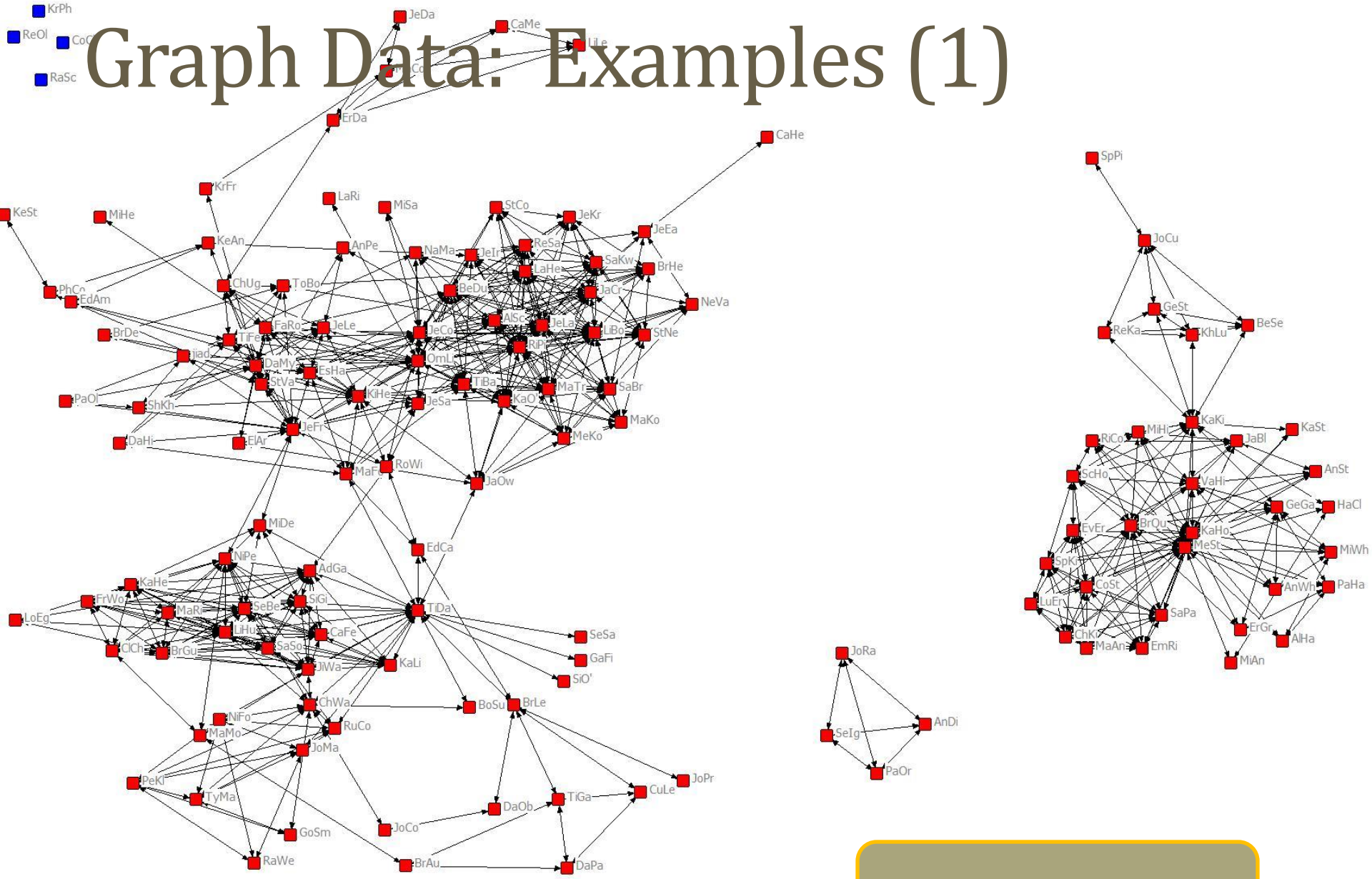


"find a walk through the city that would cross each bridge once and only once"
http://en.wikipedia.org/wiki/Seven_Bridges_of_K%C3%B6nigsberg

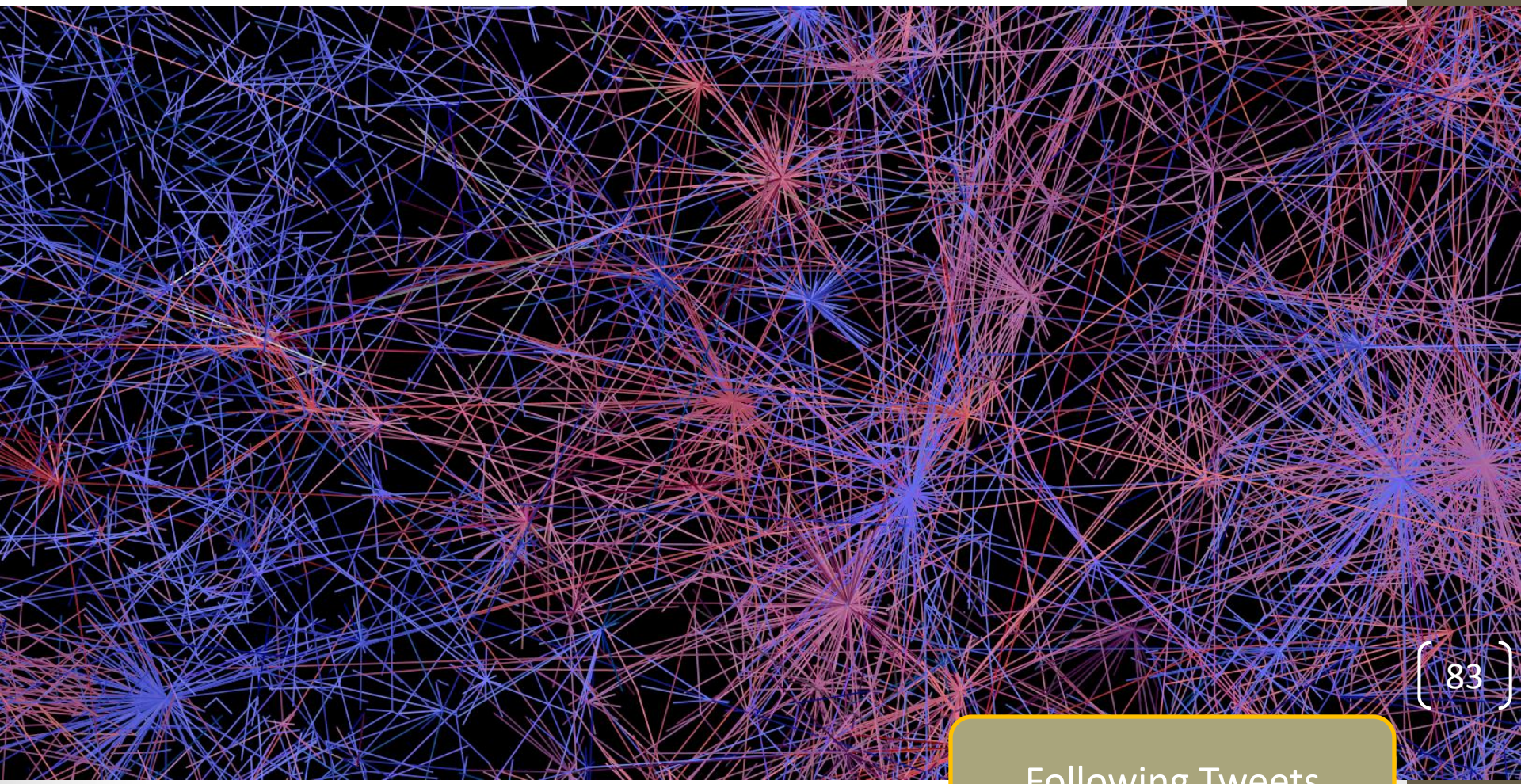
Graph Data: Examples (0)

- Graphs represent many things like Associations and Networks.
- Graphs represent the world wide web
- Graphs represent Facebook networks
- Graphs represent metabolic pathways

Graph Data: Examples (1)

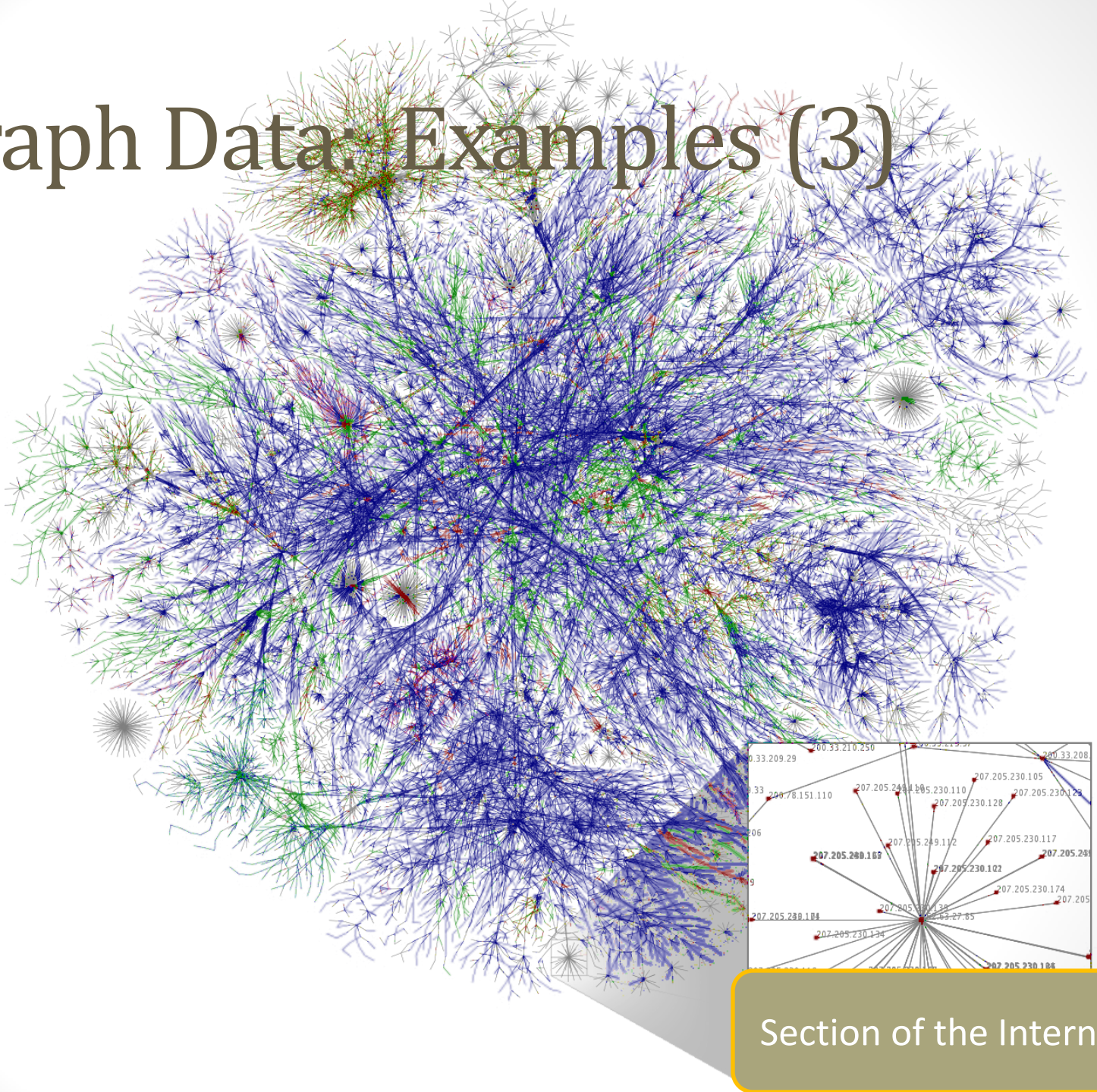


Graph Data: Examples (2)



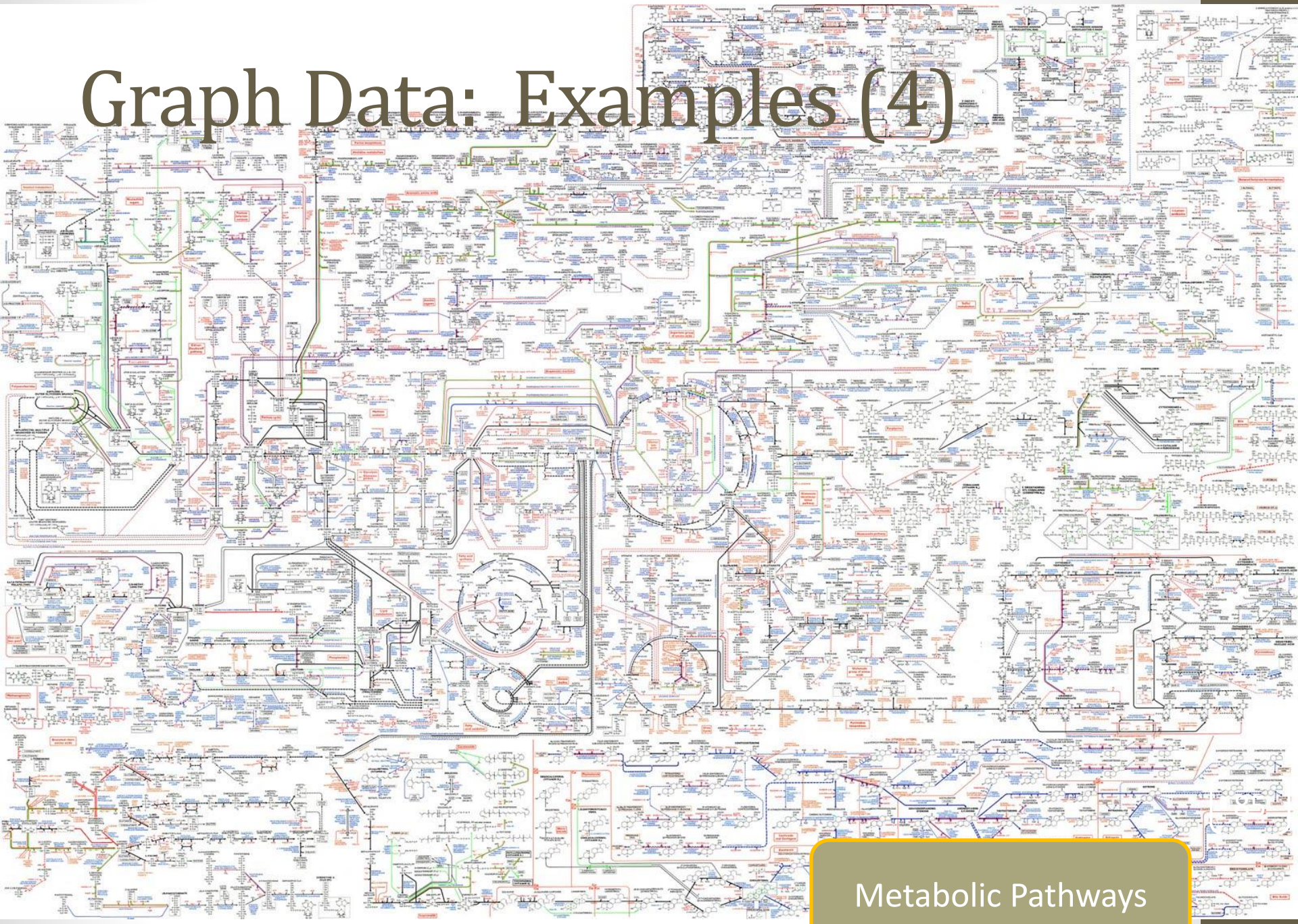
Following Tweets

Graph Data: Examples (3)



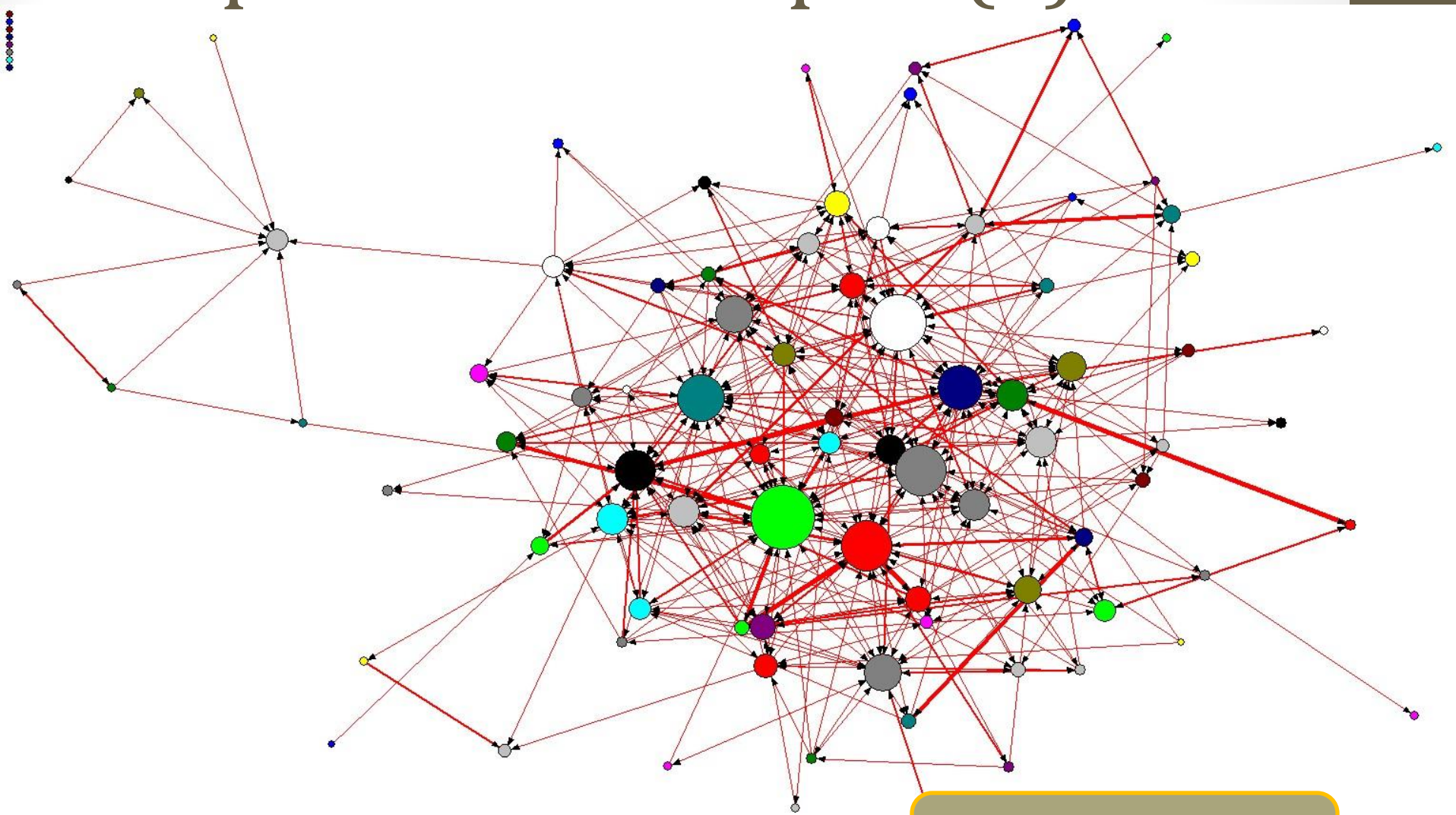
Section of the Internet

Graph Data: Examples (4)



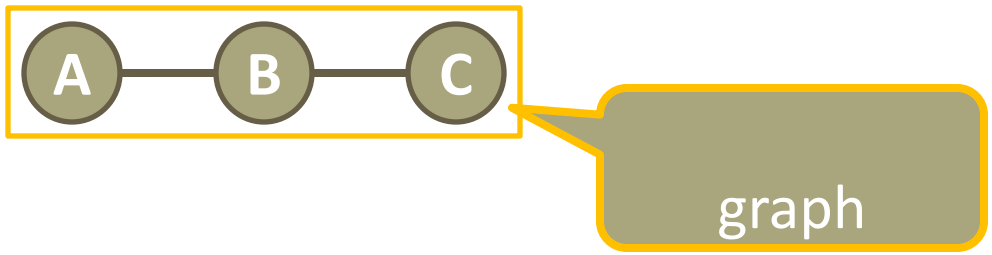
Metabolic Pathways

Graph Data: Examples (5)

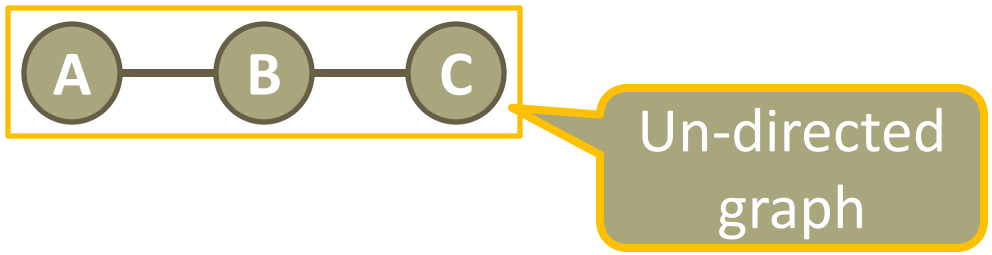


Ranked Web Pages

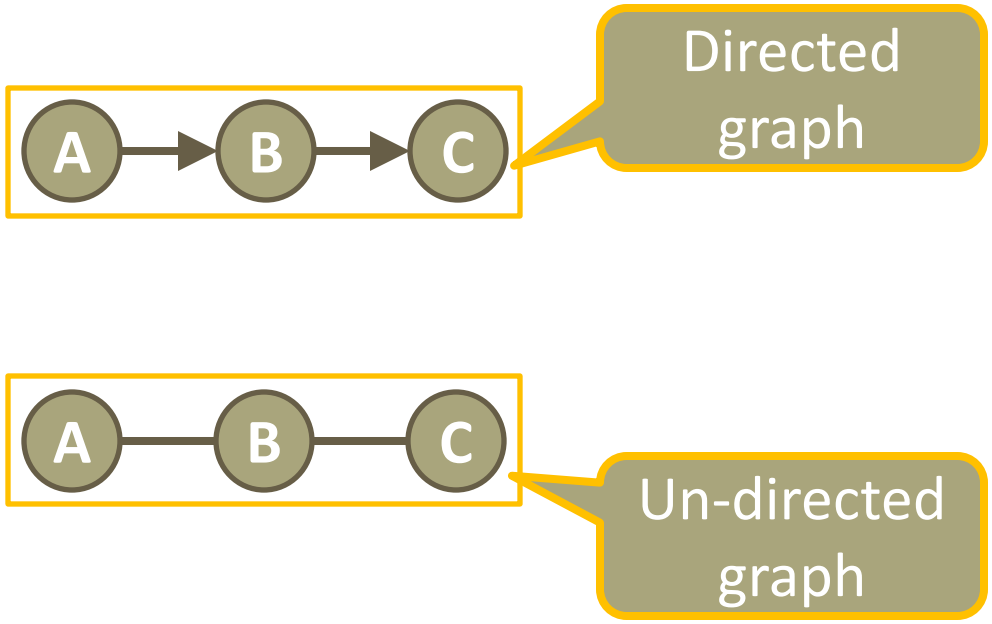
Graph Data: Directed and Un-directed Graphs (0)



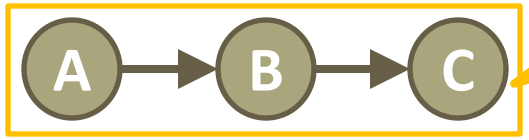
Graph Data: Directed and Un-directed Graphs (1)



Graph Data: Directed and Un-directed Graphs (2)

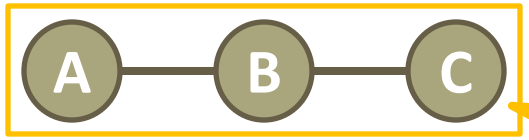


Graph Data: Directed and Un-directed Graphs (3)



Directed graph

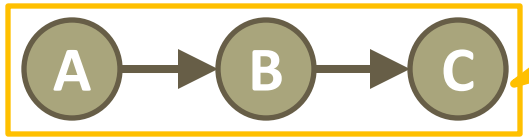
"The link goes from A to B"



Un-directed graph

"The link is between A and B"

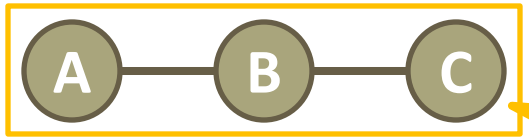
Graph Data: Directed and Un-directed Graphs (4)



Directed graph

"The link goes from A to B"

Example of directed graphs: World Wide Web

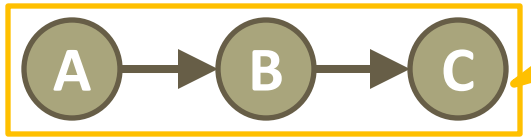


Un-directed graph

"The link is between A and B"

Examples of un-directed graphs: Facebook, LinkedIn

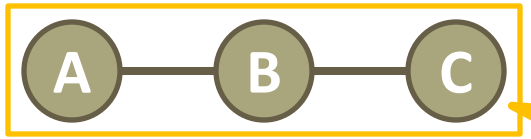
Graph Data: Directed and Un-directed Graphs (5)

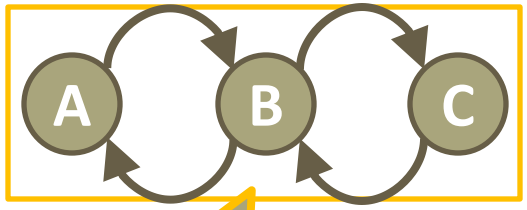


Directed graph

"The link goes from A to B"

Example of directed graphs: World Wide Web





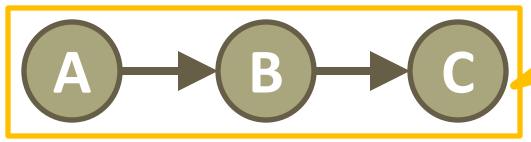
Un-directed graph

"The link is between A and B"

Examples of un-directed graphs: Facebook, LinkedIn

graph with bi-directional links

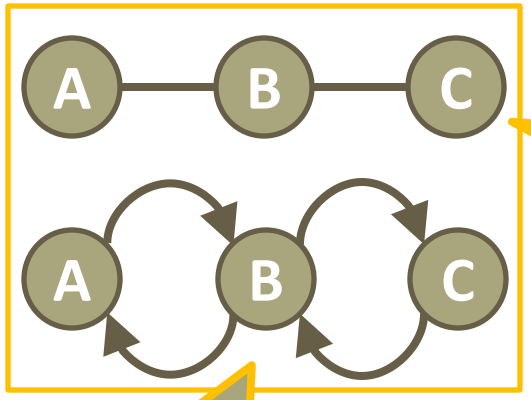
Graph Data: Directed and Un-directed Graphs (6)



Directed graph

"The link goes from A to B"

Example of directed graphs: World Wide Web



Un-directed graph

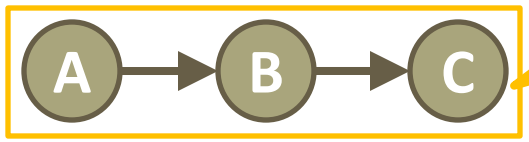
"The link is between A and B"

Examples of un-directed graphs: Facebook, LinkedIn

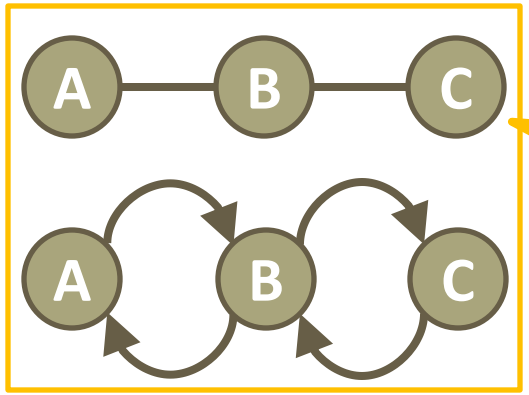
graph with bi-directional links

A graph with bi-directional links is like an undirected graph

Graph Data: Formalize as Rectangular Data (0)



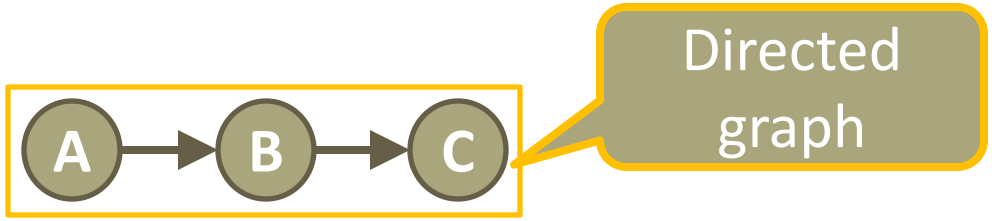
Directed graph



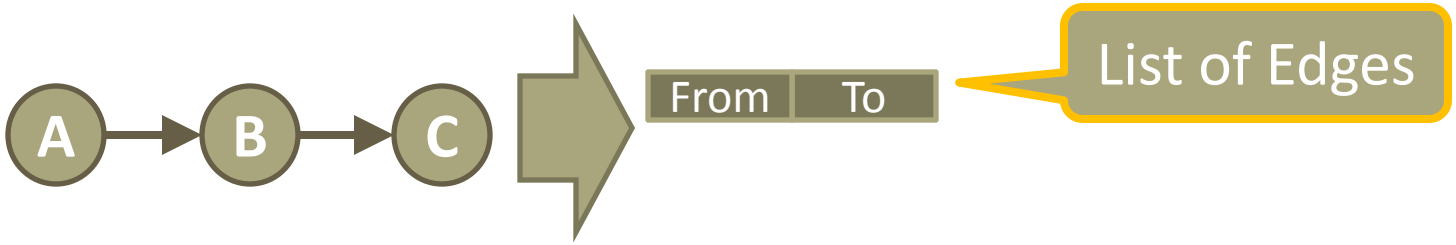
Un-directed graph

- How can we perform operations on graphs?
- How can we formalize graphs as rectangular data?

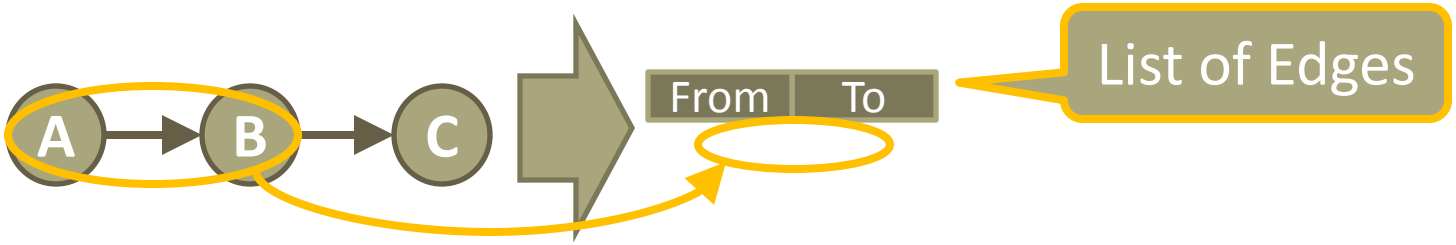
Graph Data: Formalize as Rectangular Data (1)



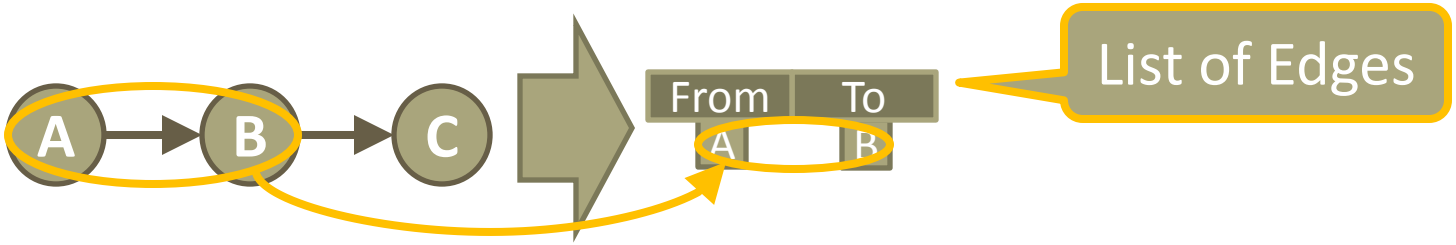
Graph Data: Formalize as Rectangular Data (2)



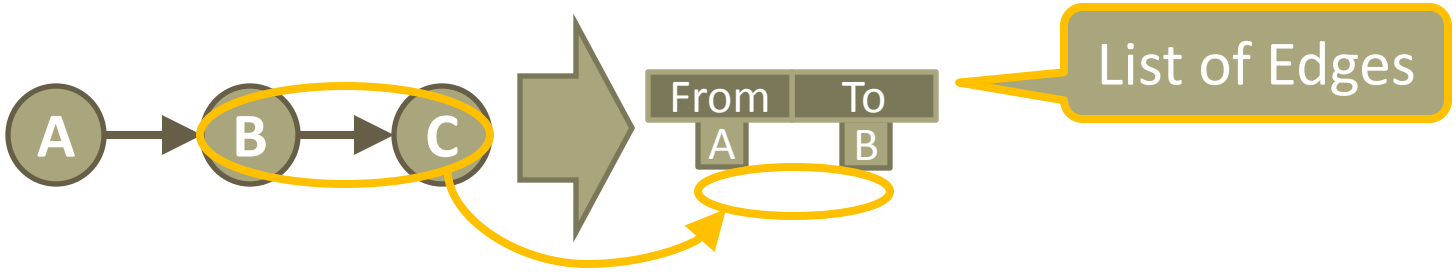
Graph Data: Formalize as Rectangular Data (3)



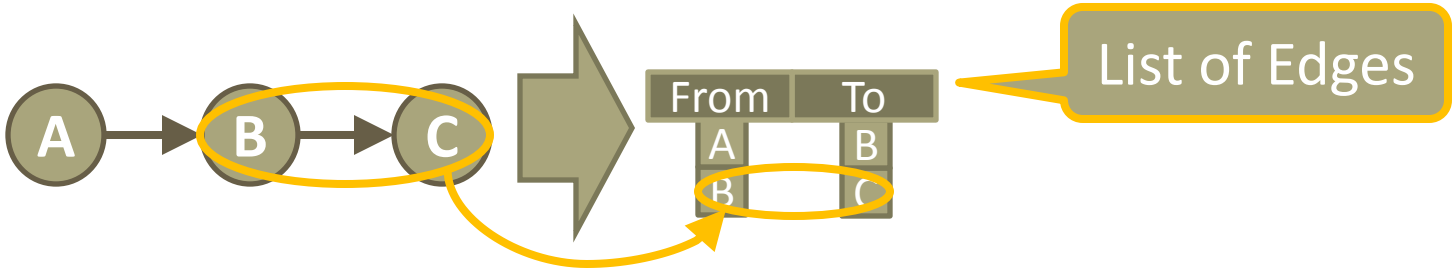
Graph Data: Formalize as Rectangular Data (4)



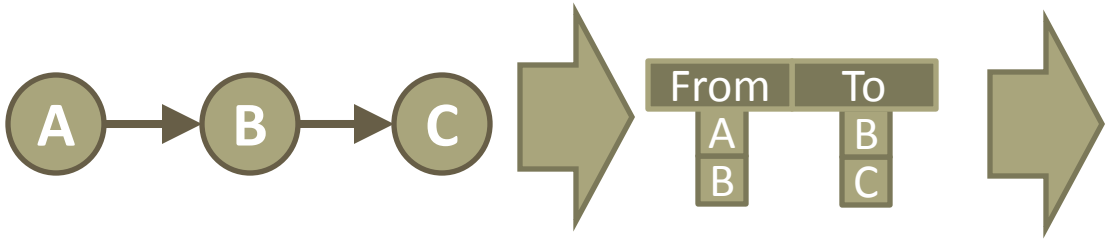
Graph Data: Formalize as Rectangular Data (5)



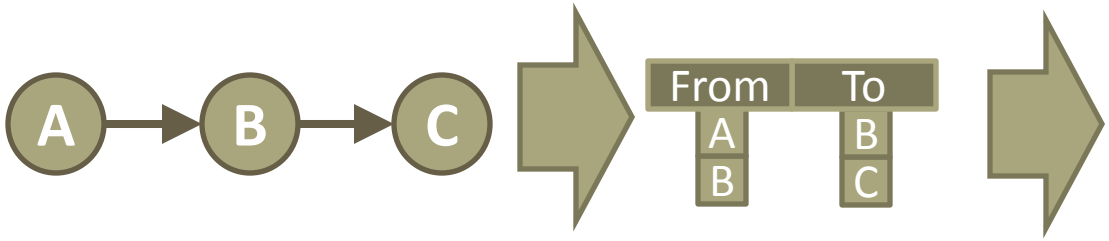
Graph Data: Formalize as Rectangular Data (6)



Graph Data: Formalize as Rectangular Data (7)

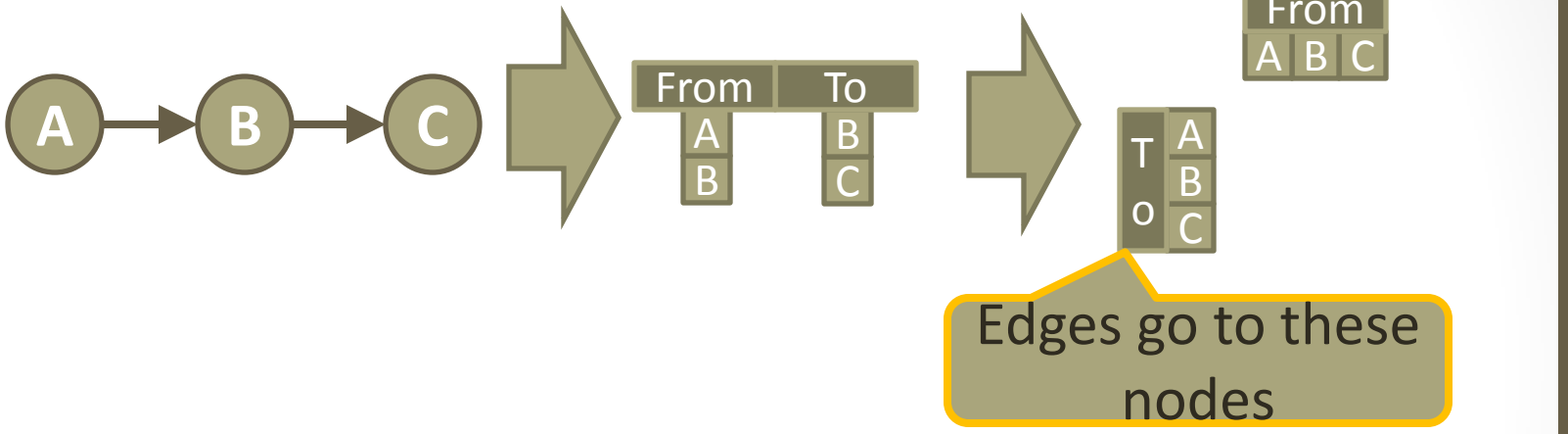


Graph Data: Formalize as Rectangular Data (8)



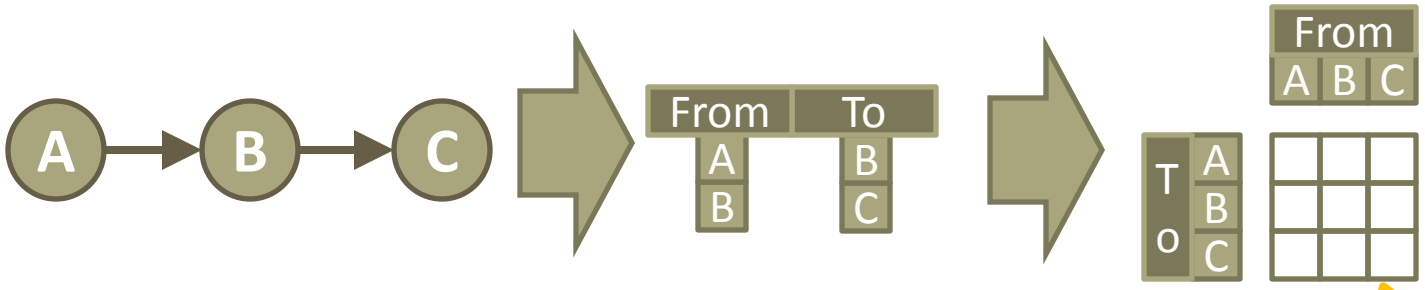
Edges start from these nodes

Graph Data: Formalize as Rectangular Data (9)



Every node could emanate and receive links.

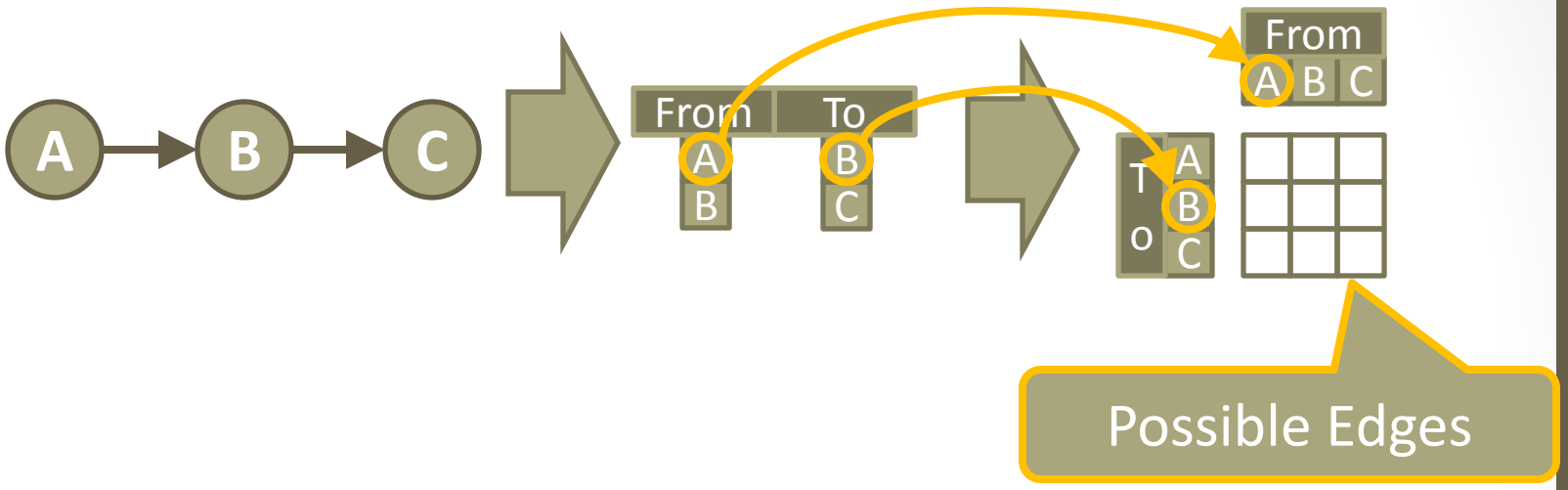
Graph Data: Formalize as Rectangular Data (10)



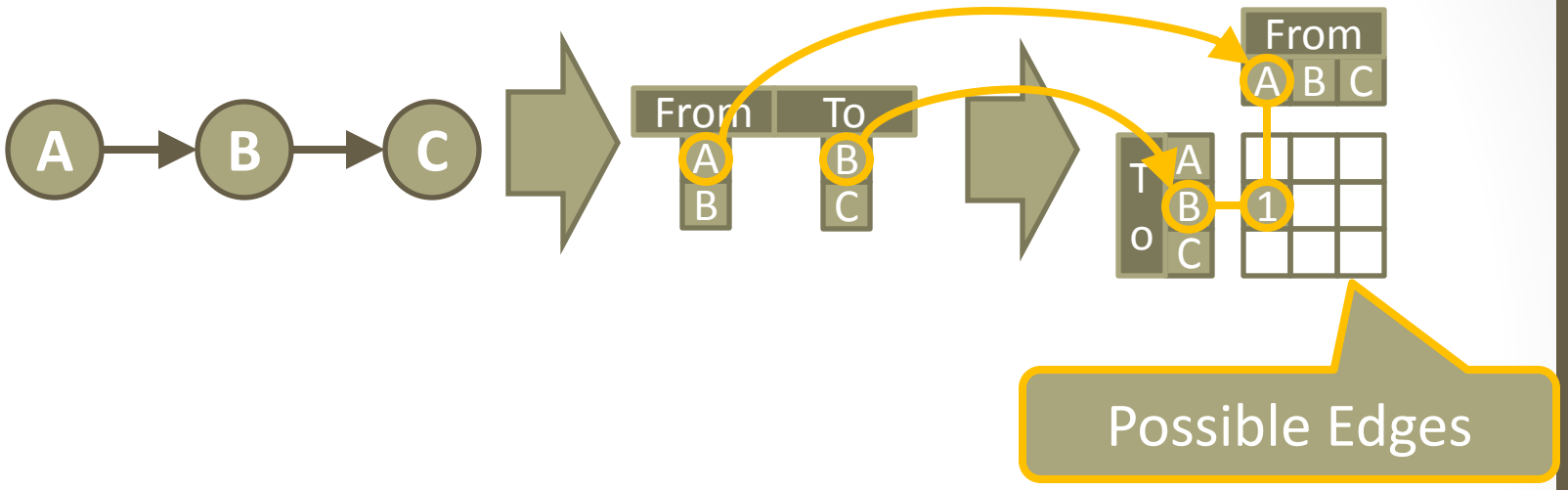
Possible Edges

Every node could emanate and receive links. Therefore the matrix is square.

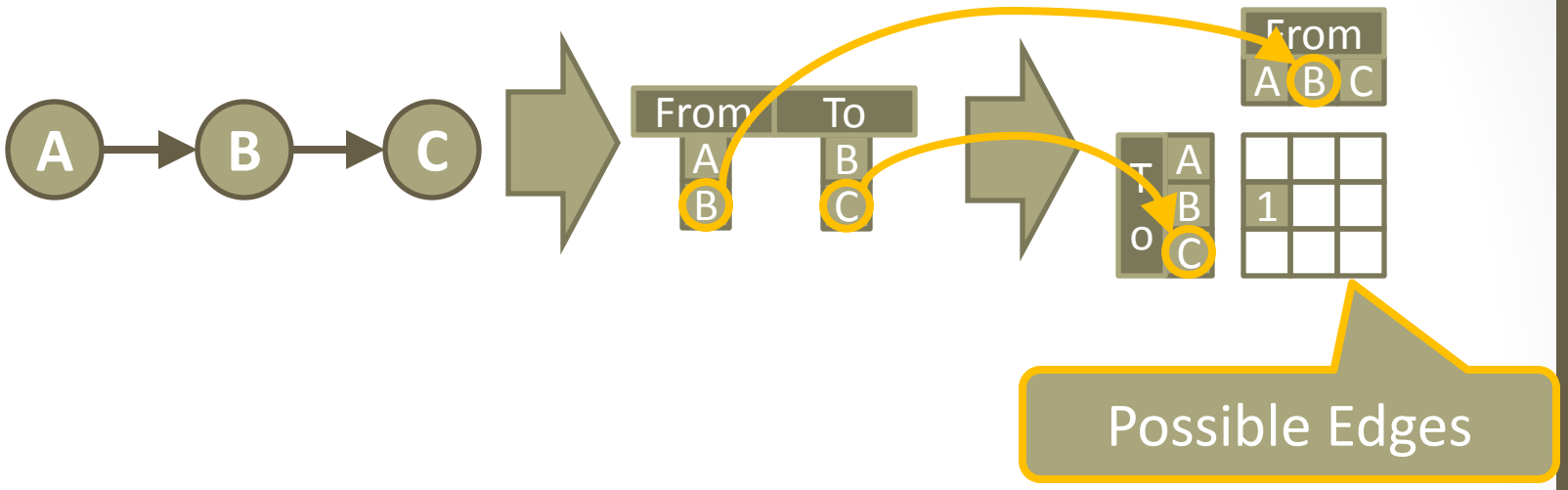
Graph Data: Formalize as Rectangular Data (11)



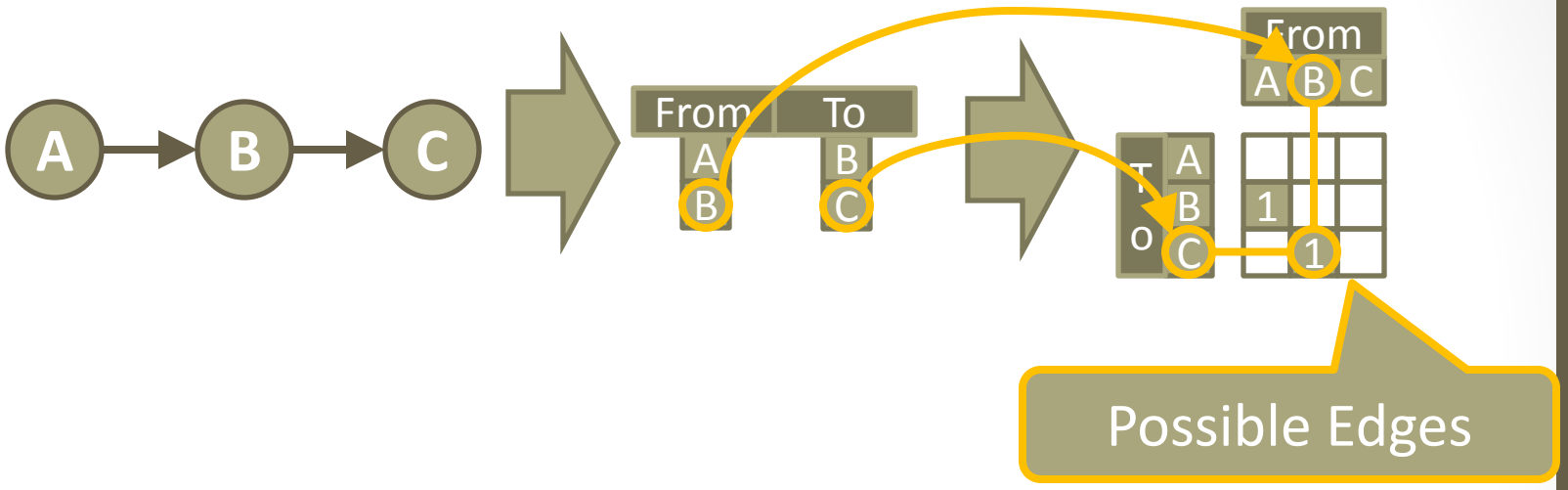
Graph Data: Formalize as Rectangular Data (12)



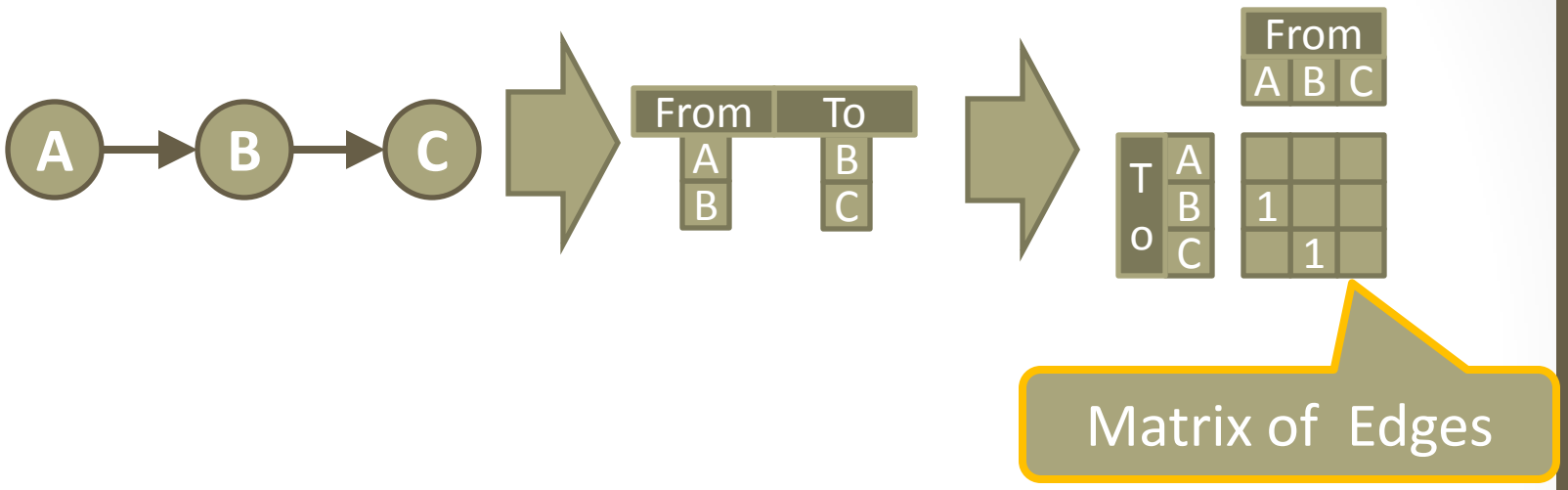
Graph Data: Formalize as Rectangular Data (13)



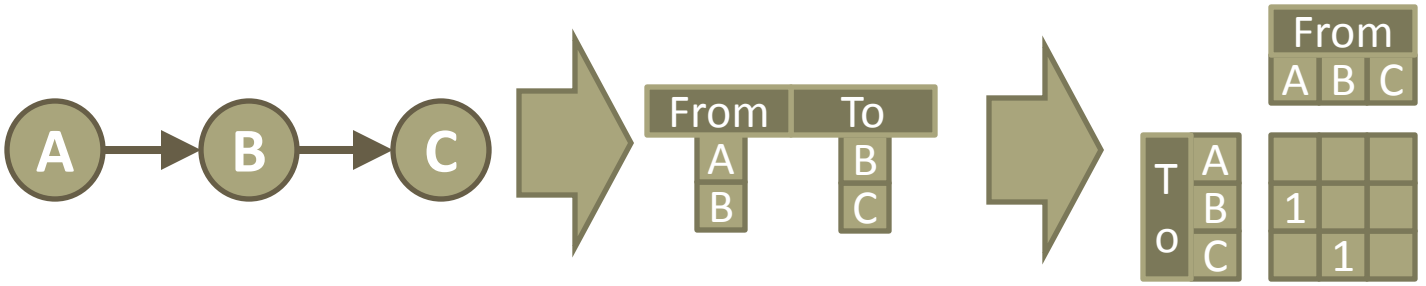
Graph Data: Formalize as Rectangular Data (14)

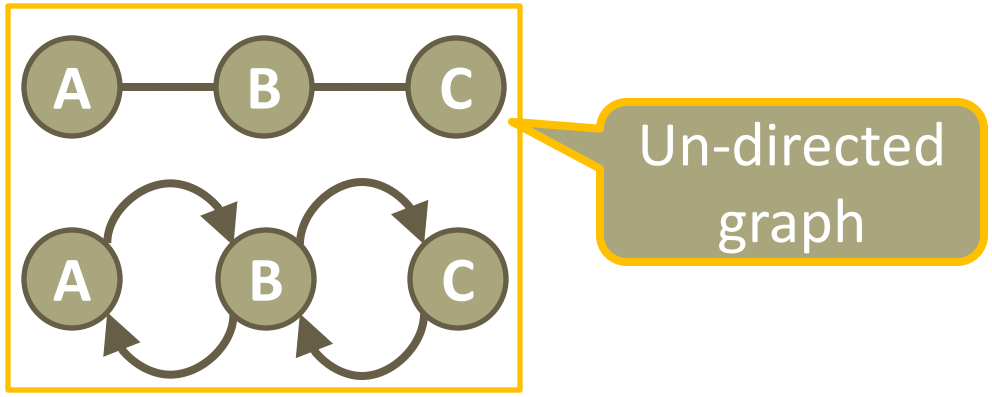


Graph Data: Formalize as Rectangular Data (15)

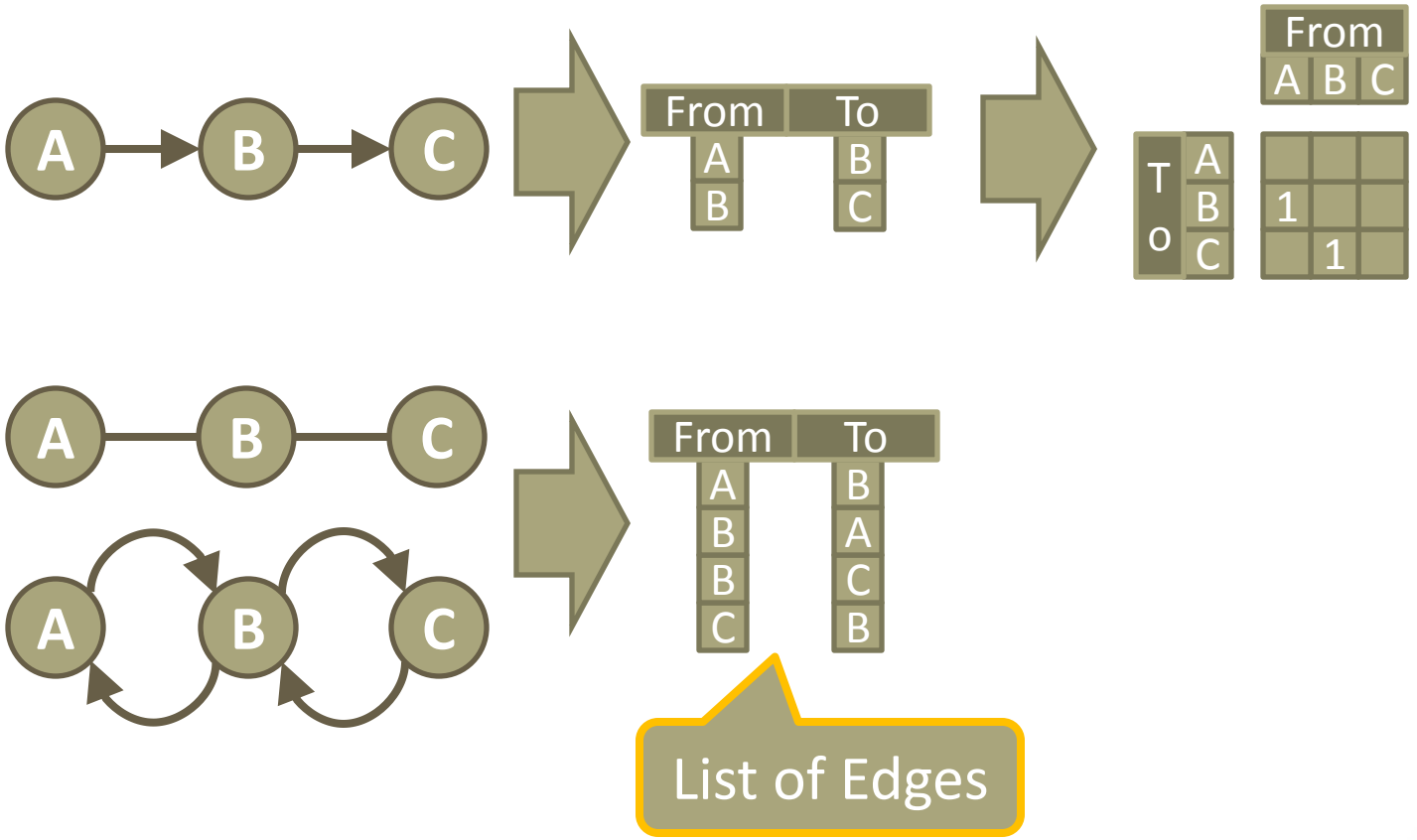


Graph Data: Formalize as Rectangular Data (16)

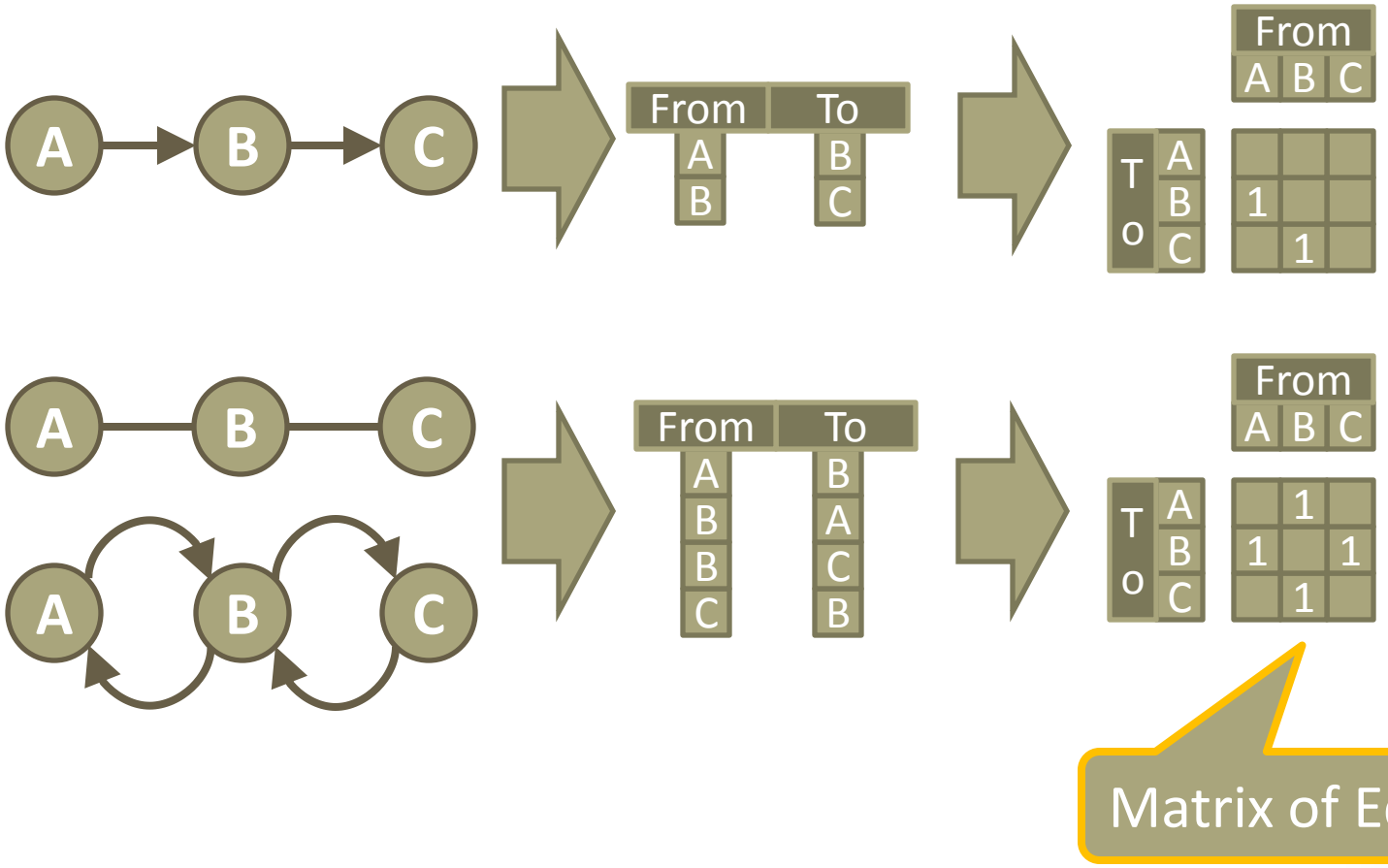




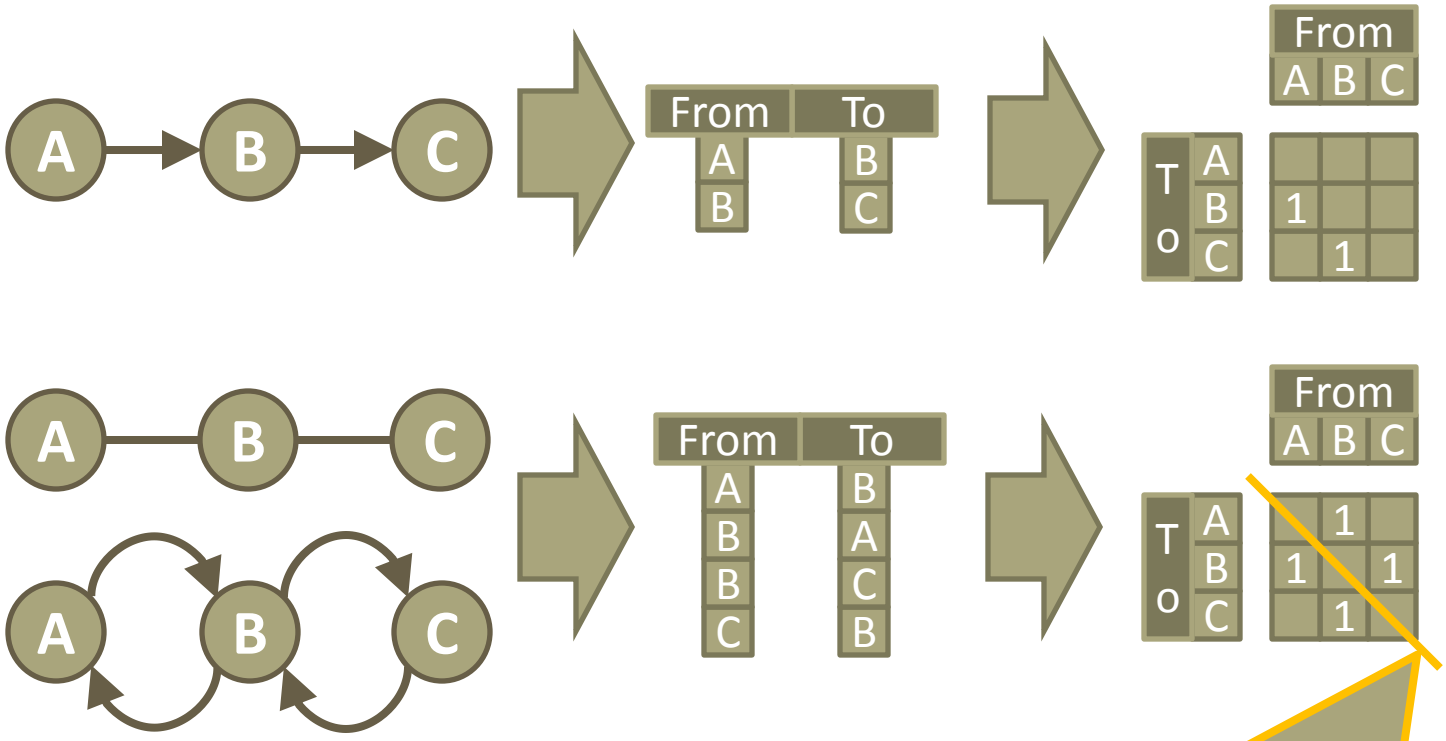
Graph Data: Formalize as Rectangular Data (17)



Graph Data: Formalize as Rectangular Data (18)

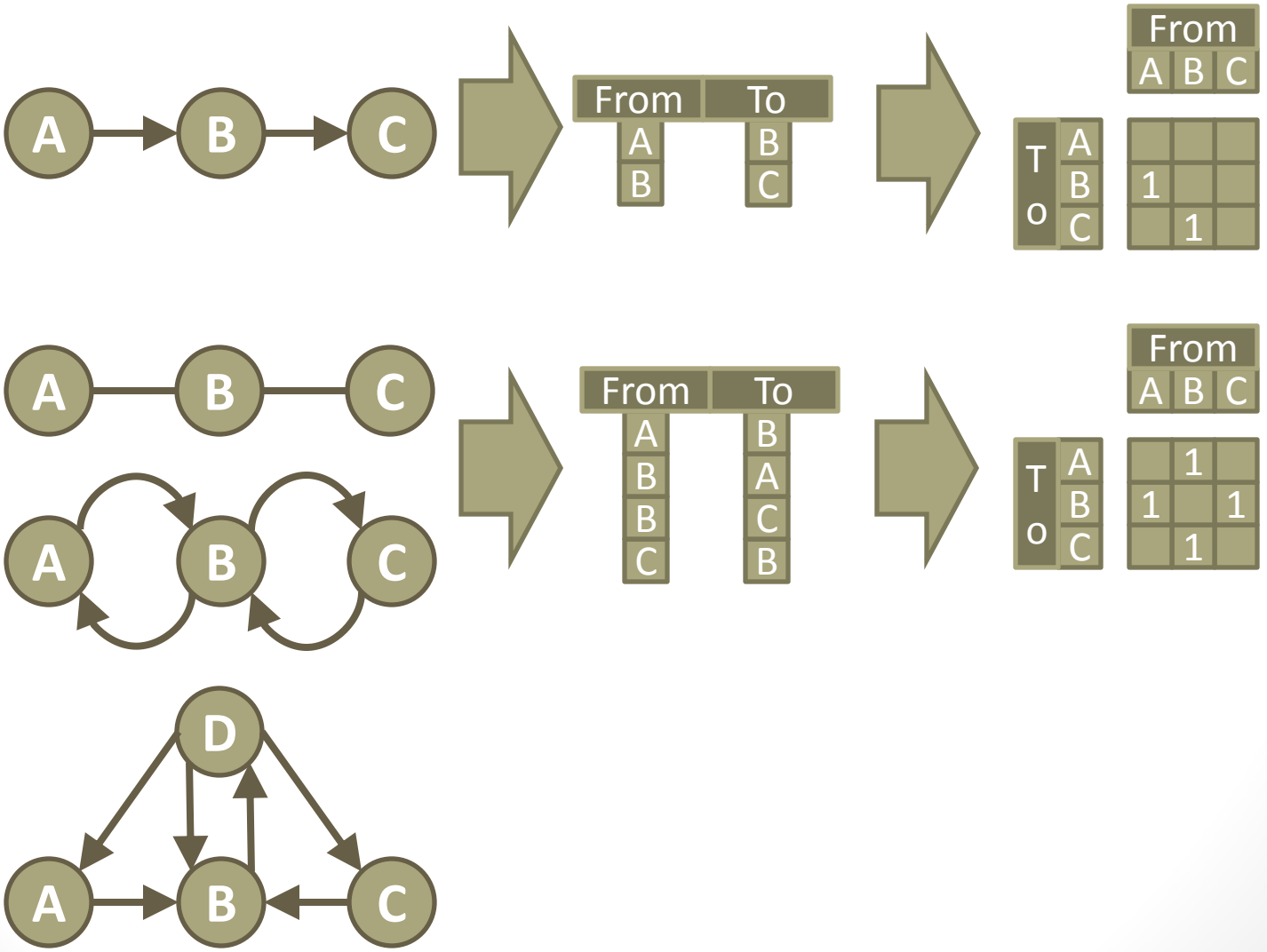


Graph Data: Formalize as Rectangular Data (19)

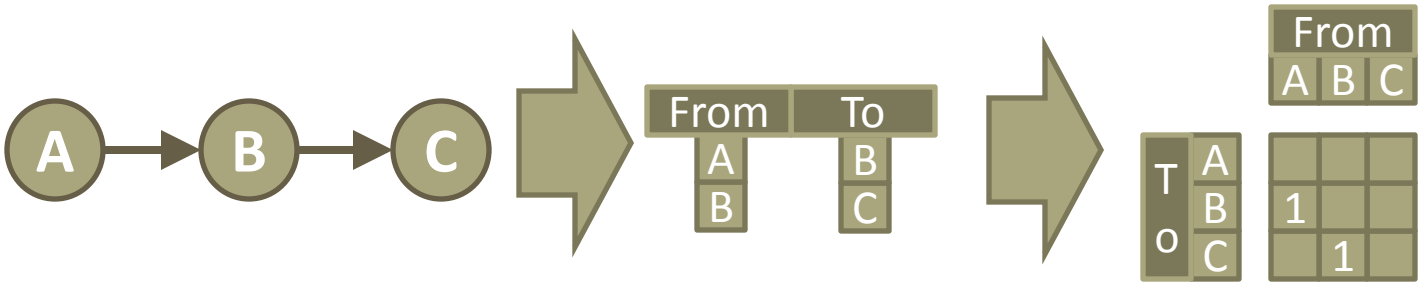


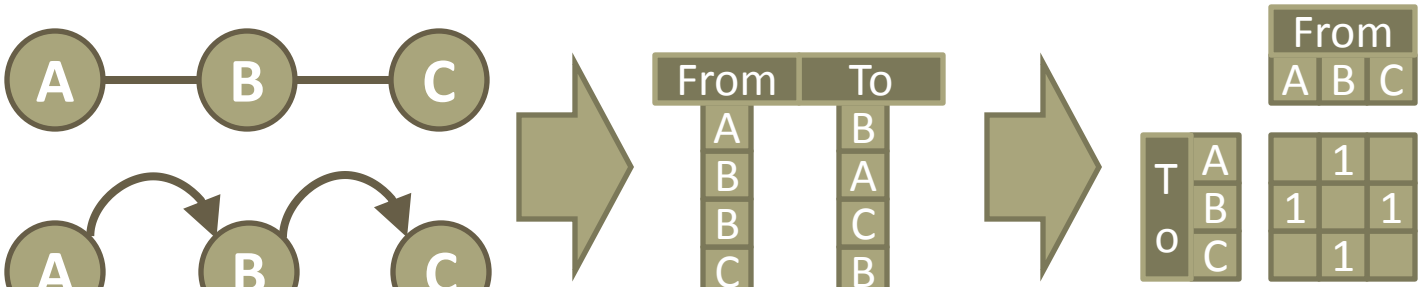
Note: undirected or bi-directional graphs Have symmetric matrices

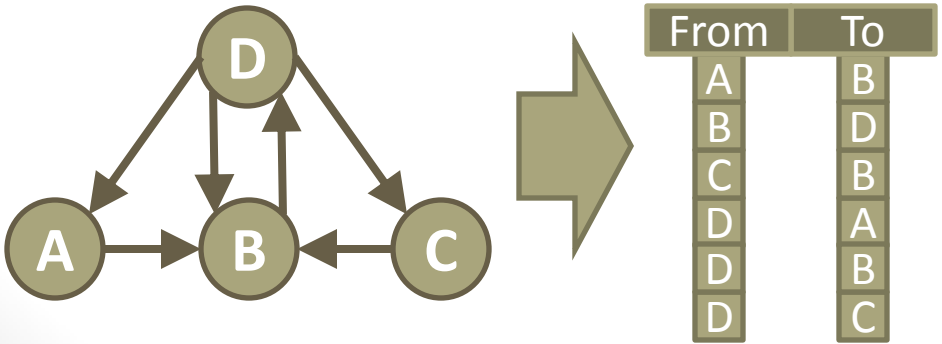
Graph Data: Formalize as Rectangular Data (20)



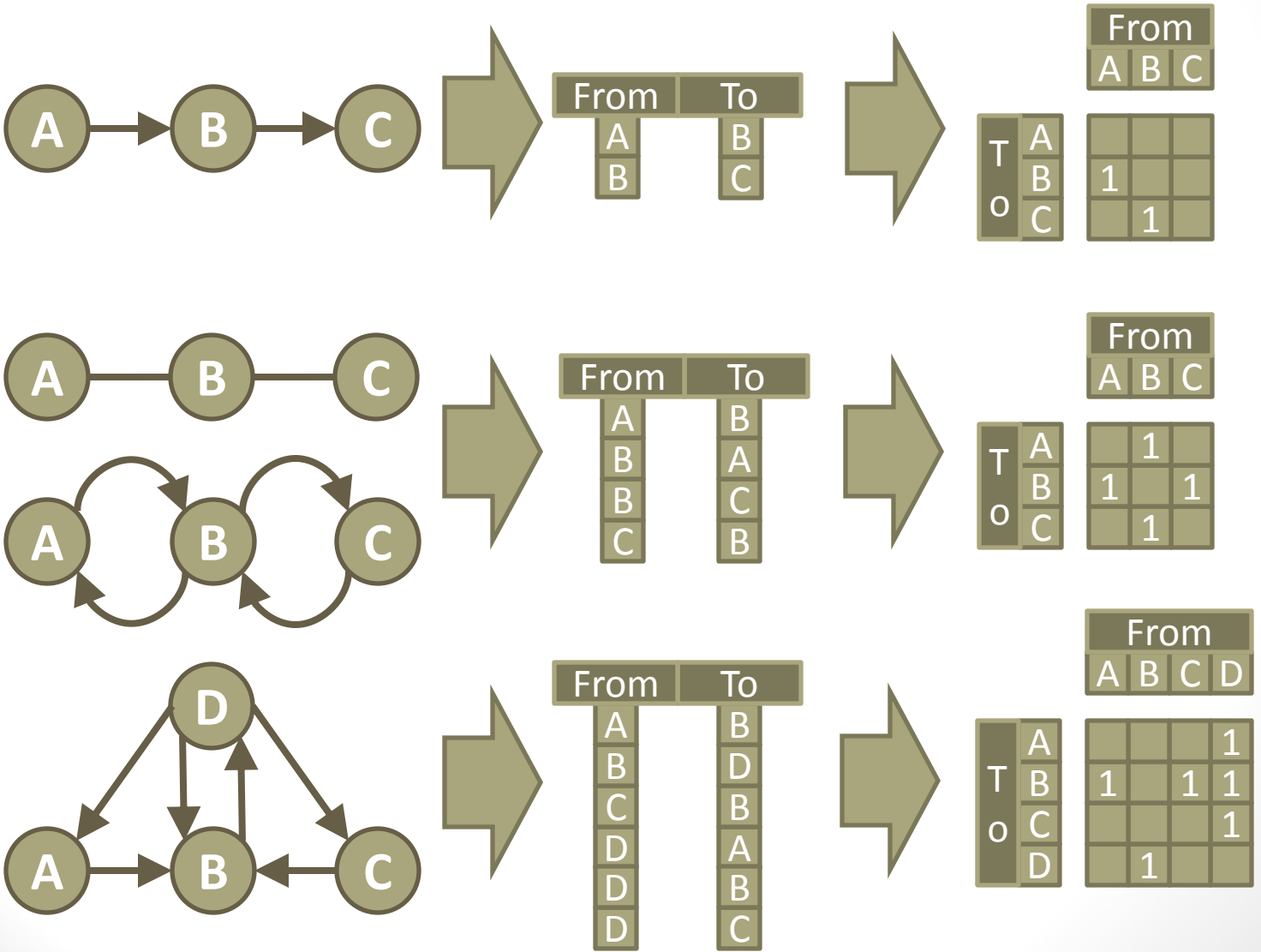
Graph Data: Formalize as Rectangular Data (21)



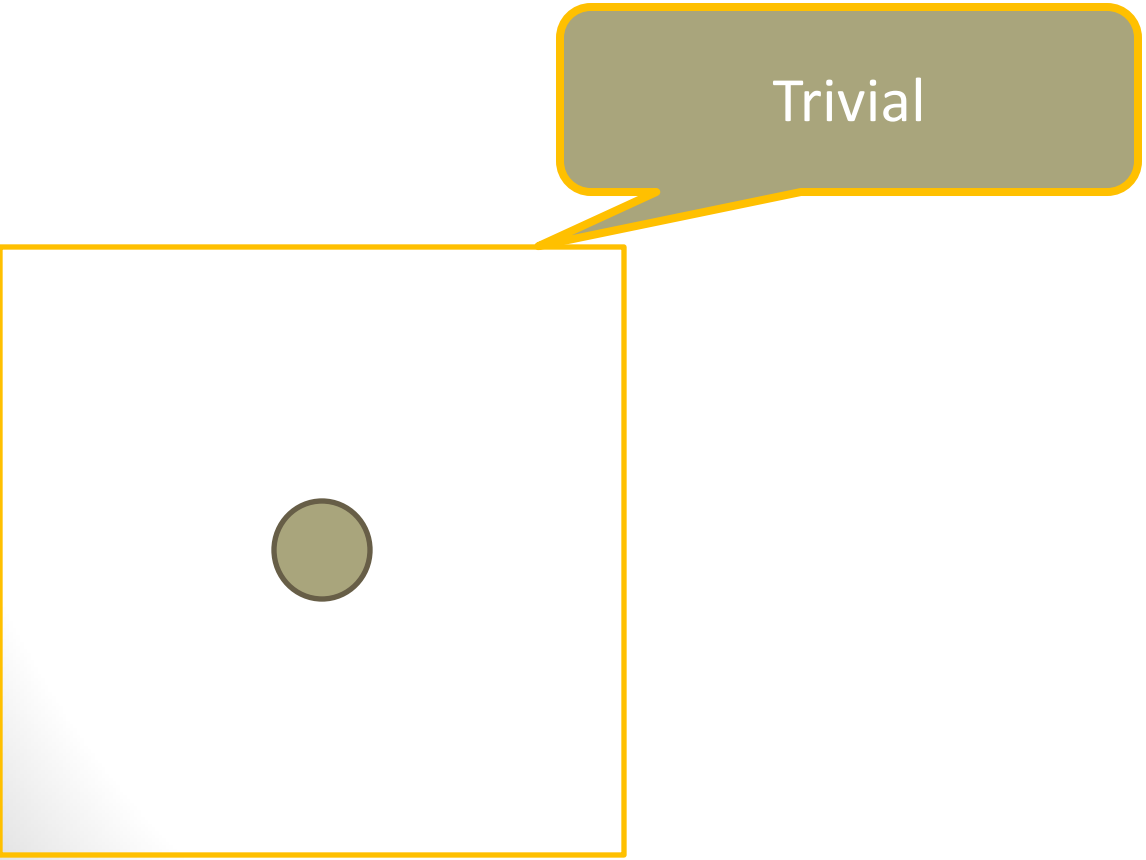




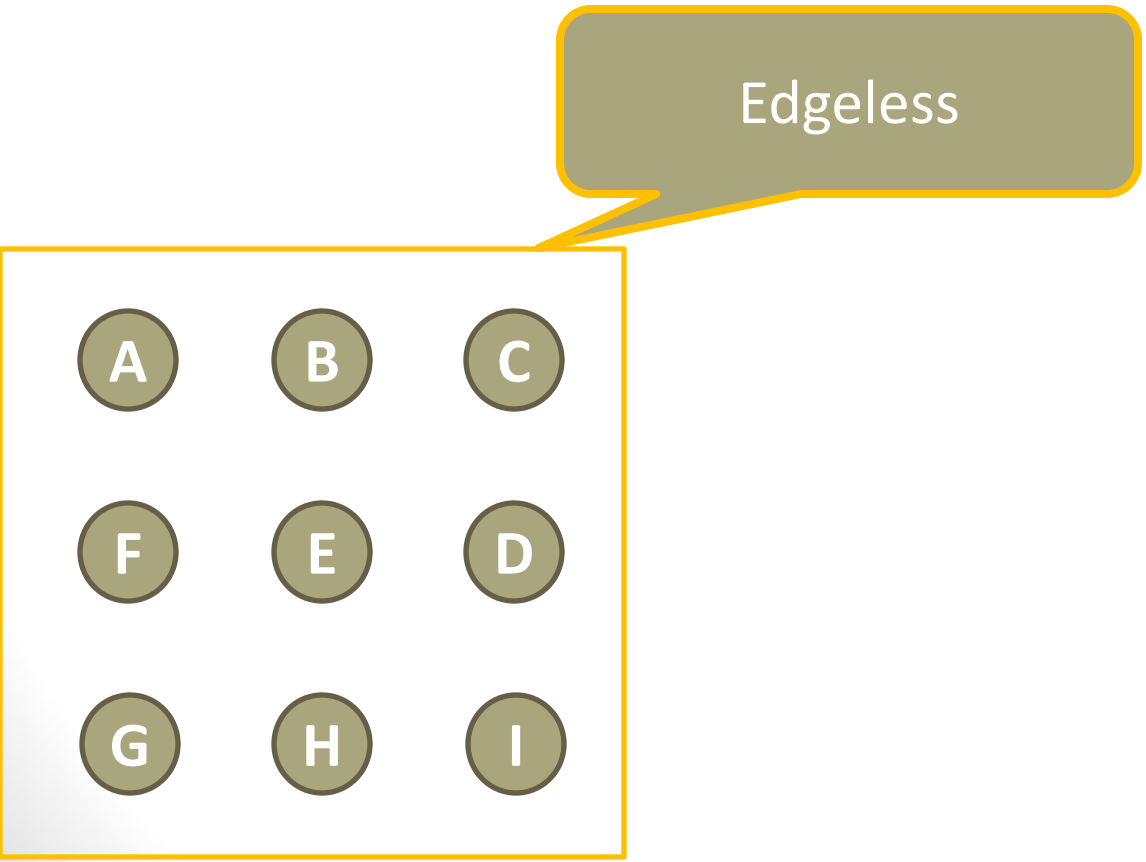
Graph Data: Formalize as Rectangular Data (22)



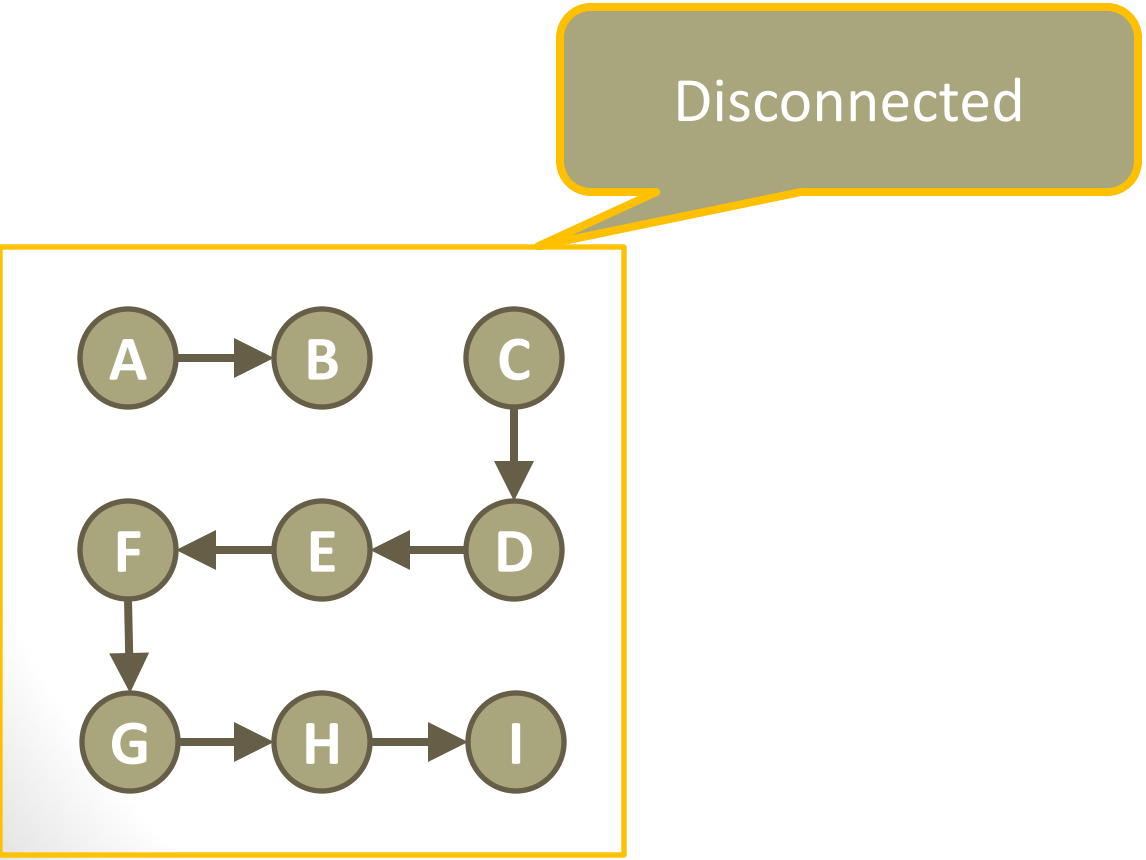
Graph Data: Connectedness and Density (0)



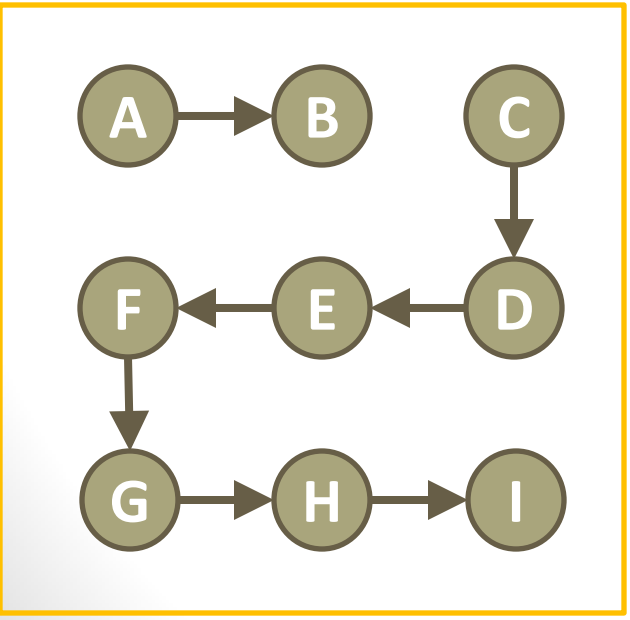
Graph Data: Connectedness and Density (1)

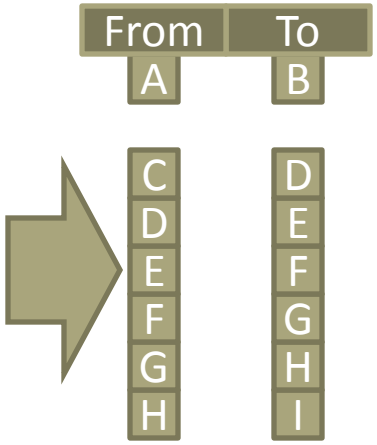


Graph Data: Connectedness and Density (2)

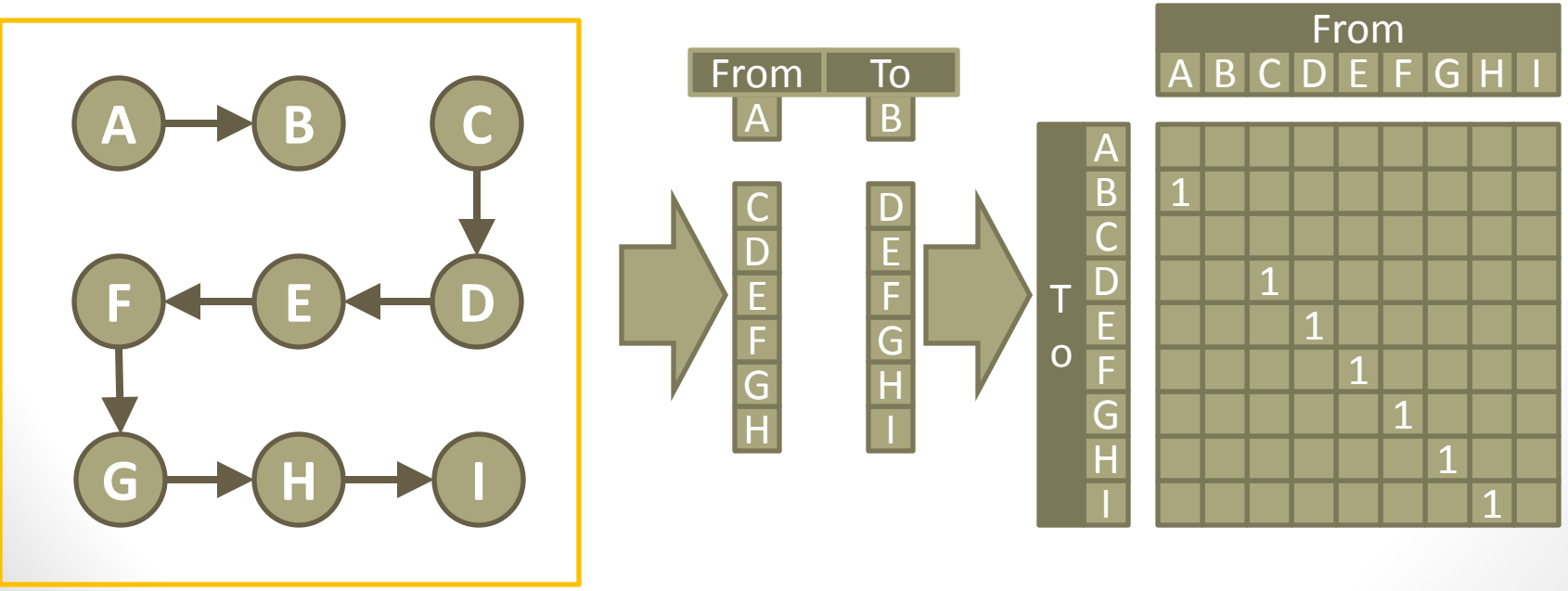


Graph Data: Connectedness and Density (3)

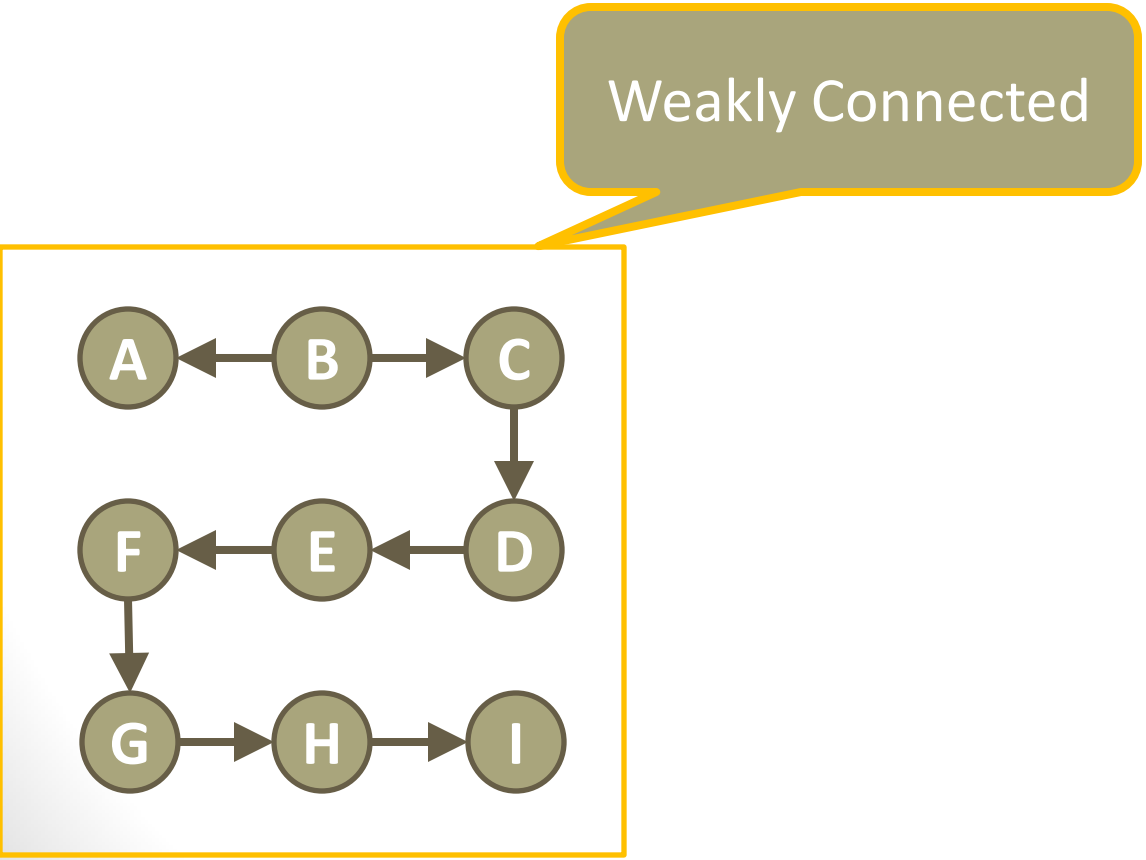




Graph Data: Connectedness and Density (4)

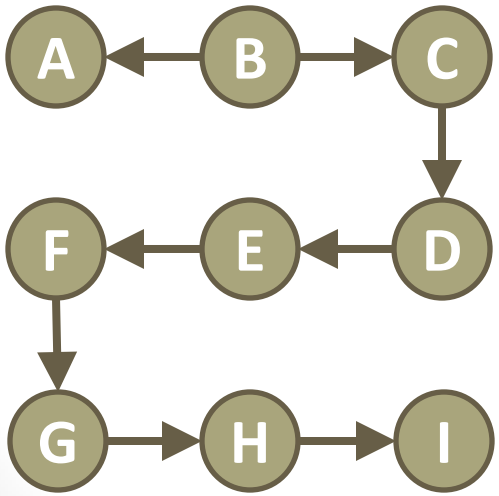


Graph Data: Connectedness and Density (5)



Graph Data: Connectedness and Density (6)

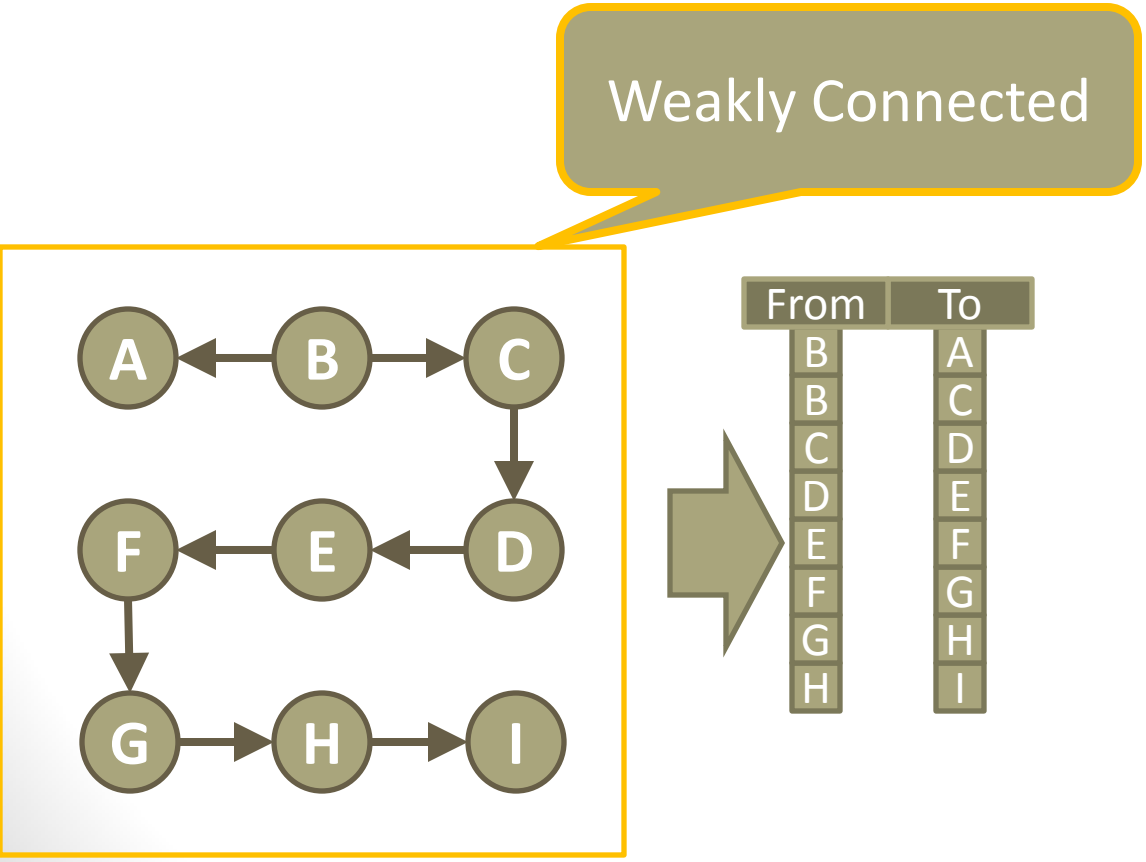
Weakly Connected



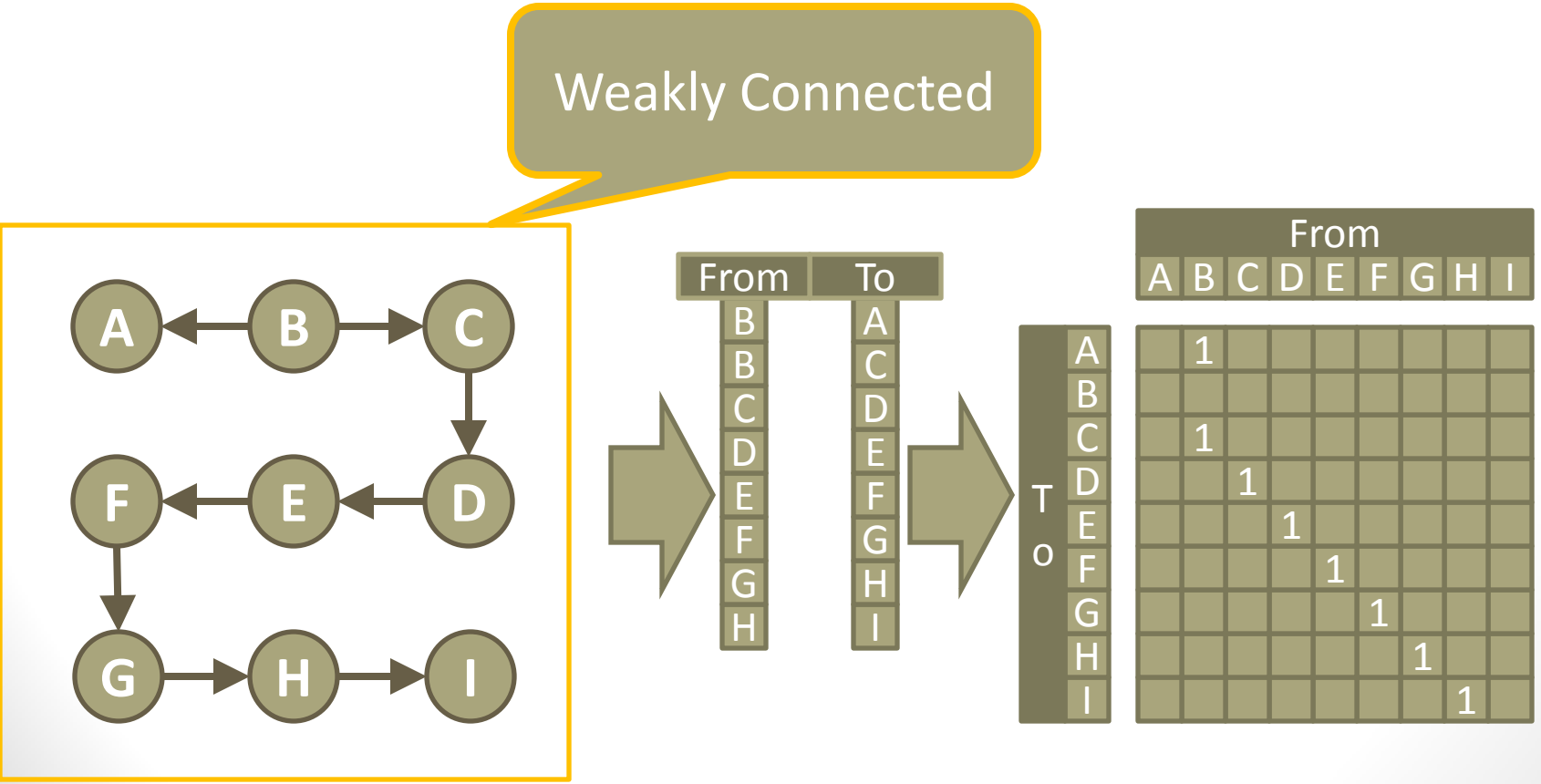
Weakly Connected:
For every pair of vertices, a path exists along the graph's edges regardless of the edge's direction. There exists at least one pair of vertices (u, v), where a path does not exist along the graph's directed edges from u to v an does not exist from v to u.

In this case: A path does not exist from A to C and does not exist from C to A.

Graph Data: Connectedness and Density (7)

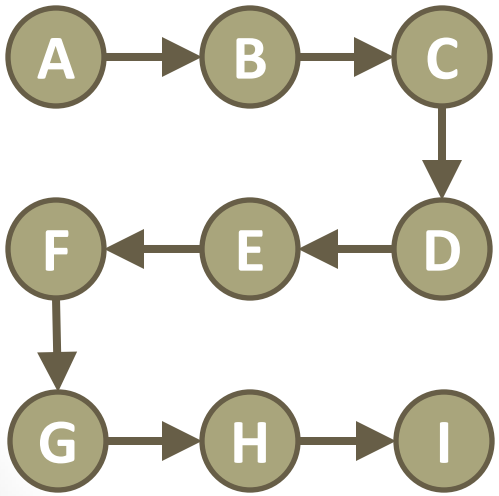


Graph Data: Connectedness and Density (8)



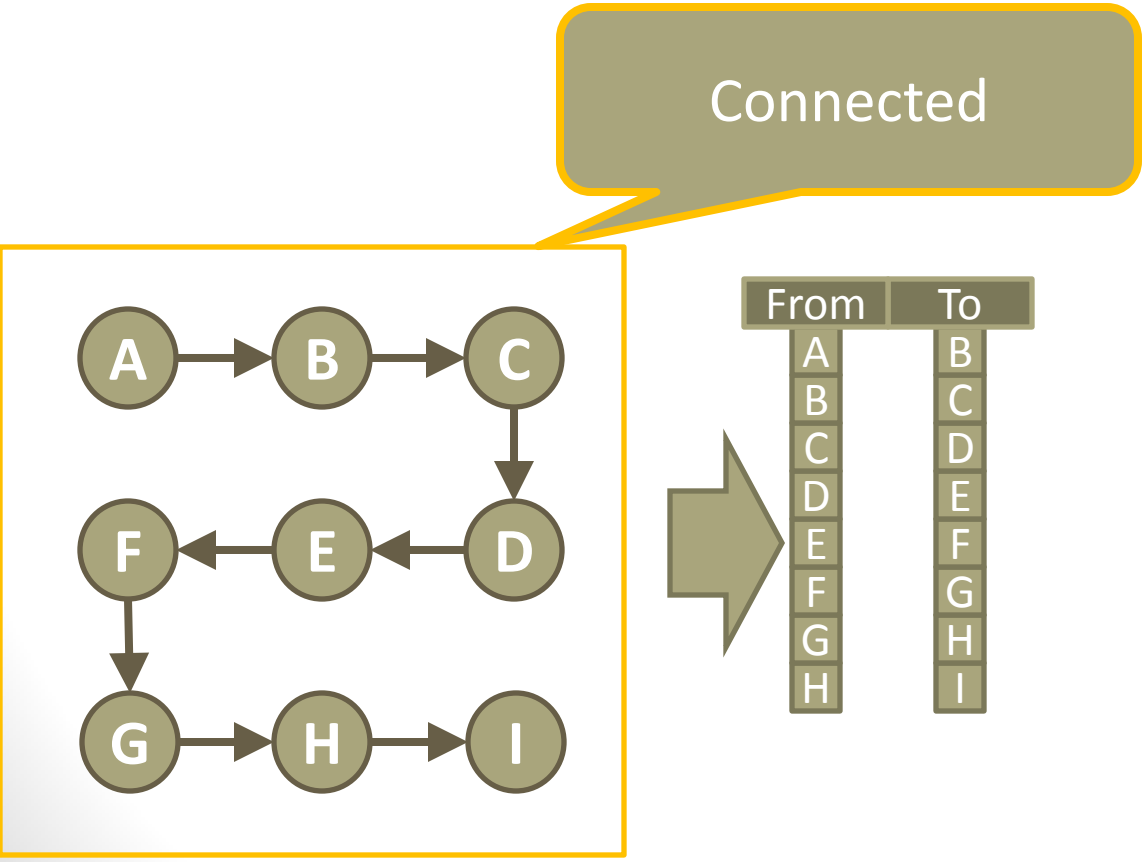
Graph Data: Connectedness and Density (9)

Connected

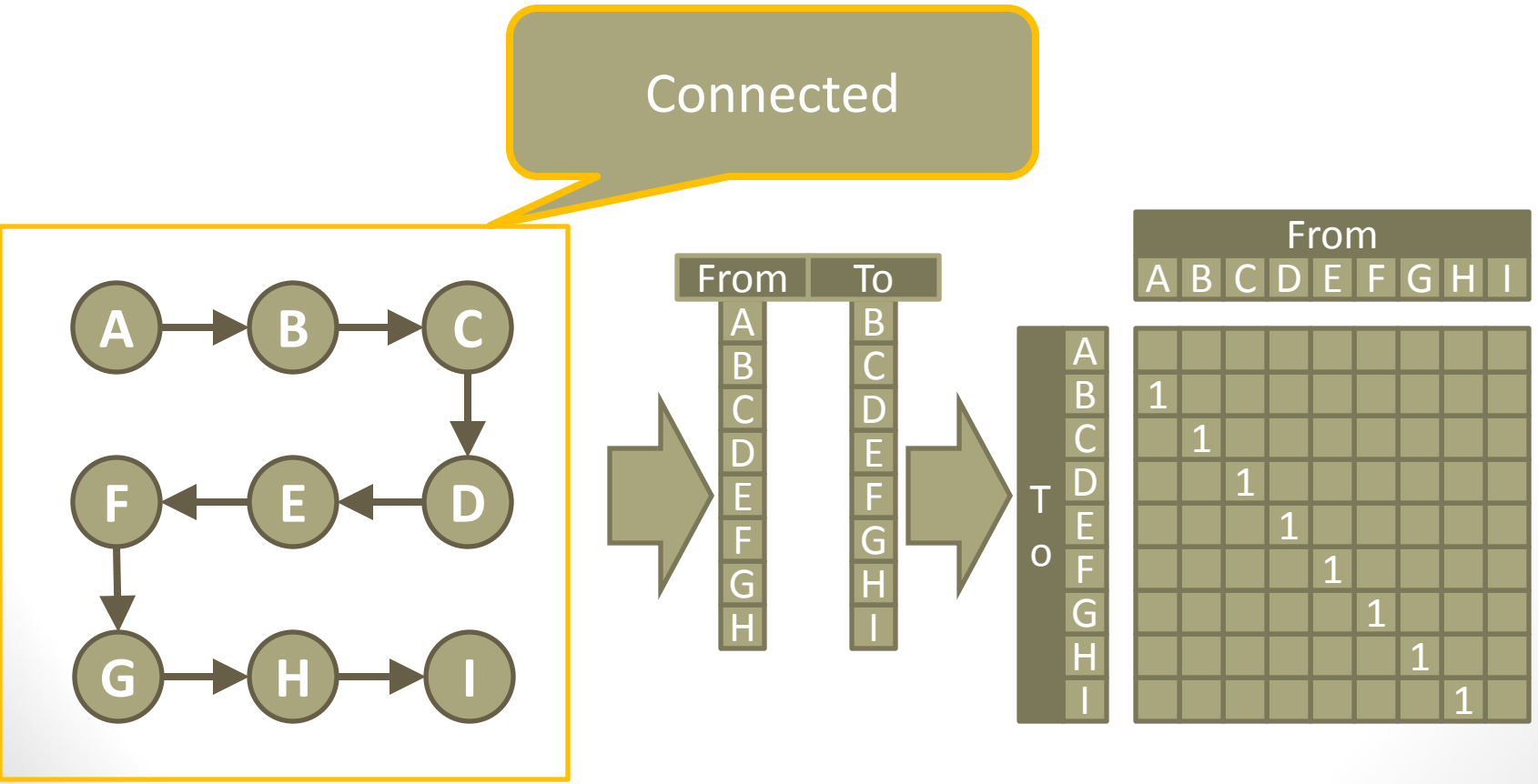


Connected:
For every pair of vertices (u, v), a path exists along the graph's directed edges from u to v or from v to u. There exists at least one pair of vertices (u, v), where a path does not exist along the graph's directed edges from either u to v or from v to u.

Graph Data: Connectedness and Density (10)

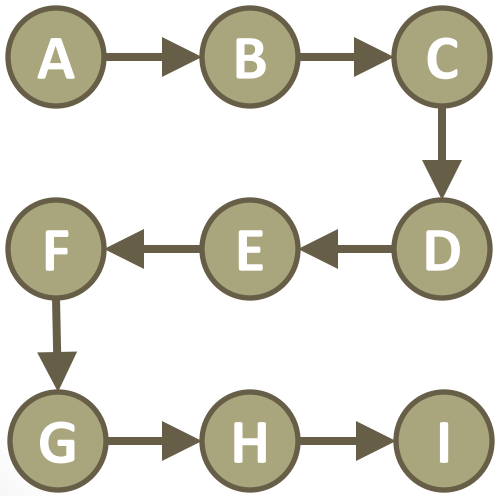


Graph Data: Connectedness and Density (11)



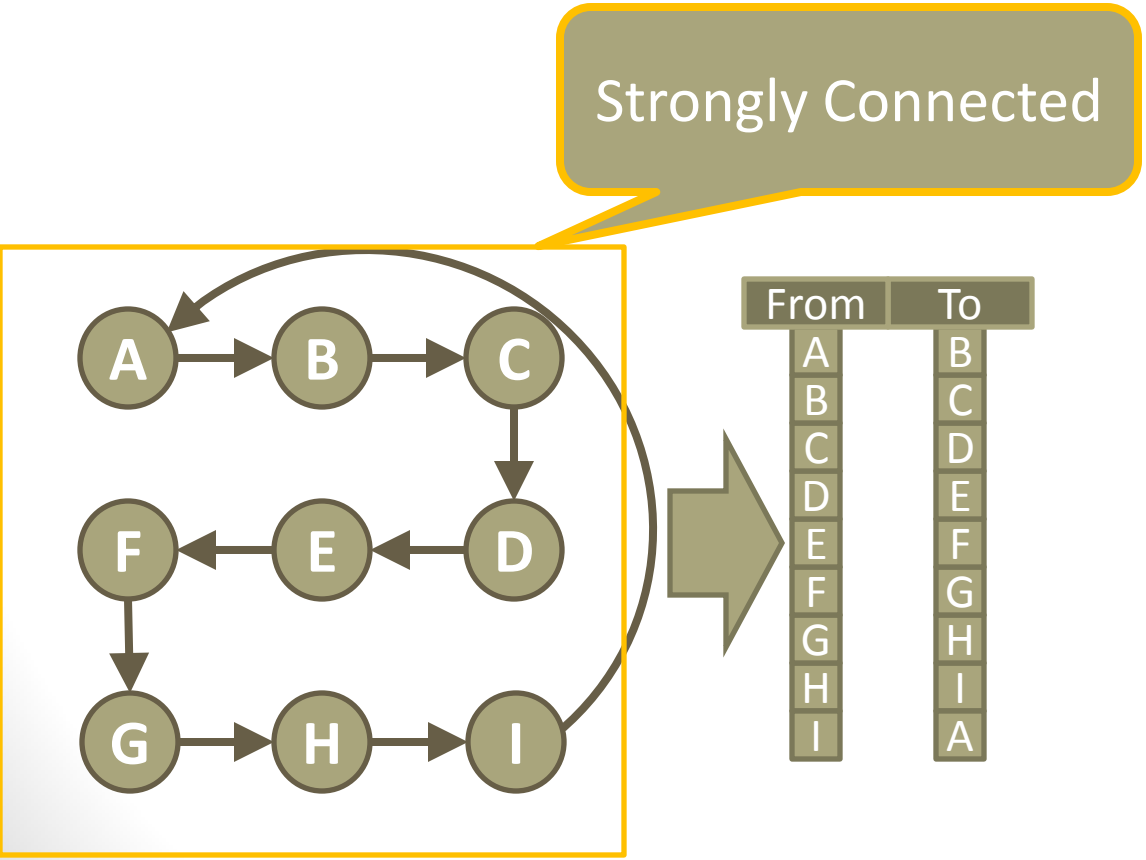
Graph Data: Connectedness and Density (12)

Connected

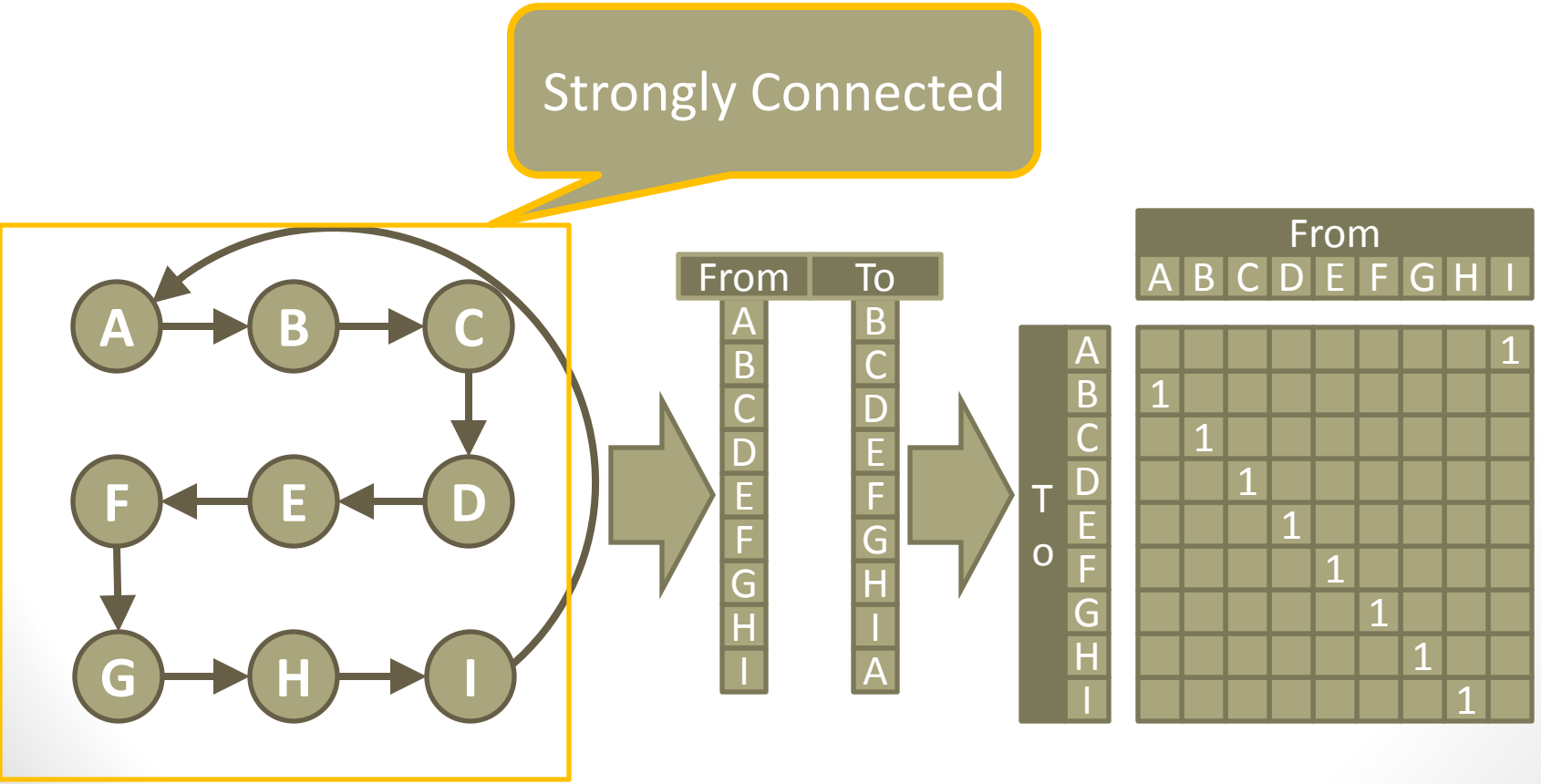


Connected:
A path exists along the graph's directed edges from every node to every node

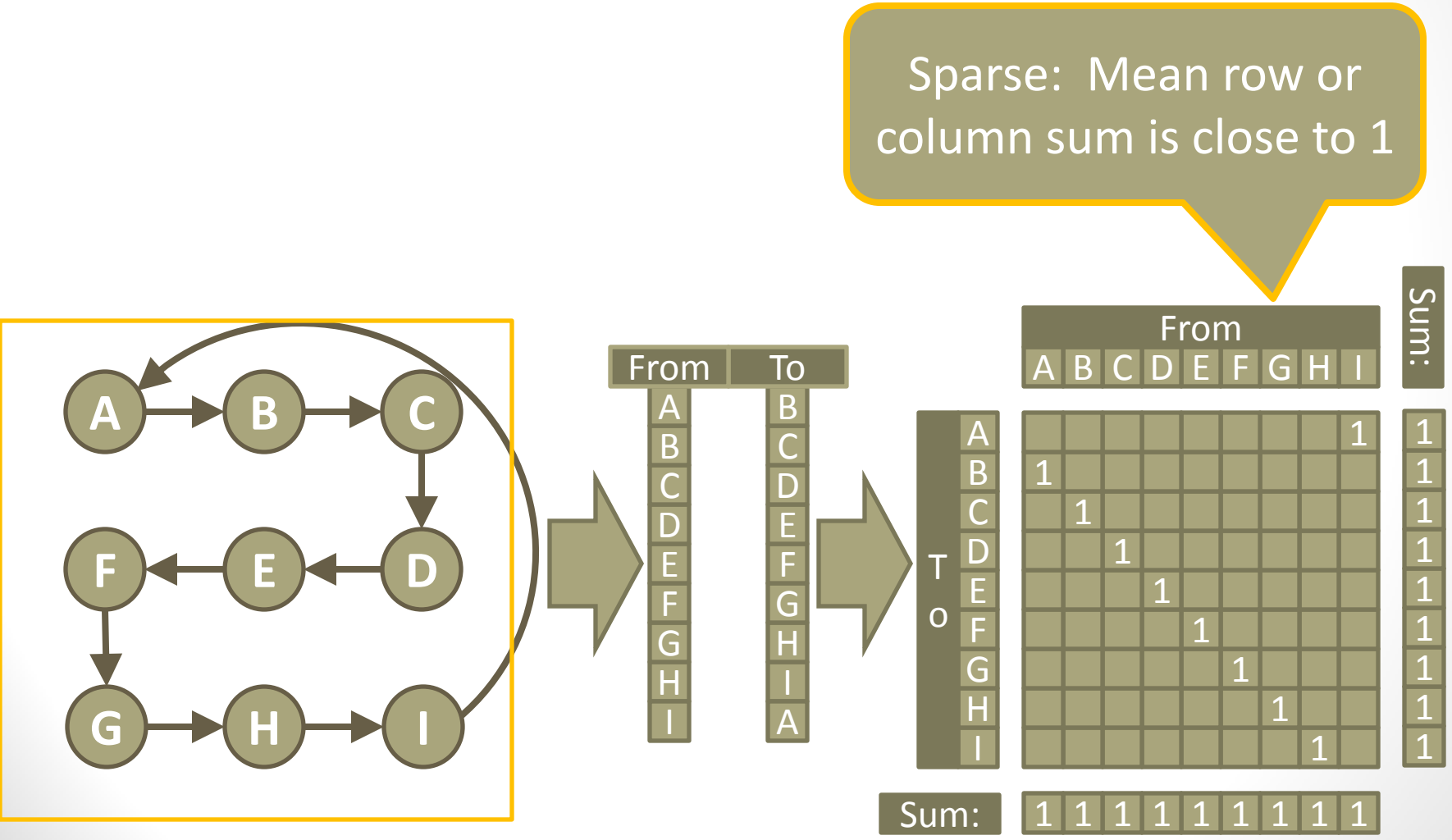
Graph Data: Connectedness and Density (13)



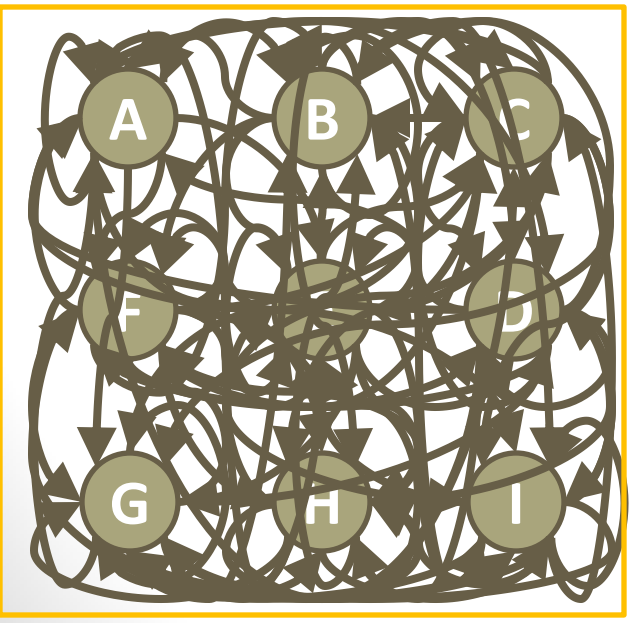
Graph Data: Connectedness and Density (14)



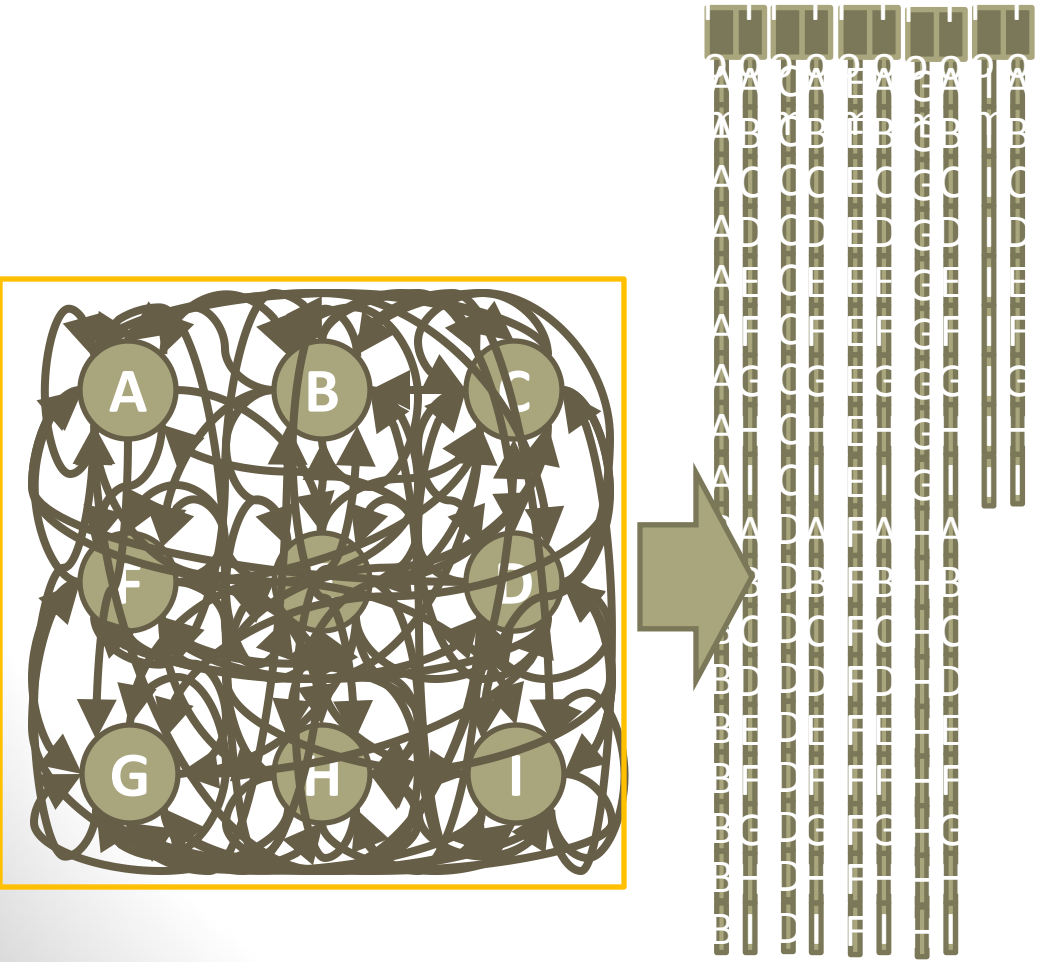
Graph Data: Connectedness and Density (15)



Graph Data: Connectedness and Density (16)



Graph Data: Connectedness and Density (17)



Graph Data: Connectedness and Density (18)

From	To
A	A
A	B
A	C
A	D
A	E
A	F
A	G
A	H
A	I
B	A
B	B
B	C
B	D
B	E
B	F
B	G
B	H
B	I

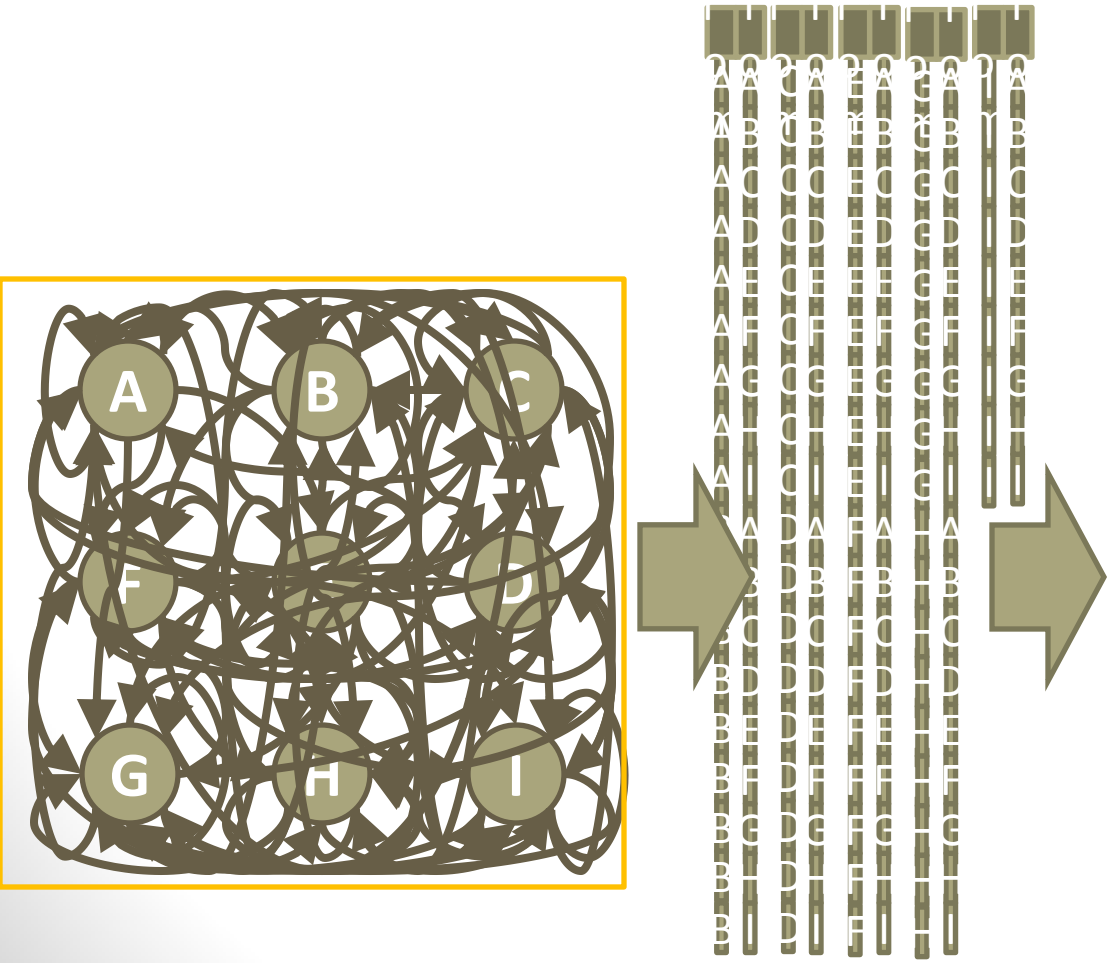
From	To
C	A
C	B
C	C
C	D
C	E
C	F
C	G
C	H
C	I
D	A
D	B
D	C
D	D
D	E
D	F
D	G
D	H
D	I

From	To
E	A
E	B
E	C
E	D
E	E
E	F
E	G
E	H
E	I
F	A
F	B
F	C
F	D
F	E
F	F
F	G
F	H
F	I

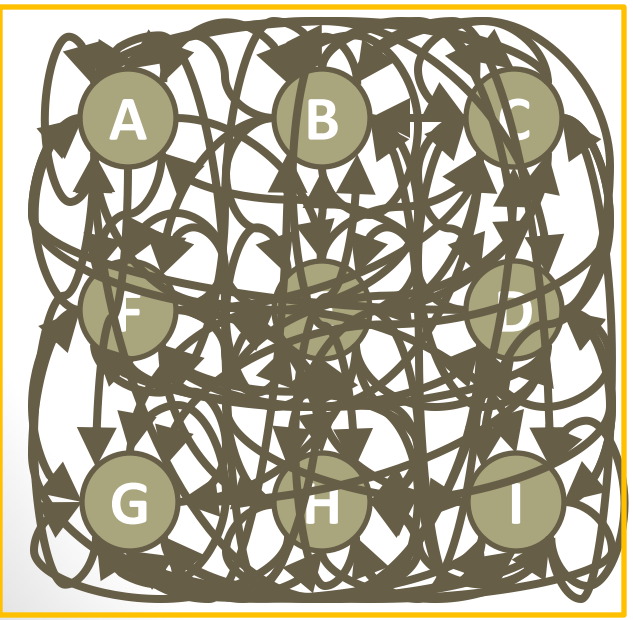
From	To
G	A
G	B
G	C
G	D
G	E
G	F
G	G
G	H
G	I
H	A
H	B
H	C
H	D
H	E
H	F
H	G
H	H
H	I

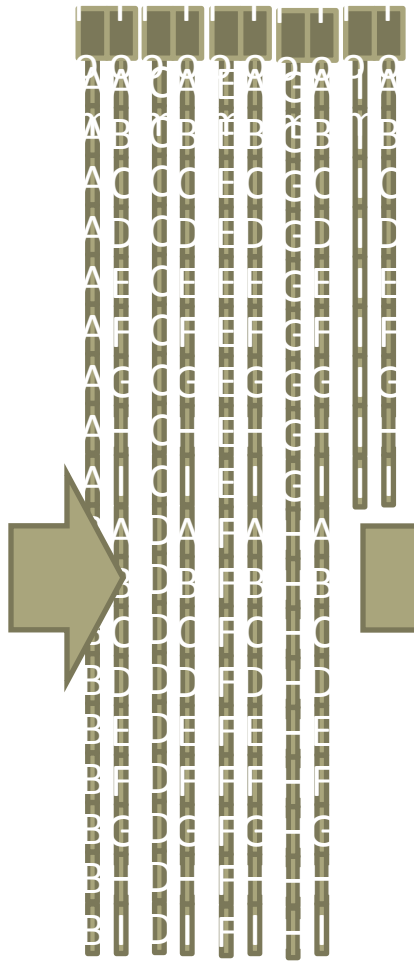
From	To
I	A
I	B
I	C
I	D
I	E
I	F
I	G
I	H
I	I

Graph Data: Connectedness and Density (19)



Graph Data: Connectedness and Density (20)

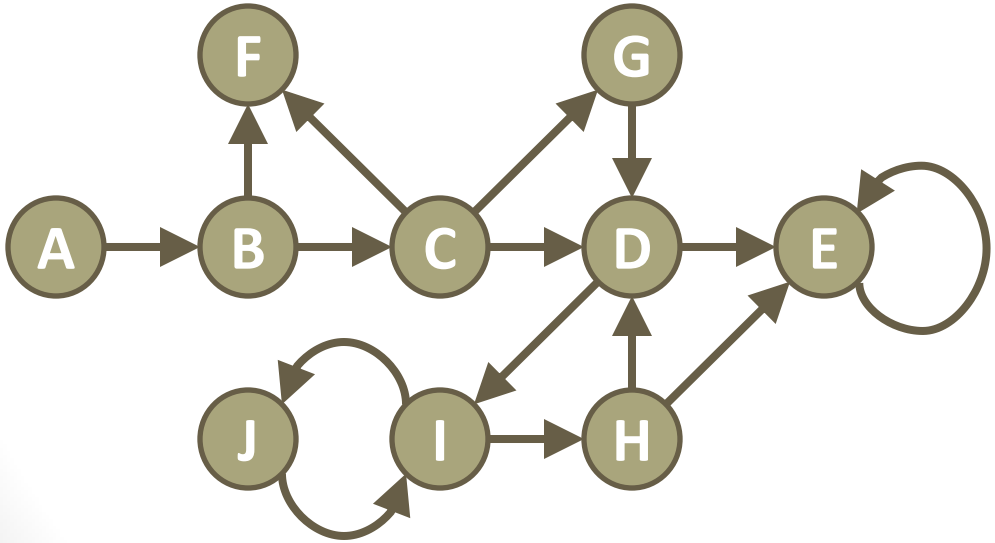




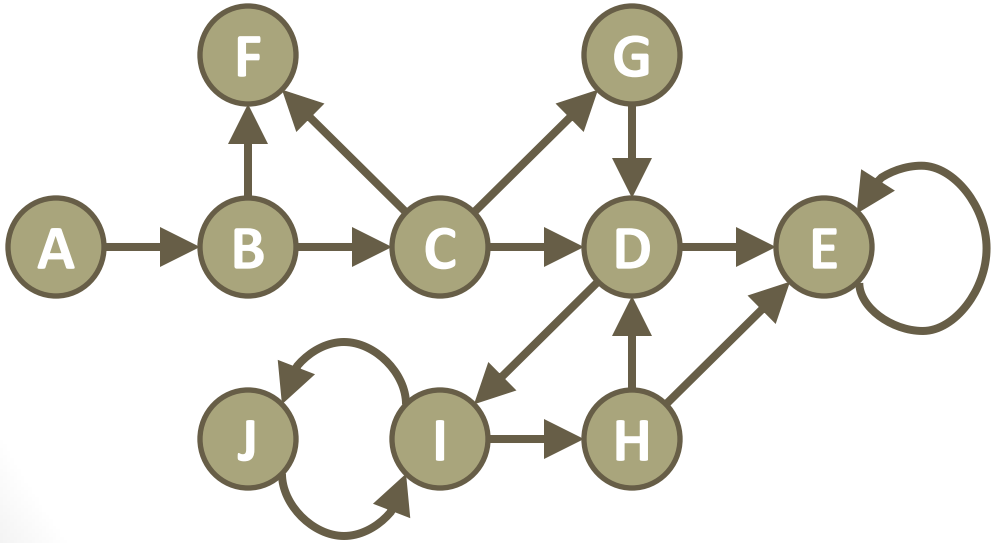
Dense: Mean row or column sum is close to number of nodes

To \ From	A	B	C	D	E	F	G	H	I	Sum:
A	1	1	1	1	1	1	1	1	1	9
B	1	1	1	1	1	1	1	1	1	9
C	1	1	1	1	1	1	1	1	1	9
D	1	1	1	1	1	1	1	1	1	9
E	1	1	1	1	1	1	1	1	1	9
F	1	1	1	1	1	1	1	1	1	9
G	1	1	1	1	1	1	1	1	1	9
H	1	1	1	1	1	1	1	1	1	9
I	1	1	1	1	1	1	1	1	1	9
Sum:	9	9	9	9	9	9	9	9	9	

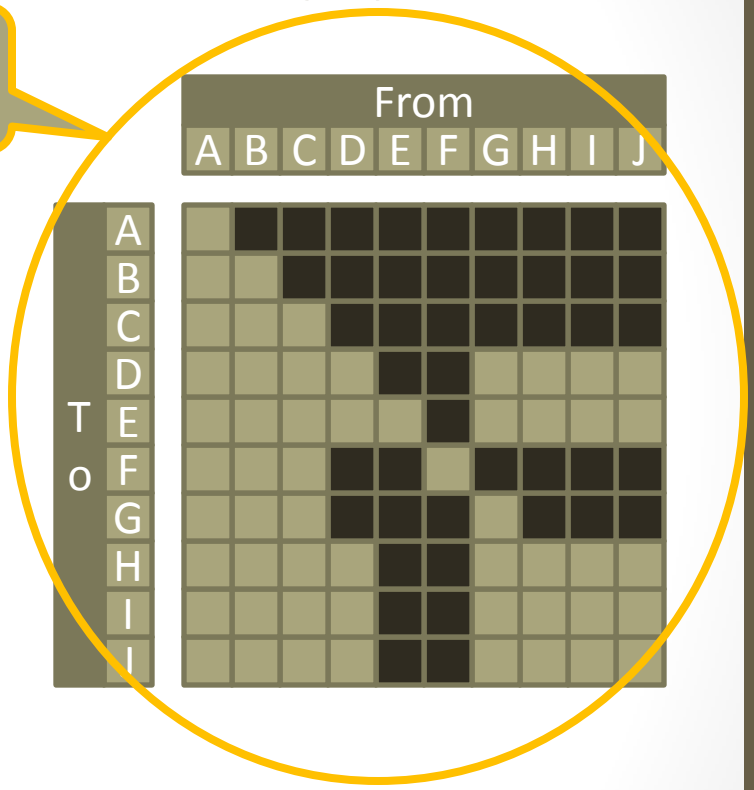
Graph Data: Allowed Paths, Distance, and Diameter (0)



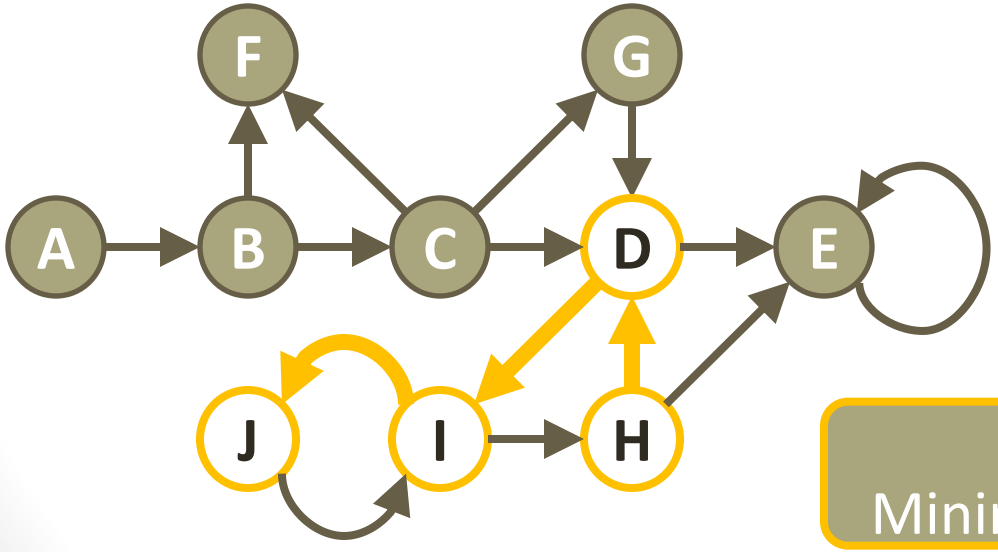
Graph Data: Allowed Paths, Distance, and Diameter (1)

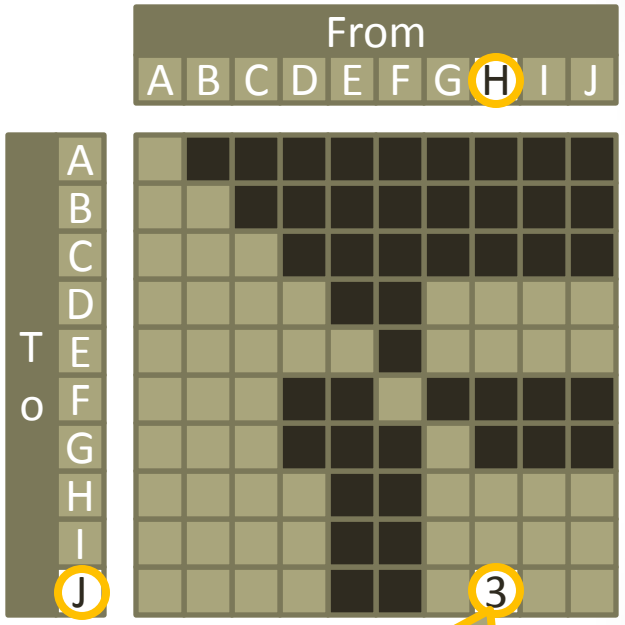


Allowed Paths



Graph Data: Allowed Paths, Distance, and Diameter (2)

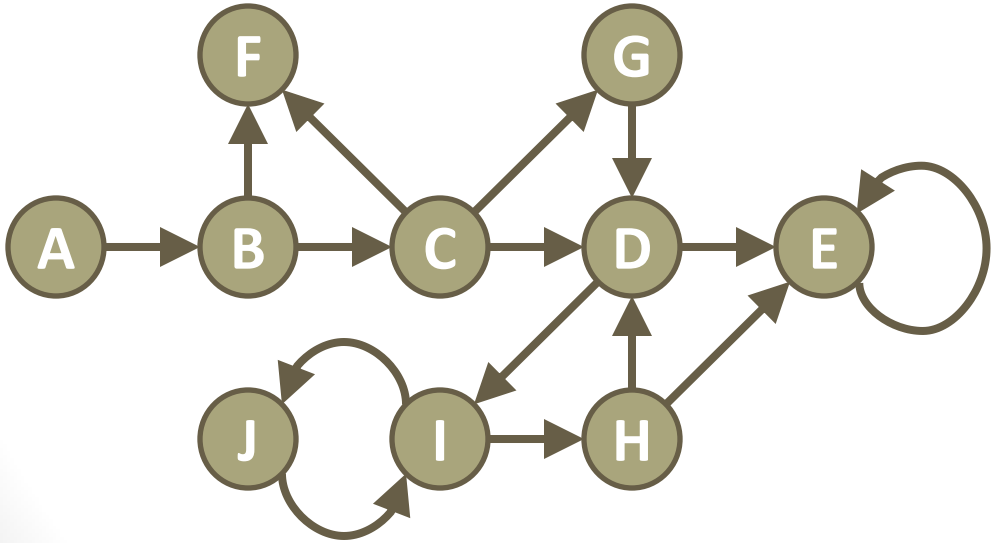




Path Distance:
Minimum Distance from H to J

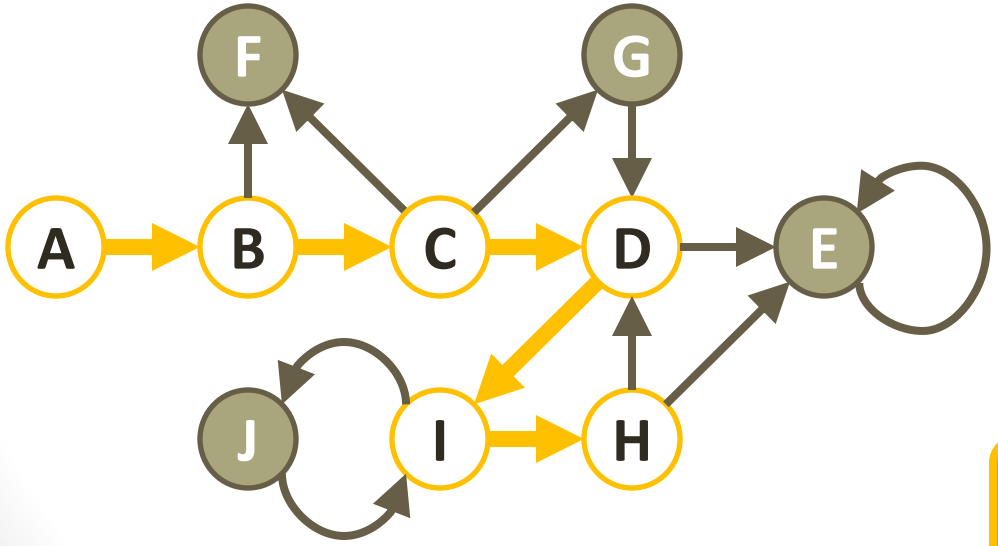
Graph Data: Allowed Paths, Distance, and Diameter (3)

Distance Matrix:
Minimum distance between any two vertices along a directed edge



To \ From	A	B	C	D	E	F	G	H	I	J
A	0									
B	1	0								
C	2	1	0							
D	3	2	1	0			1	1	2	3
E	4	3	2	1	0		2	1	2	3
F	2	1	1			0				
G	4	2	1				0			
H	5	4	3	2			3	0	1	2
I	4	3	2	1			2	2	0	1
J	5	4	3	2			3	3	1	0

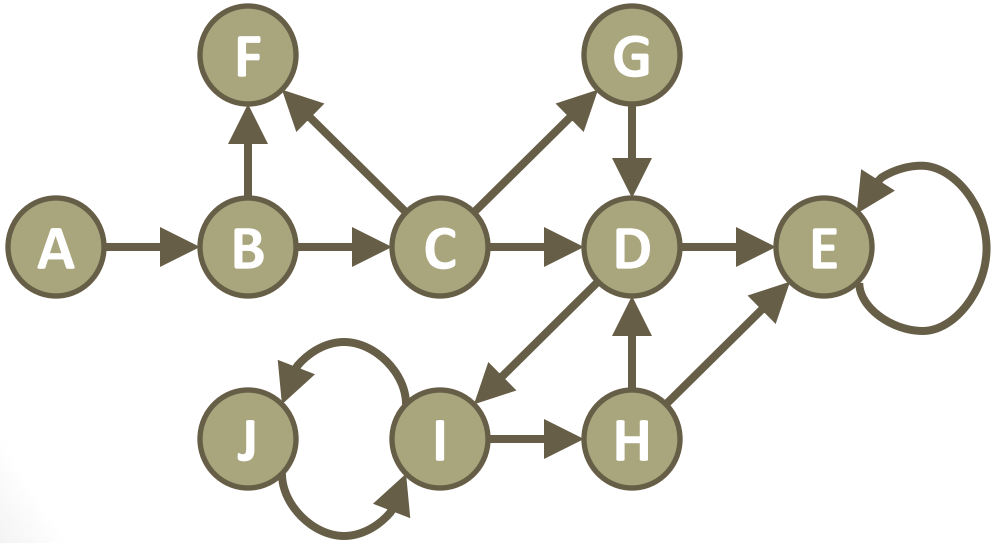
Graph Data: Allowed Paths, Distance, and Diameter (4)



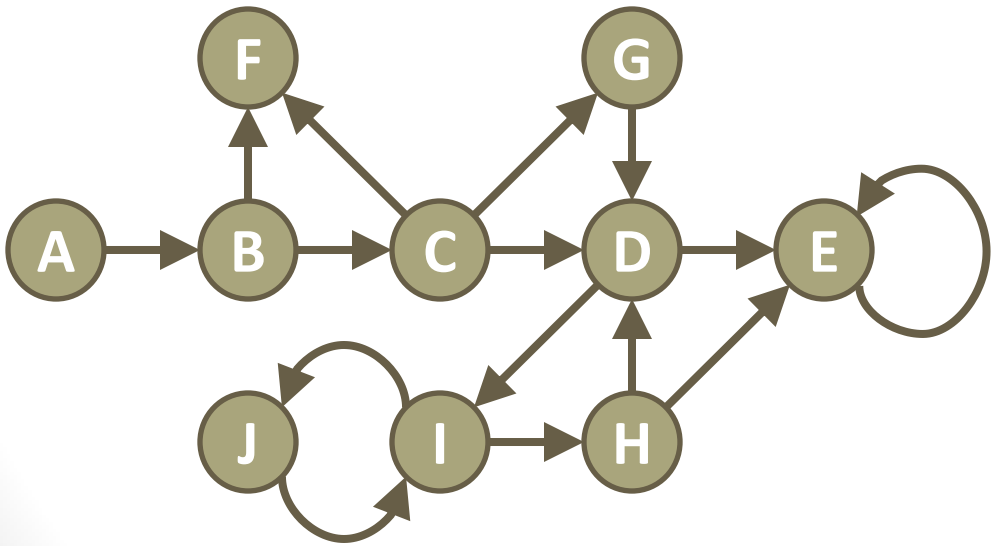
To \ From	A	B	C	D	E	F	G	H	I	J
A	0									
B	1	0								
C	2	1	0							
D	3	2	1	0			1	1	2	3
E	4	3	2	1	0		2	1	2	3
F	2	1	1			0				
G	4	2	1				0			
H	5	4	3	2			3	0	1	2
I	4	3	2	1			2	2	0	1
J	5	4	3	2			3	3	1	0

Diameter:
Max Path Distance

Graph Data: Order and Size (0)



Graph Data: Order and Size (1)

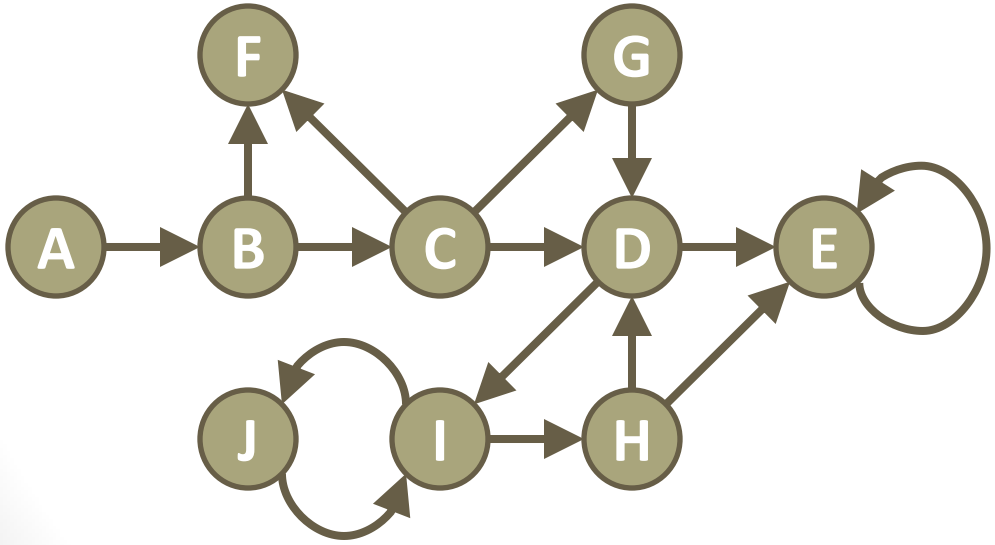


Matrix of Edges

To \ From	A	B	C	D	E	F	G	H	I	J
A										
B	1									
C		1								
D			1				1	1		
E				1	1			1		
F		1	1							
G			1							
H									1	
I				1						1
J									1	

Graph Data: Order and Size (2)

Order of Graph: 10

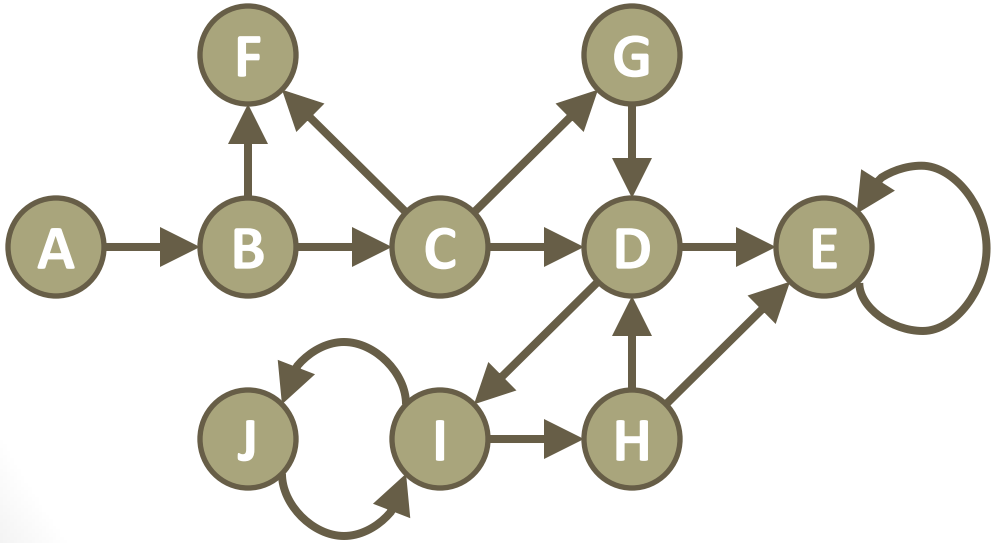


To \ From	A	B	C	D	E	F	G	H	I	J
A										
B	1									
C		1								
D			1				1	1		
E				1	1			1		
F		1	1							
G			1							
H									1	
I				1						1
J									1	

Order of Graph: Number of Nodes

Graph Data: Order and Size (3)

Order of Graph: 10
Size of Graph: 15



To \ From	A	B	C	D	E	F	G	H	I	J
A										
B	1									
C		1								
D			1				1	1		
E				1	1			1		
F		1	1							
G			1							
H									1	
I				1						1
J									1	

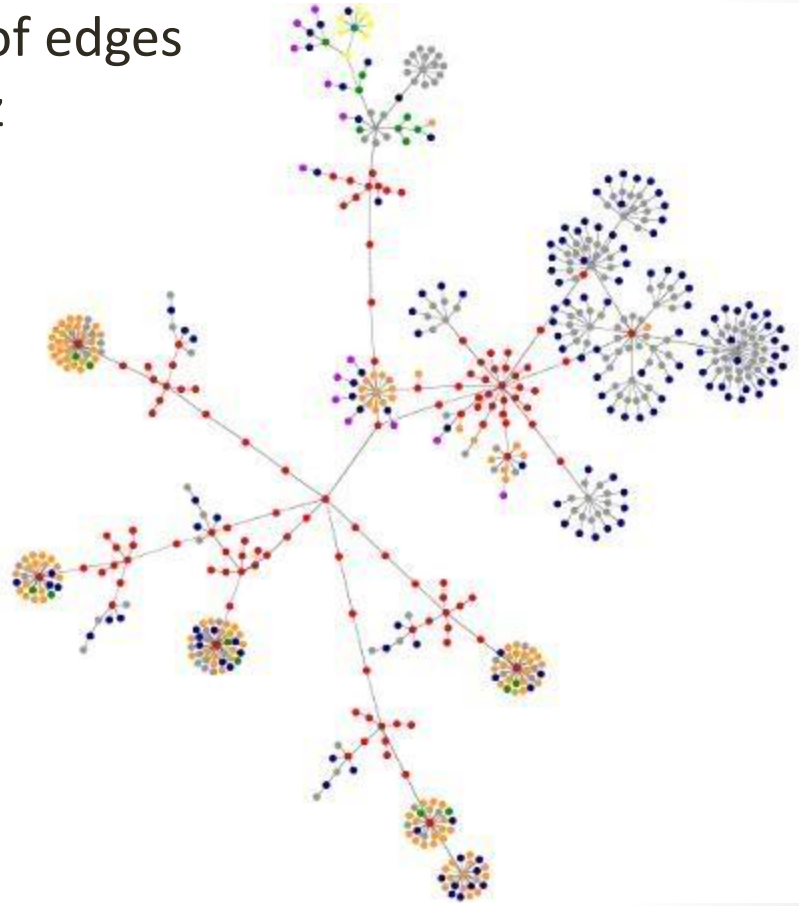
Size of Graph:
Number of Edges

Graph Data: Links (0)

- Graphs In General:
  - http://en.wikipedia.org/wiki/Graph_theory
  - https://en.wikipedia.org/wiki/Graph_(discrete_mathematics)
  - https://en.wikipedia.org/wiki/Graph_(abstract_data_type)
  - https://en.wikipedia.org/wiki/List_of_graph_theory_topics
- Graph Diameter
  - http://people.hofstra.edu/geotrans/eng/methods/diameter1.html
- Distance
  - http://en.wikipedia.org/wiki/Distance_(graph_theory)
- Resistance
  - http://en.wikipedia.org/wiki/Resistance_distance
- Centrality
  - http://en.wikipedia.org/wiki/Betweenness#Betweenness_centrality
- Connectivity
  - http://en.wikipedia.org/wiki/Connectivity_(graph_theory)
- Hypergraph: One edge can connect may nodes
  - http://en.wikipedia.org/wiki/Hypergraph
- Degree Distribution: Histogram of in-degrees, out-degrees, in-degrees normalized by out-degrees
- Popularity (Google Matrix)
- Density: #edges/(#nodes^2-#nodes)
- Clustering Coefficient
  - http://en.wikipedia.org/wiki/Clustering_coefficient

Quiz Graph Data (Measures)

- The presented slide contains a list of edges that describes the graph in the quiz



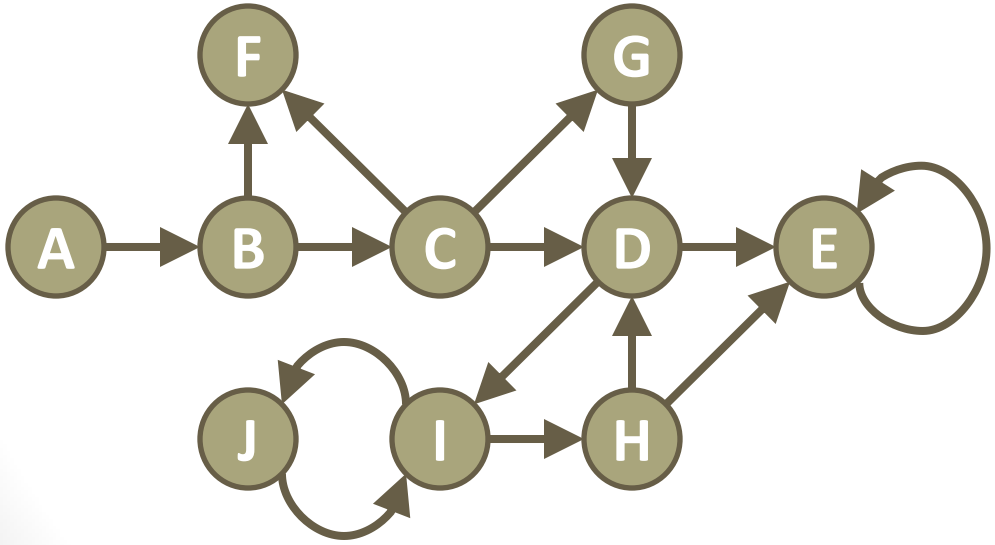
Introduction to Graph Data

Graph Data: Google Matrix

Google Matrix

- http://en.wikipedia.org/wiki/Google_matrix

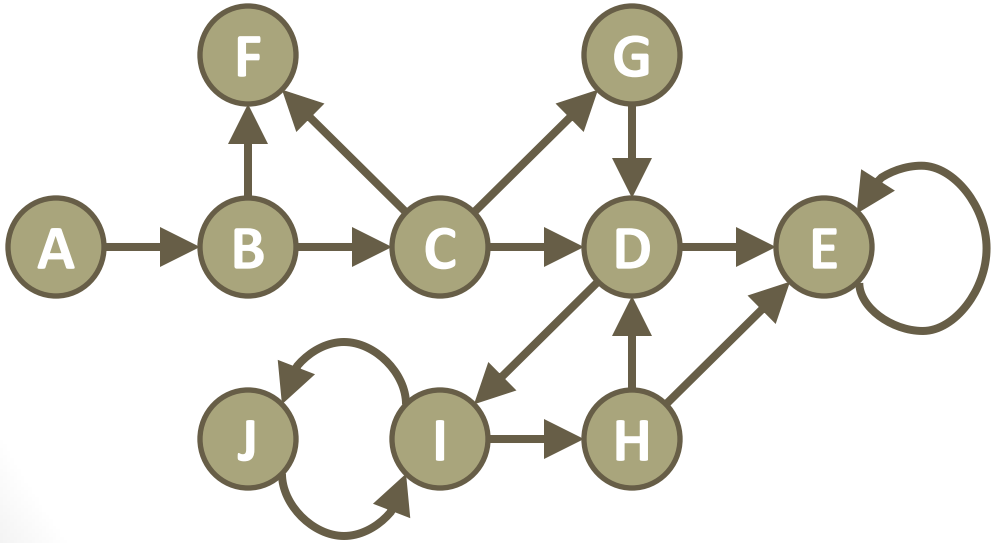
Graph Data: Popularity (0)



To \ From	A	B	C	D	E	F	G	H	I	J
A										
B	1									
C		1								
D			1				1	1		
E				1	1			1		
F		1	1							
G			1							
H									1	
I				1						1
J									1	

Graph Data: Popularity (1)

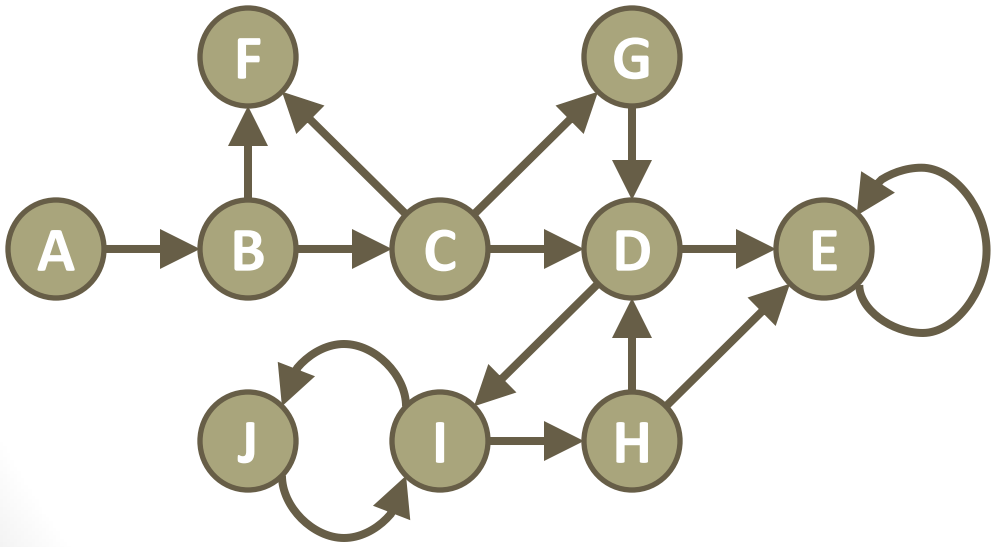
In-degree of a node: Popularity



To \ From	A	B	C	D	E	F	G	H	I	J	Sum:
A											0
B	1										1
C		1									1
D			1				1	1			3
E				1	1			1			3
F		1	1								2
G			1								1
H									1		1
I				1						1	2
J									1		1

Graph Data: Popularity (2)

In-degree of a node: Popularity



To \ From	A	B	C	D	E	F	G	H	I	J	Sum:
A											0
B	1										1
C		1									1
D			1				1	1			3
E				1	1			1			3
F		1	1								2
G			1								1
H									1		1
I				1						1	2
J									1		1
Sum:	1	2	3	2	1	0	1	2	2	1	

Out-degree of a node: Gregariousness

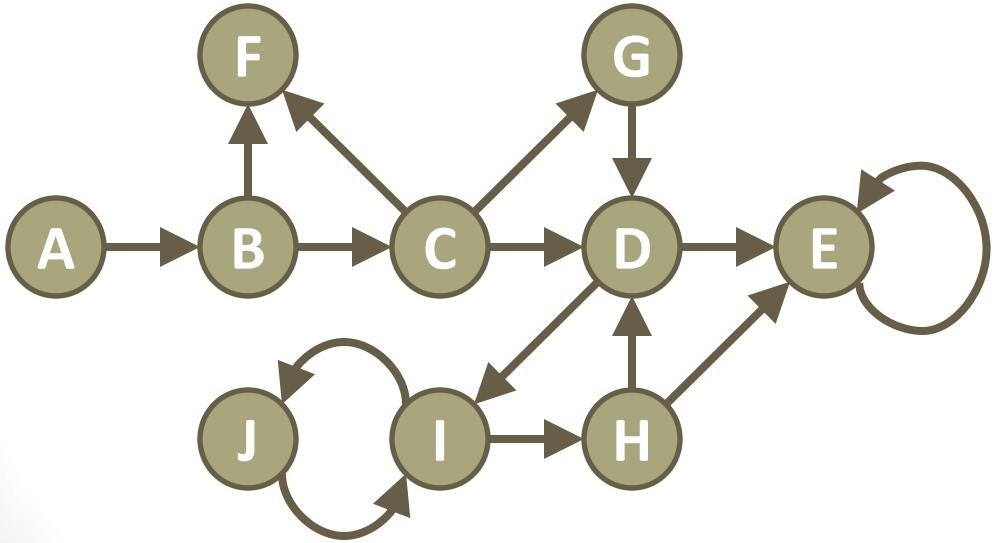
Quiz Graph Data (Popularity)

- Graph Data (Popularity)

- You need to view the projected slide to answer the questions

Graph Data: Popularity (3)

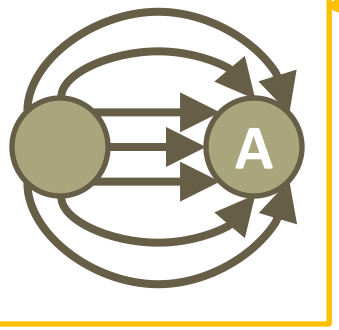
But, not every link is equally important. Otherwise, you could create a website with a thousand links that all point to your site.

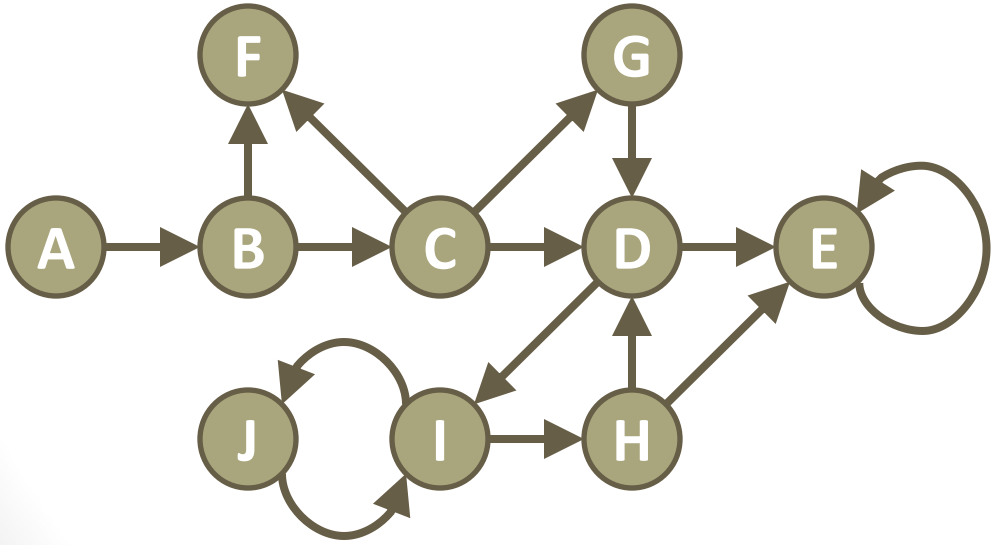


To \ From	A	B	C	D	E	F	G	H	I	J	Sum:
A											0
B	1										1
C		1									1
D			1				1	1			3
E				1	1			1			3
F		1	1								2
G			1								1
H									1		1
I				1						1	2
J									1		1
Sum:	1	2	3	2	1	0	1	2	2	1	

Graph Data: Popularity (4)

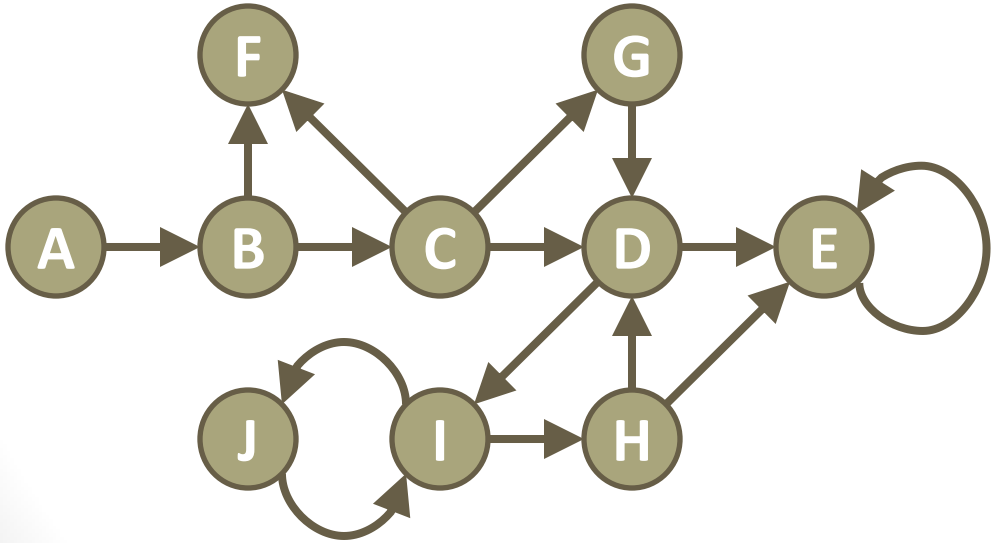
One very gregarious node could make another node very popular





To \ From	A	B	C	D	E	F	G	H	I	J	Sum:
A											0
B	1										1
C		1									1
D			1				1	1			3
E				1	1			1			3
F		1	1								2
G			1								1
H									1		1
I				1						1	2
J									1		1
Sum:	1	2	3	2	1	0	1	2	2	1	

Graph Data: Popularity (5)

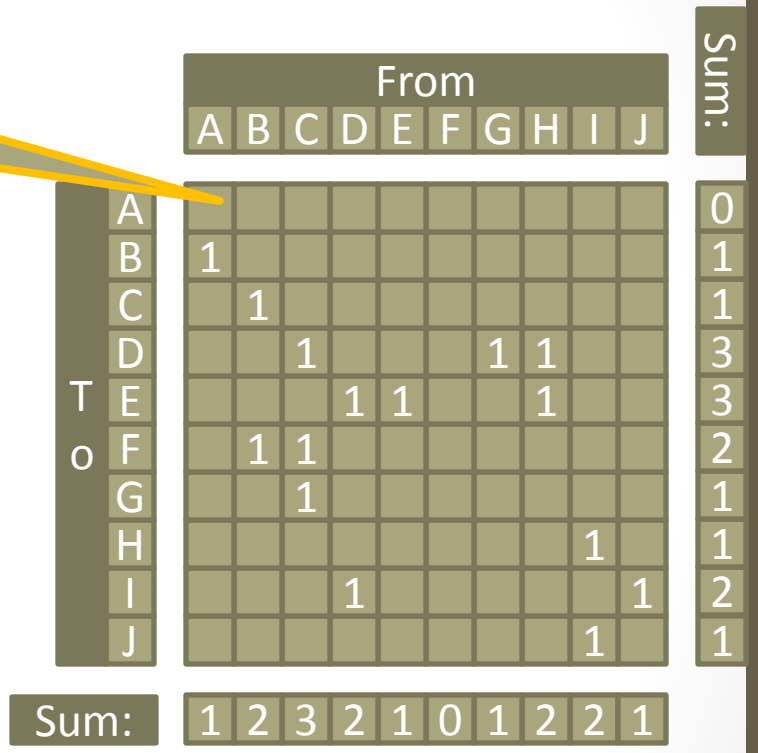


To \ From	A	B	C	D	E	F	G	H	I	J	Sum:
A											0
B	1										1
C		1									1
D			1				1	1			3
E				1	1			1			3
F		1	1								2
G			1								1
H									1		1
I				1						1	2
J									1		1
Sum:	1	2	3	2	1	0	1	2	2	1	

The amount of popularity that a single node can send out needs to be tempered by Gregariousness

Graph Data: Popularity (6)

This matrix needs some adjustments



Graph Data: Popularity (7)

To \ From	A	B	C	D	E	F	G	H	I	J	sum:
A											0
B	1										1
C		1									1
D			1				1	1			3
E				1	1			1			3
F		1	1								2
G			1								1
H									1		1
I				1						1	2
J									1		1
sum:	1	2	3	2	1	0	1	2	2	1	

Graph Data: Popularity (8)

To \ From	A	B	C	D	E	F	G	H	I	J	sum:
A											0
B	1										1.00
C		1/2									0.50
D			1/3				1	1/2			1.83
E				1/2	1			1/2			2.00
F		1/2	1/3								0.83
G			1/3								0.33
H									1/2		0.50
I				1/2						1	1.50
J									1/2		0.50
sum:	1	1	1	1	1	0	1	1	1	1	

Graph Data: Popularity (9)

To \ From	A	B	C	D	E	F	G	H	I	J	sum:	Popularity:
A											0	0
B	1										1.00	0.111
C		1/2									0.50	0.056
D			1/3				1	1/2			1.83	0.204
E				1/2	1			1/2			2.00	0.222
F		1/2	1/3								0.83	0.092
G			1/3								0.33	0.037
H									1/2		0.50	0.056
I				1/2						1	1.50	0.167
J									1/2		0.50	0.056
sum:	1	1	1	1	1	0	1	1	1	1		

Graph Data: Popularity (10)

- Rules of a Popularity Network
  - Rule 0: Popularity transmits from one node to a downstream connected node.
  - Rule 1: Popularity transmissions from multiple nodes to a single node are additive.
  - Rule 2: A more popular node transmits more popularity.
  - Rule 3: A node splits up its popularity among its recipients.

- Continue with **EigenVectorOfGraphMatrix.R**

- Solve (or Iterate until convergence)
  - Modified Popularity is the Popularity divided by Gregariousness (Out-Degree)
  - Popularity of a website is the sum of the Modified Popularities that point to the website.

Graph Data: Google Matrix

Assignment

Assignment

1. Comment in our LinkedIn Group on a discussion started by yourself or a fellow student about statistics. Copy the post to a text file called post.txt. This assignment item is similar to last week's assignment.
2. Create a MapReduce program that lists word lengths next to the number of times a word of that length occurs. The word lengths are sorted as texts, even though they are numbers. Specifically, modify package wordcount.solution as described in the MapReduce Lab for "Word Length Count 1". In this lab, the output keys are of type text. Save all code of the modified java file(s) and a screen shot of all the results (approx. 18 rows).
3. Create a MapReduce program that lists word lengths next to the number of times a word of that length occurs. The word lengths are sorted numerically. Specifically, modify package wordcount.solution as described in the MapReduce Lab for "Word Length Count 2". In this lab, the output keys are of type IntWritable (numeric). Save all code of the modified java file(s) and a screen shot of all the results (approx. 18 rows).
4. Submit to Canvas by Saturday 11:57 PM the following: post.txt. The code as *.java files from "Word Length Count 1". The screenshot of the results from "Word Length Count 1". The code as *.java files from "Word Length Count 2". The screenshot of the results from "Word Length Count 2".
5. Look through the Preview section, especially the SPARQL Exercises, prior to next week's lecture. (There will be a quiz).

Assignment

Introduction to Data Science