빅데이터 분석 및 응용

L04: Graph Mining on MapReduce

Summer 2020

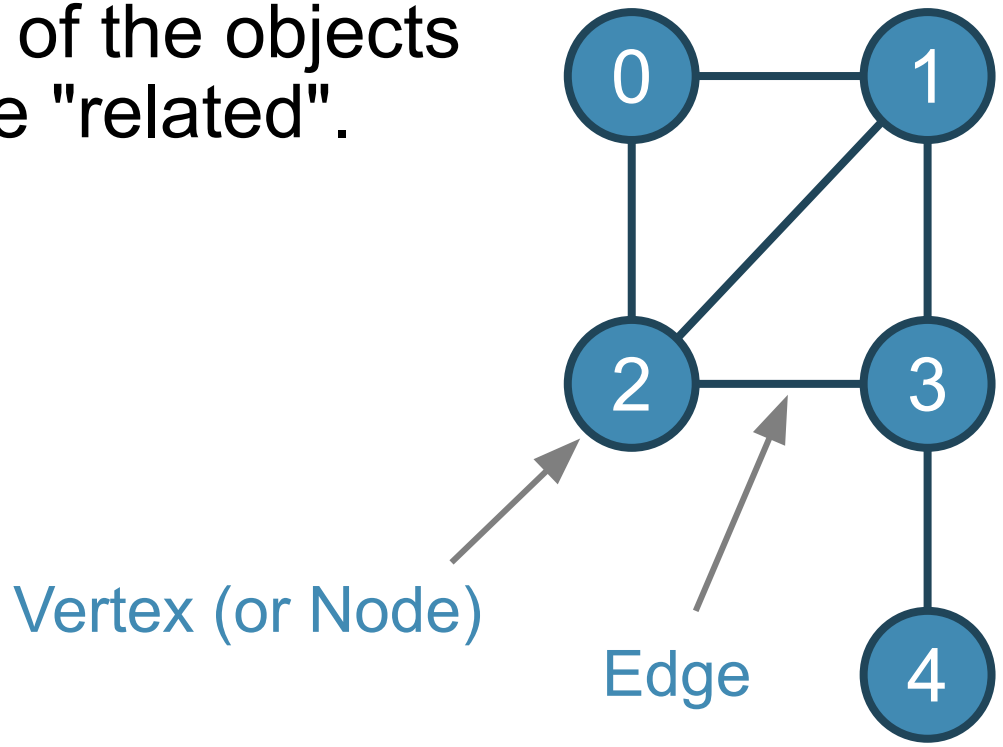
Kookmin University

Contents

- Graph
- Graph Mining
- A Graph Mining Method: Triangle Listing
- Triangle Listing on MapReduce

Graphs

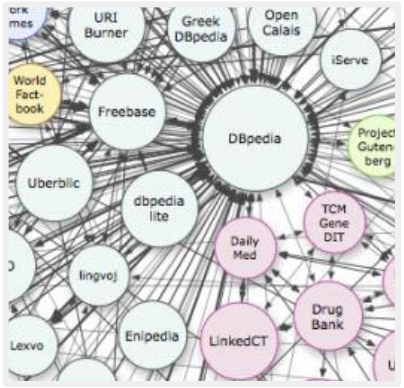
- A **graph** is a structure amounting to a set of objects in which some pairs of the objects are in some sense "related".
- G = (V, E)
  - G: a graph
  - V: a set of vertices
  - E: a set of edges



Graphs are everywhere

Friendship Network

Phonecall Network



Knowledge Base

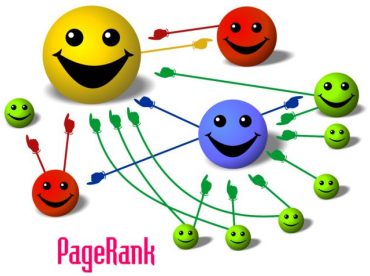
Internet, WWW

- Co-authorship network of papers
- Computer network
- Protein-protein interaction network
- etc...

Graph mining

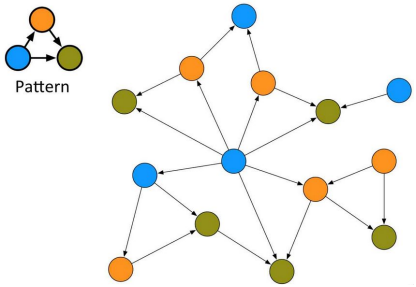
Structural Analyses

PageRank
RWR
Radius/Diameter
Degree
SSSP
Label Propagation
Connected Components



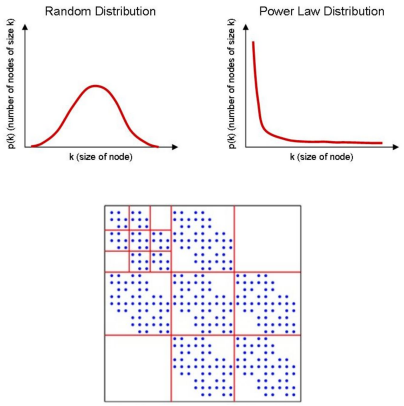
Subgraph Enumeration (Graph Pattern Matching)

Triangle
Clique
Other Pattern Graphs

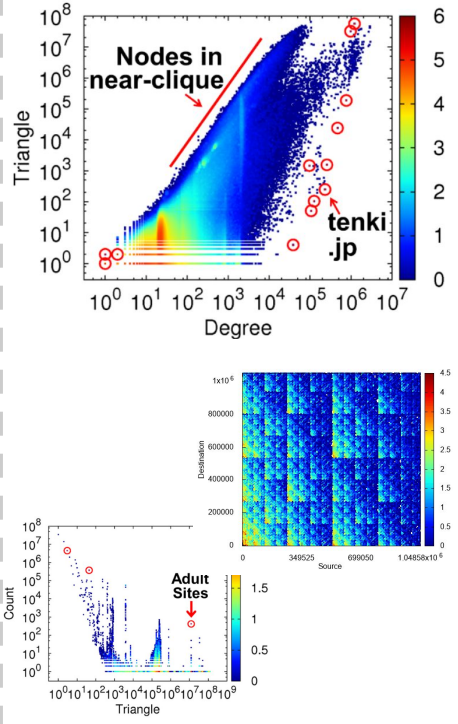


Graph Modeling (Graph Generation)

Kronecker
R-Mat
Random

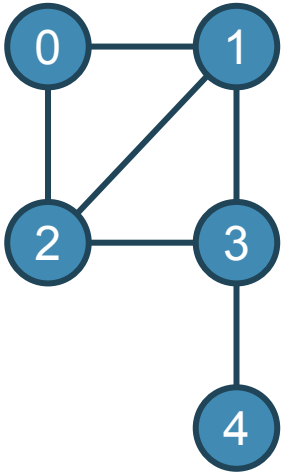


Graph Visualization



File formats for graphs

- Two representative file formats for sparse undirected unweighted graphs
  - Edge list file format
  - Adjacency list file format



edge_list.txt

```
0 1
0 2
1 2
1 3
2 3
3 4
```

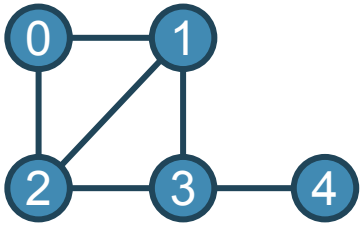
adj_list.txt

```
0 1 2
1 0 2 3
2 0 1 3
3 1 2 4
4 3
```

Problem definition: Triangle Listing

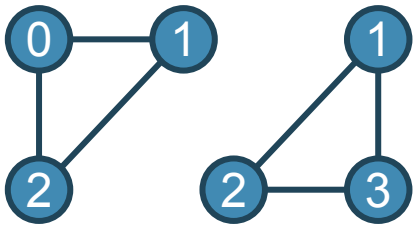
Triangle Listing

Given a simple graph G=(V, E), **list all triangles** in G.



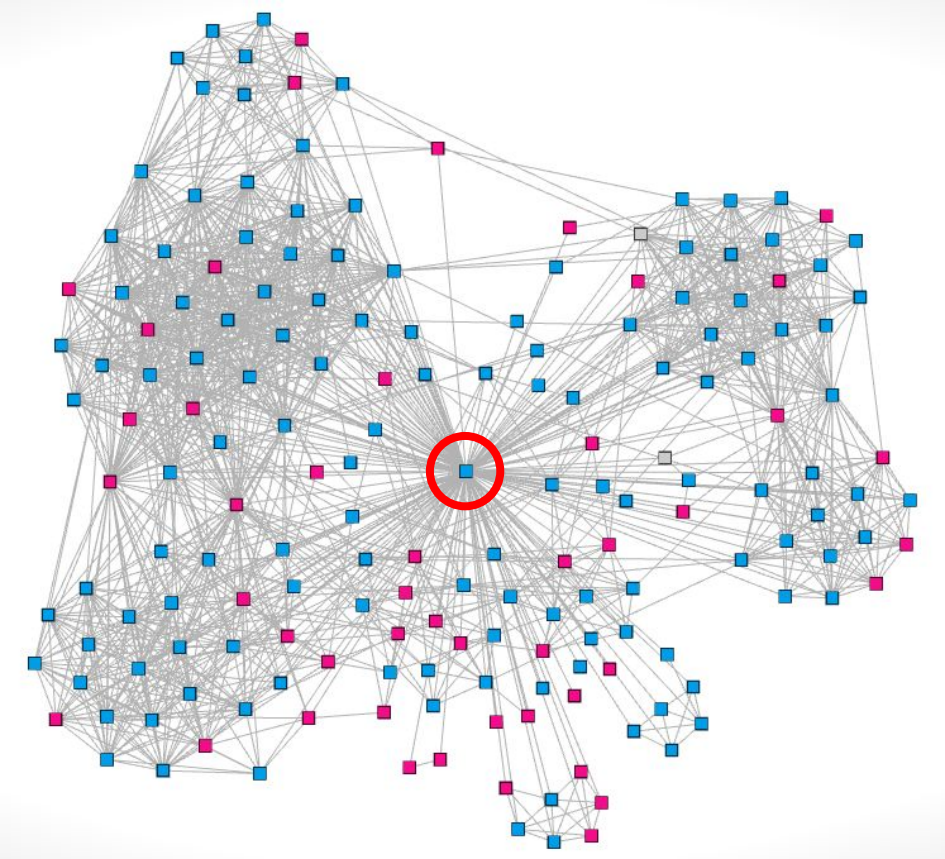
0	1
0	2
1	2
1	3
2	3
3	4

Triangle Listing

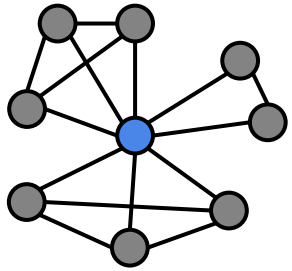


0	1	2
1	2	3

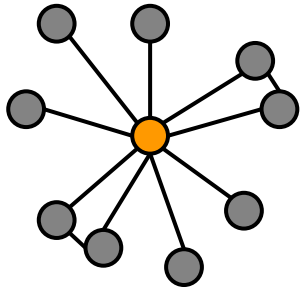
Application1. spam/fake user detection



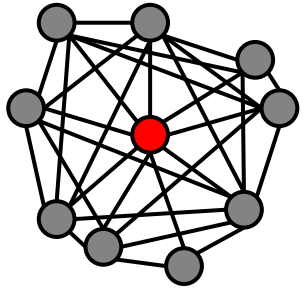
An egonet of a user in Facebook.
The user seems to be normal user.



Normal User with Many Triangles

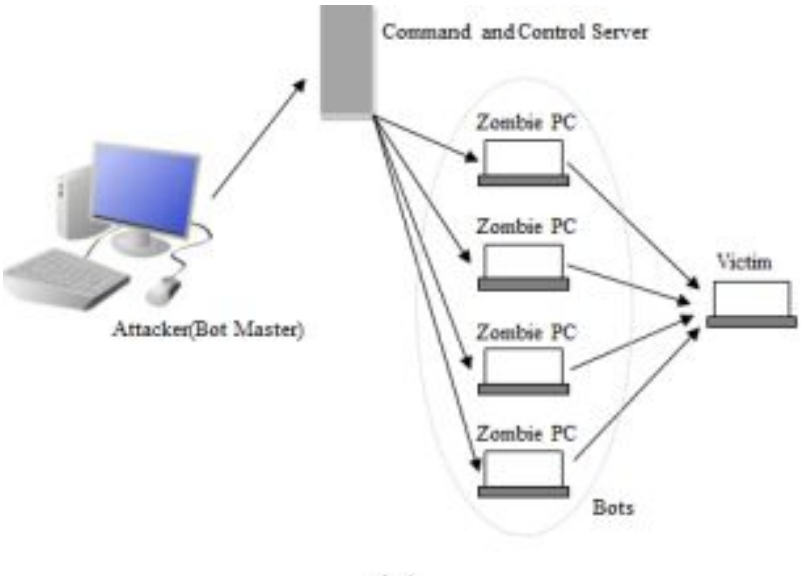


Spammer with Few Triangles

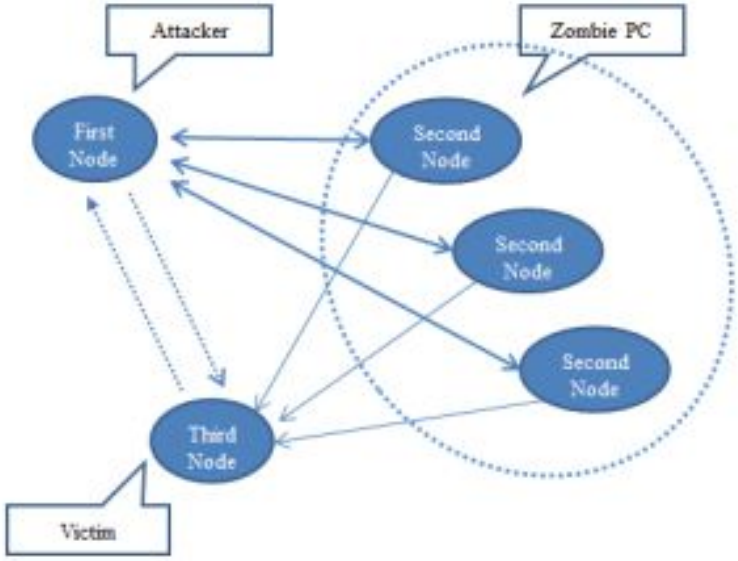


Fake User with Massive Triangles

Application2. DDoS attack detection



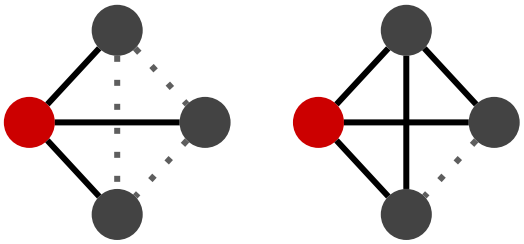
Botnet Attack



Triangle Expectation of relationship

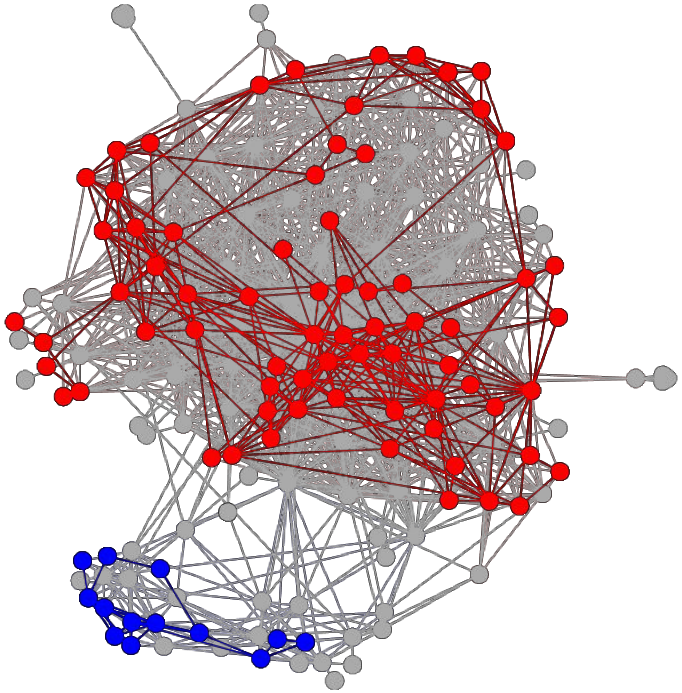
Application3. Community detection

Clustering coefficient = # triangles / # possible triangles



0/3 2/3

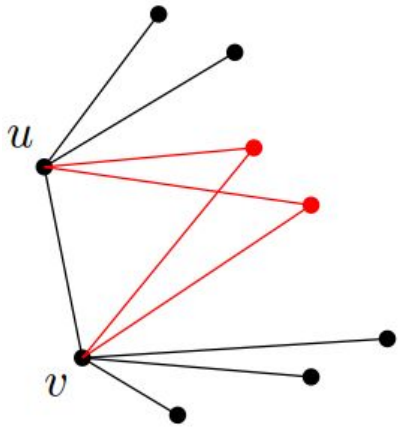
Clustering Coefficient Example



A single machine algorithm for triangle listing

- Load a graph as an adjacency list format
- List all triangles

For each edge (u, v) in E
    For each n in $N(u) \cap N(v)$
        output $\Delta(u, v, n)$



Intersection of two neighbor sets
$\rightarrow$ triangles

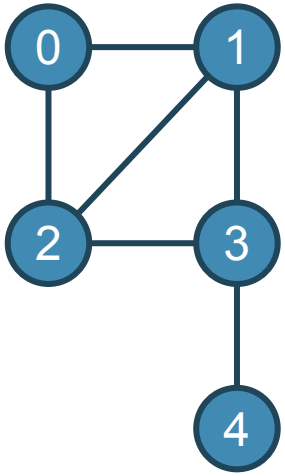
Graphs are **ENORMOUS**

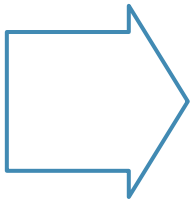
1 billion active users in **facebook**
1 trillion pages on **the Web**

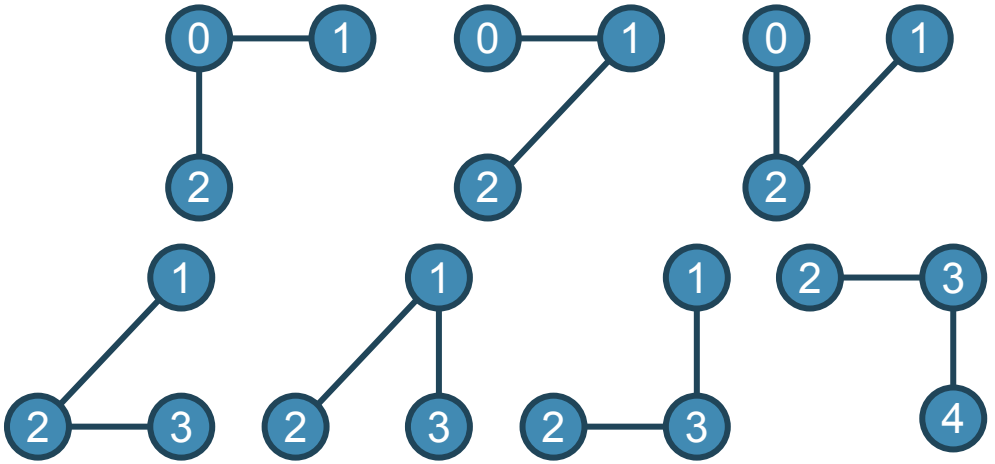
Q. How can we handle such large graphs?
A. Let's use MapReduce!

Triangle Listing on MapReduce

- Step 1. find every wedge

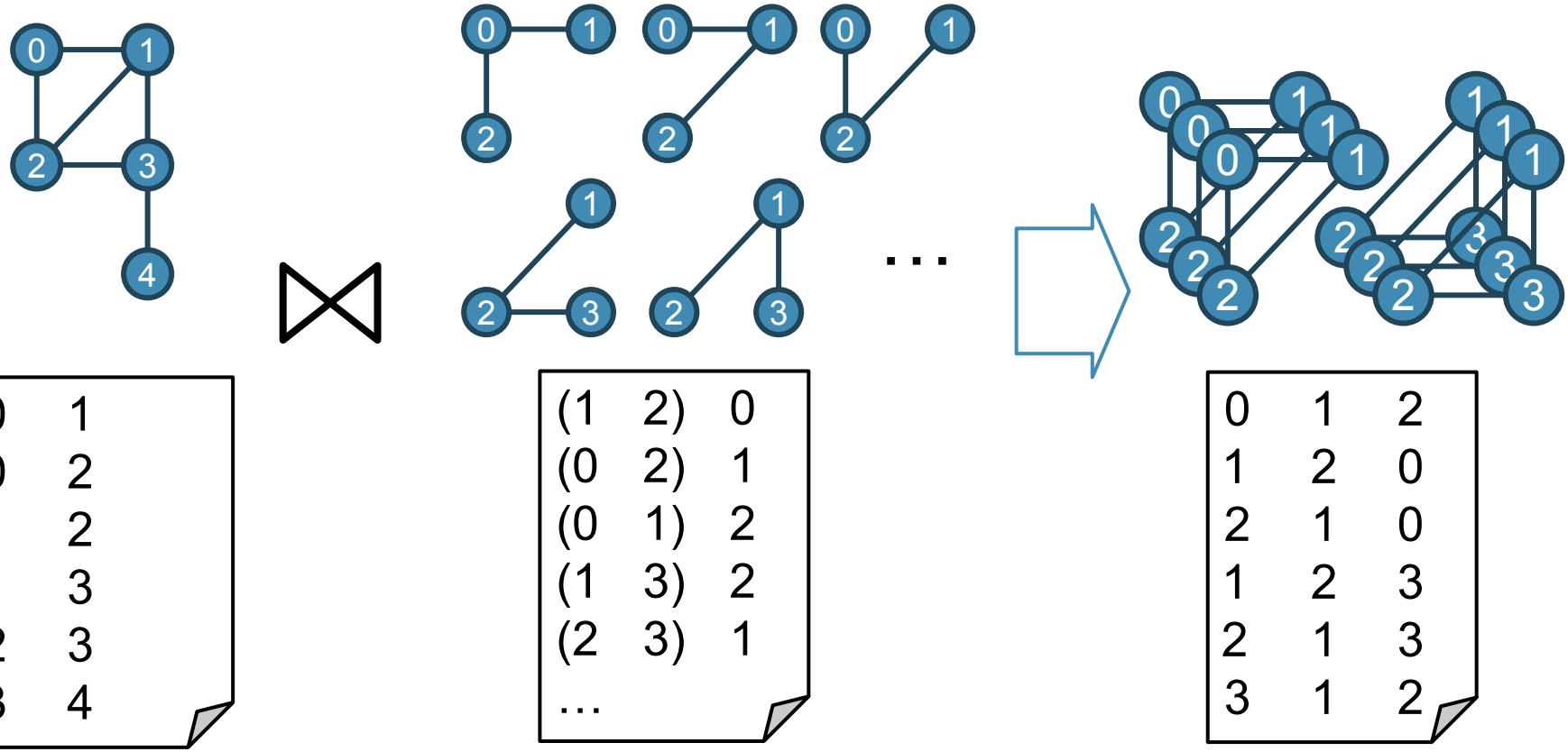






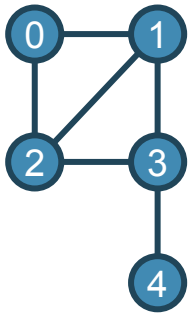
Triangle Listing on MapReduce

- Step 2. join edges and wedges

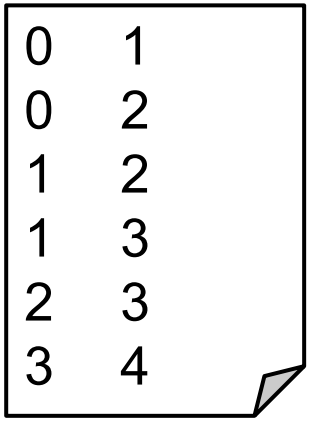


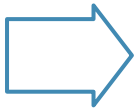
Triangle Listing on MapReduce

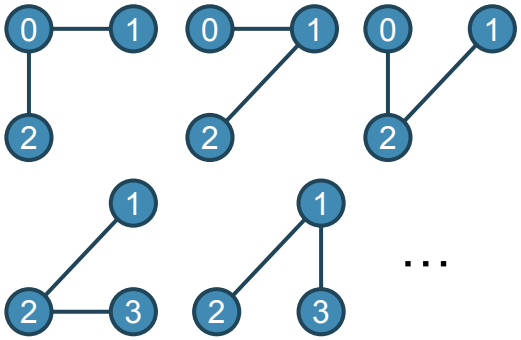
- Step 1. find every wedge

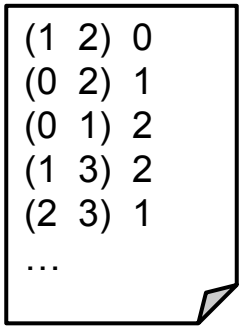



```
map(key, value):
    emit(key, value)
    emit(value, key)

reduce(key, values):
    for each value v1 in values:
        for each value v2 in values:
            if(v1 < v2):
                emit((v1, v2), key)
```









Triangle Listing on MapReduce

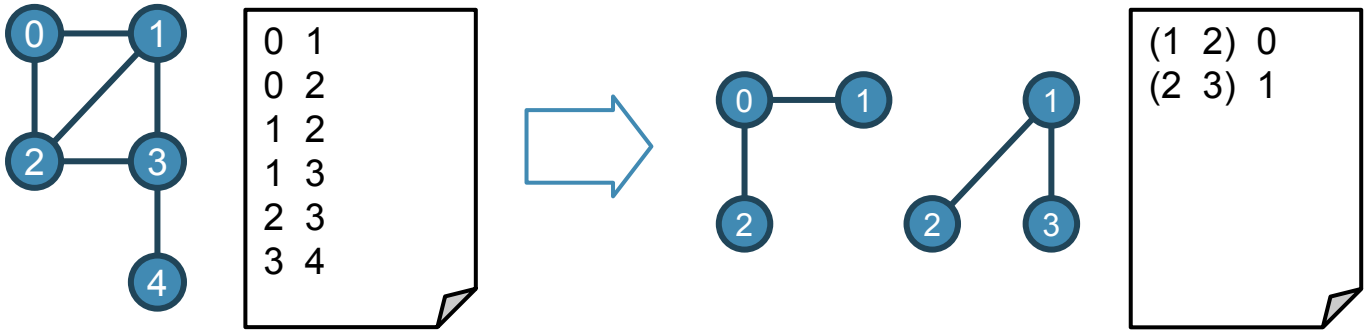
- Step 2. join edges and wedges

```
map(key, value):
    if the input is an edge:
        emit((key, value), $)
    else:
        // the input is a wedge
        // key is a vertex pair, value is a vertex
        emit(key, value)

reduce(key, values):
    // key is a vertex pair
    if values contains $:
        for each vertex v in values except $:
            emit(key, v)
```

Refinement: Remove Duplicates

- Modify the map function of step 1.

```
map(key, value):
    emit(key, value)
    emit(value, key)

reduce(key, values):
    for each value v1 in values:
        for each value v2 in values:
            if(v1 < v2):
                emit((v1, v2), key)
```



Questions?