2018-2

PA03 Report

Team01

21400646 Lim Chae Eon 21500273 Park Su Hyun 21700130 Kim Yeagoon
21700332 Bae Junhyeon

https://github.com/PASTANERD/DM_

Introduction

- Our solution scope

❑ In choosing our solution, we discussed between two different approaches; matrices and graphs

Matrix	Graph
Easy to program and understand	Difficult to make and understand
May take huge space resources.	Needs only vertices and edges,
It takes V^2 spaces.	It takes V*E spaces
Transitive calculation takes $O(n^4)$	Transitive calculation takes O(VE)

$$\begin{pmatrix} a_{11} & \dots & a_{1n} \\ \cdot & & \\ \cdot & & \\ \cdot & & \\ a_{n1} & & a_{nn} \end{pmatrix}$$

Figure 1. Example of matrix

$$\begin{tikzcd} A \arrow[rr] \arrow[dr] & & B \arrow[dl] \\ & C & \end{tikzcd}$$

Figure 2. Example of graph

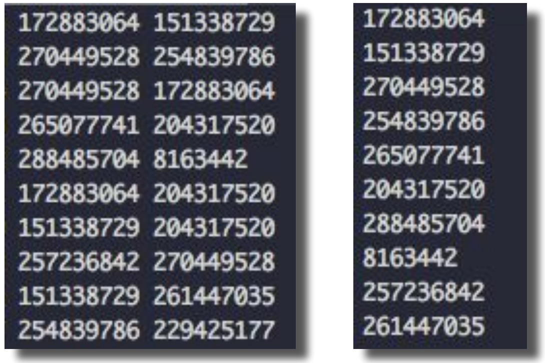
❑ We decided to use graphs for our problem.

Although we thought that the given sample data was small enough for us to use matrices, if it was a bigger dataset (such as the original data set), it was highly likely that we would not have been able approach it with this method.

Since our purpose is to make a good program, we chose to use the graph method in order to address the problem for other cases that may involve bigger datasets.

Introduction

Point for making graph



```
172883064 151338729
270449528 254839786
270449528 172883064
265077741 204317520
288485704 8163442
172883064 204317520
151338729 204317520
257236842 270449528
151338729 261447035
254839786 229425177
```

```
172883064
151338729
270449528
254839786
265077741
204317520
288485704
8163442
257236842
261447035
```

Figure 3. edges(left) and nodes(right)

- ❑ Graph needs vertices and edges
  - We extract vertices from the edges dataset. (Figure 3, right)
- ❑ We defined graph terms

Vertex :	User / Node
Edge :	Edge

- ❑ Also designed the program structure (Figure 4)

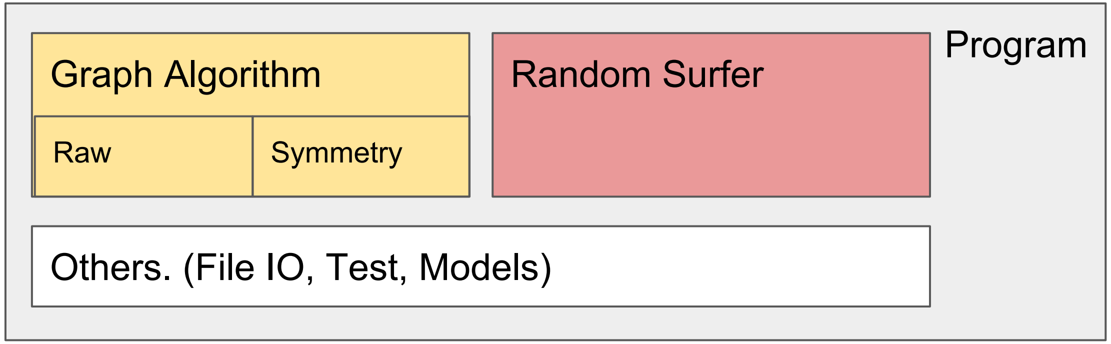


Figure 4. Draft design of program

❑ Graph algorithm

- Analyzing raw or symmetric edges.

<u>With raw edges</u>	<u>With symmetry edges</u>
Analyze all transitive relationships	Analyze all symmetric relationships
Find connectivity and longest minimum distance	Find sets of friend partitions

❑ Random Surfer

- Analyzing the user who is frequently visited.

❑ Others.

- Set up the data all nodes and edges into memory
- Do some iterative tasks like finding the same node id in the set of edges
- Produce result as a file.
- Etc.

Setup for problems 02~05

Our strategy is to make a model which analyzes all kinds of relations in this graph needed for problems from 02 to 05 into a single file.Then, for each problem, we can analyse that file for each problem, as needed.

1. First, make a set of unique users
2. Pick the first node from the unique set of users as a starting point and enqueue it.
3. When deque, give the *Weight*(distance) from the starting node by adding one to *Weight*, and store in *edge*.
4. Every time when the node comes out, enqueue each and every nodes that the previous node was following.
5. Whenever nodes are enqueued, check *Visit* to avoid revisiting.
6. If there is no queue, then go back to (1.) and start analyzing with a different user as a starting point.
7. At the end of this whole process, we have all information of each and every nodes and edges in a file.
8. **Partition* and *Influence* will be defined after the file is printed out.

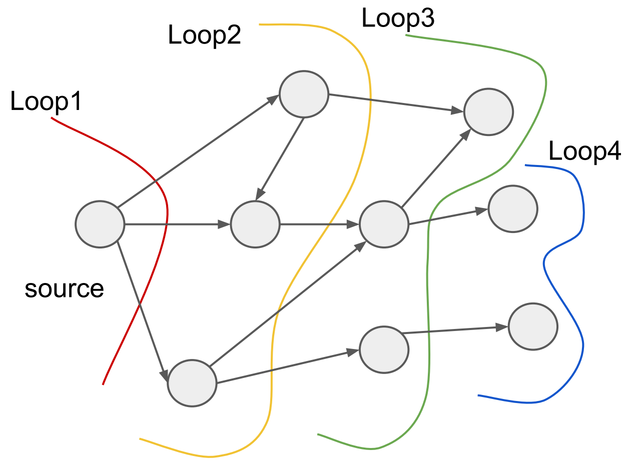


Figure 5. Graph searching by loop

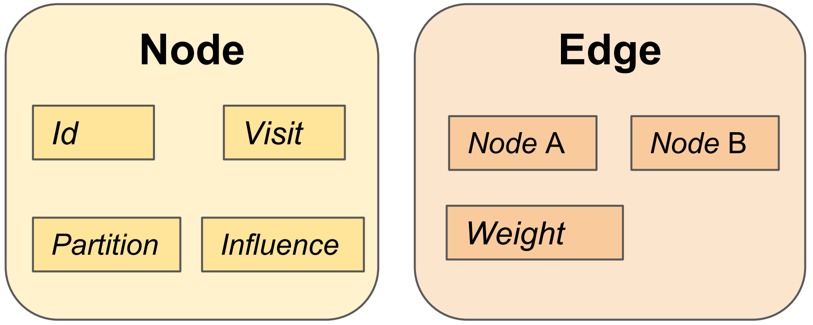


Figure 6. Structure in node and edge

PA01

Problem: Find the number of followers a user has, and the number of people the user follows. (Since the user is not predefined, the user will be defined when his/her id is typed in.)

From the given twitter sample list (without repetition), take A as the node that follows node B. If the user id (vertice) 123 is typed in, go through the list of As where A equals 123 in order to find B, the person whom 123 follows. Go through the list of Bs where B equals 123 in order to find A, 123's follower. Output the number of As and Bs as results.

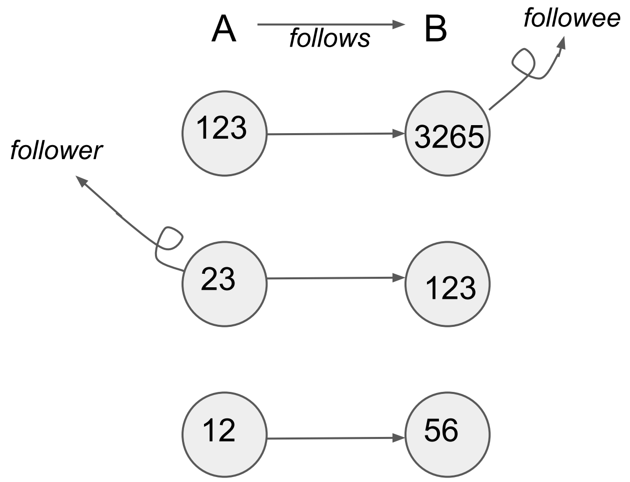


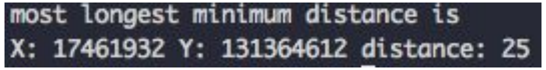
Figure 7. Structure of edge, follower and followee

PA 02

Problem: Find A and B where the minimum transitional leap (weight) from A to B is greatest.

From the file of every nodes and edges built in *Setup for problems 02~05*, find the edge with the greatest weight value and return nodes A and B of that edge.

Result :

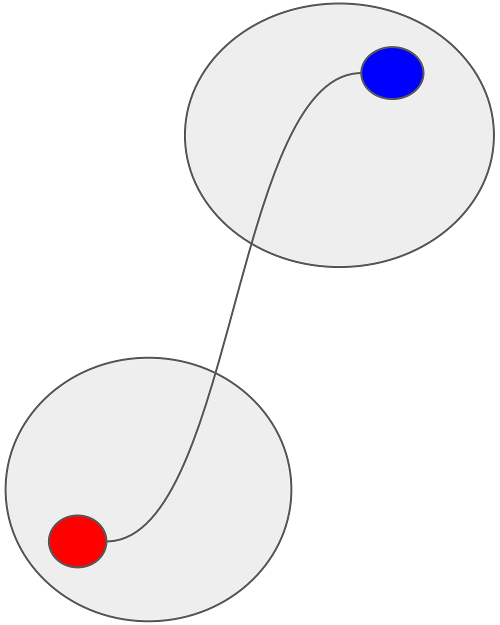


```
most longest minimum distance is
X: 17461932 Y: 131364612 distance: 25
```

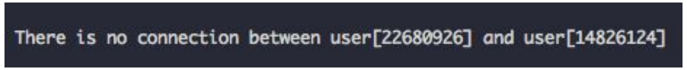
PA 03

Problem 3: A and B are connected when A follows B or B follows A. We can see one cluster of these connections as a single partition. Is A and B connected?

If X follows Y or Y follows X, then X and Y have a connection. Thus, for this problem we do not have to take account of the directions. Since we already have all transitive relationships between the nodes from problem #2 as a file (from Setup for problems 02~05), then we can easily know whether they are connected or not. return whether the two users are connected or not.

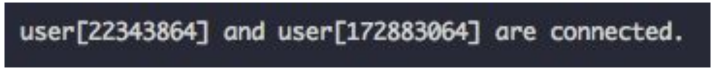


Result when unconnected :



```
There is no connection between user[22680926] and user[14826124]
```

Result when connected :

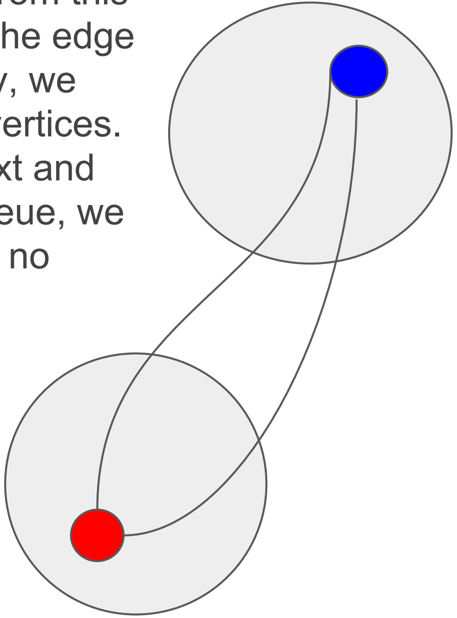


```
user[22343864] and user[172883064] are connected.
```

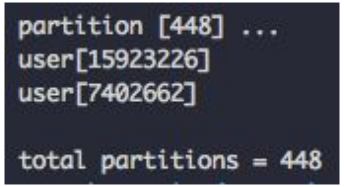
PA 04

Problem 4: Find the number of partitions from the set of friends, where A and B are friends when A follows B and B follows A, or when A and C are friends and C and B are friends, and each friend node is its own friend.

Make a new file of friend list. With this list, go back and run it in the setup stage (*setup for problems 02~05*) to make another file with every friend nodes and friend edges with their information. From this newly produced file, start at one vertex (*node*) v that was not yet visited. Then use the edge to visit adjacent list(s). When we have visited all the vertices that are connected to v, we know that this is one partition. Repeat the steps above until there are no unvisited vertices. The number of repetition is the number of partitions. We used mutable follow set text and queue to implement this. We enqueued adjacent vertices, and if we could not dequeue, we determined that there are no more connected vertices and repeated until there was no unvisited vertices.



Result :



```
partition [448] ...
user[15923226]
user[7402662]

total partitions = 448
```

PA 05.

Problem: Find the top 20 most influential users, where influential users are those who are most frequently visited by a random surfer who surfs into one of the twitter accounts that the user follows by a 90% chance and visits a completely random user(twitter account) by a 10% chance.

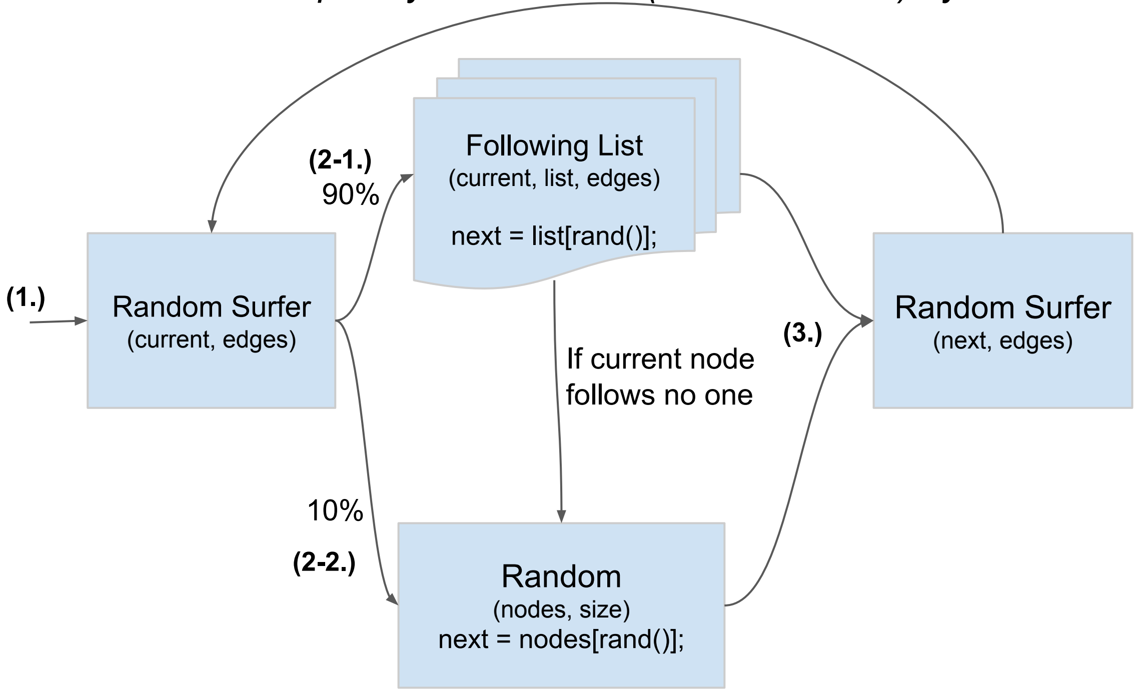


Figure 8. Diagram for Random Surfer

- ❏ We made a recursive random surfer algorithm. Figure 8 shows how our algorithm works.

1. Give random user to Random Surfer
'current' is the given user from its former random surfer or the program
'edges' is the set of all edges. In this case 'twitter_sampled.txt' is the set.

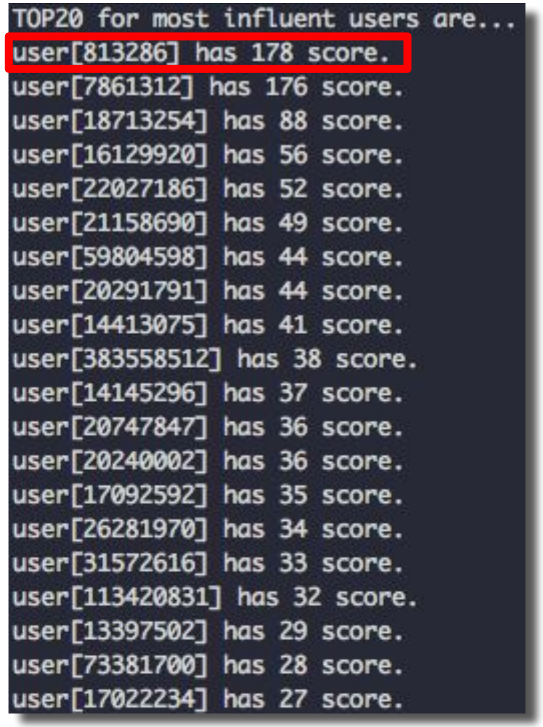
2-1. 90% of probability
random surfer finds the list of followers of 'current', then chooses one from that list and assign it to 'next'. If there is nothing in the list (no follower), then change the state to Random.

2-2. 10% of probability
random surfer gives random node to next (magic)

3. Give 'next' user to Random Surfer again
go through recursion while surfing time.

PA 05.

- Find influential users according to the PageRank



```
TOP20 for most influent users are...
user[813286] has 178 score.
user[7861312] has 176 score.
user[18713254] has 88 score.
user[16129920] has 56 score.
user[22027186] has 52 score.
user[21158690] has 49 score.
user[59804598] has 44 score.
user[20291791] has 44 score.
user[14413075] has 41 score.
user[383558512] has 38 score.
user[14145296] has 37 score.
user[20747847] has 36 score.
user[20240002] has 36 score.
user[17092592] has 35 score.
user[26281970] has 34 score.
user[31572616] has 33 score.
user[113420831] has 32 score.
user[13397502] has 29 score.
user[73381700] has 28 score.
user[17022234] has 27 score.
```

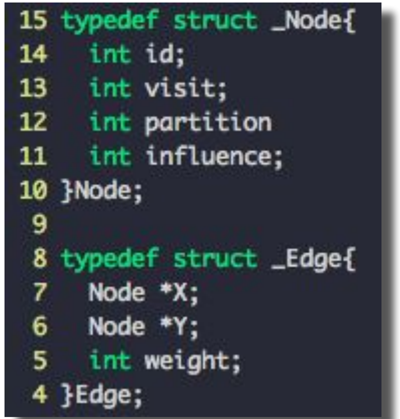
Figure 9 result of Random Surfer program

- ❑ Figure 9 is the result of Random Surfer program.

  In this result, the surfing time was 10000 times. The most influential user's id is 813286. The score of this user '178' means the random surfer visited this user 178 times.

- ❑ While we solving this problem, we faced that it is hard to change the data of User by using data of Edge.

  So we decided to make each edges' components indicate the address of User. (in Figure 10) With this idea, we can easily add their influences.



```
typedef struct _Node{
  int id;
  int visit;
  int partition
  int influence;
}Node;

typedef struct _Edge{
  Node *X;
  Node *Y;
  int weight;
}Edge;
```

Figure 10 Code for make edge indicate user