Random Walker

Manoj Kumar Ganne
Computer Science
University of North Texas
Denton Texas USA
ManojKumarGanne@my.unt.edu

Sri Sai Charan Yarramsetty
Computer Science
University of North Texas
Denton Texas USA
srisaicharanyarramsetty@unt.edu

Objective

Here we have chosen Random Walk (Graph Walker) algorithm and applied it to Twitter database to find the relationship between nodes. Here we have successfully analyzed and implemented the algorithm on Twitter datasets. Below you can find more info on this topic clearly step by step.

Application

We have used Random walker algorithm to find the nodes in Twitter datasets. This algorithm we can use further for below use cases

a. To find nearby nodes

b. Calculate Distance between two nodes

c. To find Node in dataset

d. Find the total number of visited Nodes and Edges

Work Done

- Here we have implemented our own algorithm and implemented the code to achieve the Random Walker functionality.
- Applied random walker algorithm in twitter dataset.
- Process the Huge twitter data into Vector form and create a Grid to apply Random walker algorithm.
- Provided the option to config below options to run algorithm based on our requirements
- walker_path_length - to increase the efficiency of algorithm
- walker_break_point - to stop graph walker. The total functionality we have implemented in 2 parts
    - Next step generation
    - Condition to stop walker

Implementation

We have Implemented the Random walker algorithm on Twitter data and it's too huge to compile and process so we reduced and procced it with 6000*6000 matrix size.

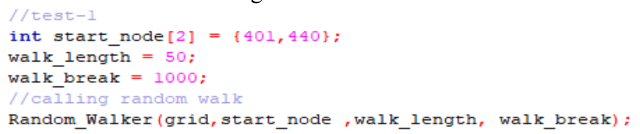
- Std: vector<std::vector<int> > grid(6000, std::vector<int>(6000));

Graph walker will use Radom choices between 0-3 to decide the next move.

- 0 – Down
- 1 – Right
- 2 – Up
- 3 – Left

The Random_walker algorithm function will take below as inputs to find the number of visited nodes and edges.

- Start Node
- Walk Length
- Break Point Length



```
//test-1
int start_node[2] = {401,440};
walk_length = 50;
walk_break = 1000;
//calling random walk
Random_Walker(grid,start_node ,walk_length, walk_break);
```

Tests:

Done the test to check the algorithm execution speed and correctness. Using the sample data and different datasets

- Here observed the time difference between two datasets when we changed the number of walks and path lengths in the algorithm.
- Have to improve the probability to jump into proper node in small iterations.

Experiments:
Done few experiments with different Probability logics,

- Provided different number of walks – 40, 50
- Provided different starting Nodes and walk length.

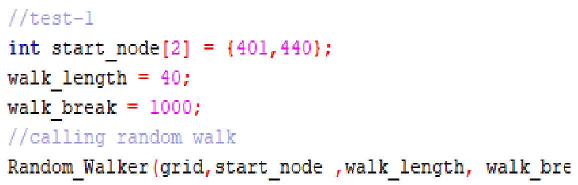
We can decrease the time, by increasing number of walks and walk length in random walks.

Theorem/Proof/Lemma. Insert text here for the enunciation or Math statement. Insert text here for the enunciation or Math statement. Insert text here for the enunciation or Math statement. Insert text here for the enunciation or Math statement. Insert text here for the enunciation or Math statement.

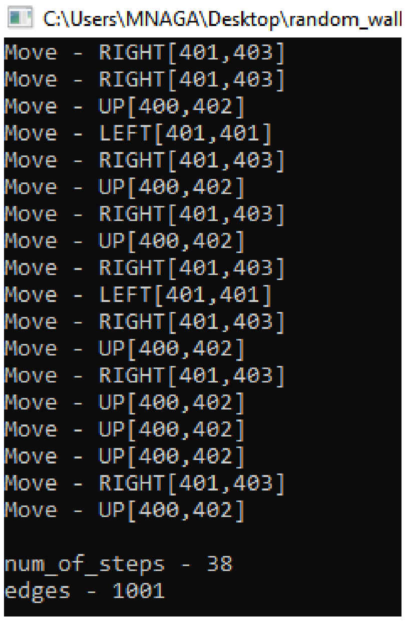
.... Insert text here for the Quotation or Extract, Insert text here for the Quotation or Extract, Insert text here for the Quotation or Extract, Insert text here for the Quotation or Extract, Insert text here for the Quotation or Extract, Insert text here for the Quotation or Extract.

Results

a. Input Test data:



```
//test-1
int start_node[2] = {401,440};
walk_length = 40;
walk_break = 1000;
//calling random walk
Random_Walker(grid,start_node ,walk_length, walk_bre
```

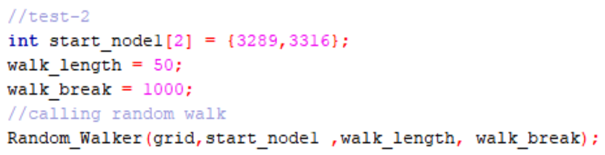


C:\Users\MNAGA\Desktop\random_wall

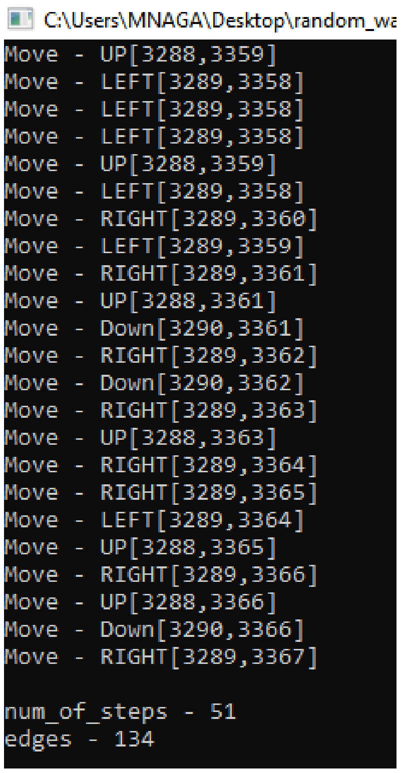
```
Move - RIGHT[401,403]
Move - RIGHT[401,403]
Move - UP[400,402]
Move - LEFT[401,401]
Move - RIGHT[401,403]
Move - UP[400,402]
Move - RIGHT[401,403]
Move - UP[400,402]
Move - RIGHT[401,403]
Move - LEFT[401,401]
Move - RIGHT[401,403]
Move - UP[400,402]
Move - RIGHT[401,403]
Move - UP[400,402]
Move - UP[400,402]
Move - UP[400,402]
Move - RIGHT[401,403]
Move - UP[400,402]

num_of_steps - 38
edges - 1001
```

b. Input Test data 2



```
//test-2
int start_node1[2] = {3289,3316};
walk_length = 50;
walk_break = 1000;
//calling random walk
Random_Walker(grid,start_node1 ,walk_length, walk_break);
```

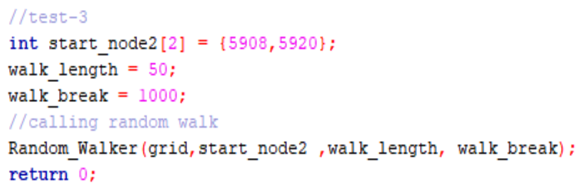


C:\Users\MNAGA\Desktop\random_wa

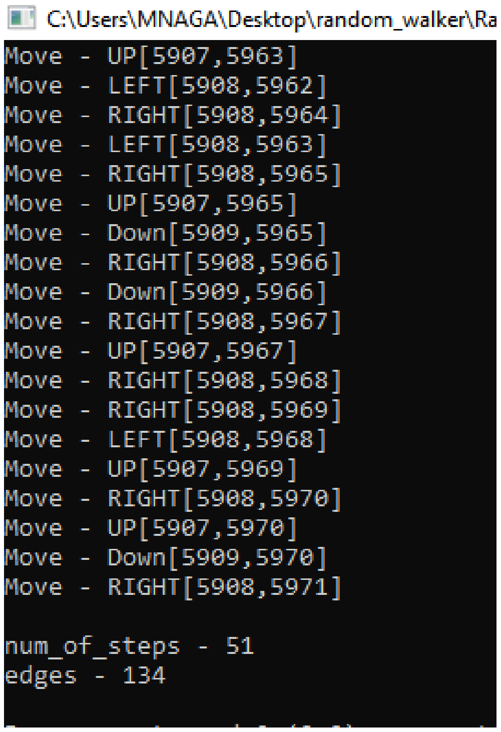
```
Move - UP[3288,3359]
Move - LEFT[3289,3358]
Move - LEFT[3289,3358]
Move - LEFT[3289,3358]
Move - UP[3288,3359]
Move - LEFT[3289,3358]
Move - RIGHT[3289,3360]
Move - LEFT[3289,3359]
Move - RIGHT[3289,3361]
Move - UP[3288,3361]
Move - Down[3290,3361]
Move - RIGHT[3289,3362]
Move - Down[3290,3362]
Move - RIGHT[3289,3363]
Move - UP[3288,3363]
Move - RIGHT[3289,3364]
Move - RIGHT[3289,3365]
Move - LEFT[3289,3364]
Move - UP[3288,3365]
Move - RIGHT[3289,3366]
Move - UP[3288,3366]
Move - Down[3290,3366]
Move - RIGHT[3289,3367]

num_of_steps - 51
edges - 134
```

c. Input Test data 2



```
//test-3
int start_node2[2] = {5908,5920};
walk_length = 50;
walk_break = 1000;
//calling random walk
Random_Walker(grid,start_node2 ,walk_length, walk_break);
return 0;
```



```
C:\Users\MNAGA\Desktop\random_walker\Ra
Move - UP[5907,5963]
Move - LEFT[5908,5962]
Move - RIGHT[5908,5964]
Move - LEFT[5908,5963]
Move - RIGHT[5908,5965]
Move - UP[5907,5965]
Move - Down[5909,5965]
Move - RIGHT[5908,5966]
Move - Down[5909,5966]
Move - RIGHT[5908,5967]
Move - UP[5907,5967]
Move - RIGHT[5908,5968]
Move - RIGHT[5908,5969]
Move - LEFT[5908,5968]
Move - UP[5907,5969]
Move - RIGHT[5908,5970]
Move - UP[5907,5970]
Move - Down[5909,5970]
Move - RIGHT[5908,5971]

num_of_steps - 51
edges - 134
```

Conclusion

Here we successfully implemented and tested the Graph walker on Twitter dataset(reduced). And used rand() to decide the Up, Down, Right and Left moves. Have also executed testcases with different inputs to test the Algorithm performance.

REFERENCES

[1] https://en.wikipedia.org/wiki/Random_walk

[2] Feng Xia, Senior Member, IEEE, Jiaying Liu, Hansong Nie, Yonghao Fu, Liangtian Wan, Member, IEEE, Xiangjie Kong, Senior Member, IEEE

[3] https://neo4j.com/docs/graph-data-science/current/alpha-algorithms/random-walk/

[4] Tong, H., Faloutsos, C., Pan, J.-Y. 2006. Fast random walk with restart and itsapplications. in Sixth international conference on data mining (ICDM’06). IEEE