Introduction to Programming EE2310　Homework 10

103061142 楊淳佑

Problem

We would like to investigate the number of links to reach neighbor nodes which are connected randomly in a ring-shaped network by using BFS.
Find the distance of the farthest node from node 0, then add links one by one randomly and find how the distances between node 0 and the farthest node change accordingly.

Solution, Additional Feature, Program Flow & Structure

Classes and Structures

- `Node` : Include below data of a node in a ring-shaped network:
  - id(integer): The id of the node starts from 0.
  - level(integer): The level or the distance between node 0 and the node. Used in BFS.
  - neighbor(Node pointer vector): The pointers to the neighbor nodes.
- `Link`: Save two nodes to represent the link between those two nodes.

Functions, data structure and program flow

- `main()`
  - Get the node number of the network. (We set it “n+1” here.)
  - Create the node vector and the list vector.
  - Setup the ID of the nodes, starts from 0.
  - Link these nodes on circle:
    1. Link the first node (Node 0) to the last node (n).
    2. Starts from node 0 to node n-1, add links between every node and the next node.
  - Build the list of links:
    1. Link node 0 with every node from node 2 to node n.
    2. Link every node from node 1 to node n-2 with nodes behind (except the next node).
  - Randomize the links list.
    1. Randomly swap the list.
  - Find max length and add links from the randomized links list
    1. Run `FindLenghs` function one time first.

2. Add a link from the top of the links list and pop it back.
3. Run `FindLengths` function again.
4. Repeat 2.~3., until the links list is empty (all links are added to the network).

- `FindLengths(vector<Node> all_nodes, int n, ofstream& output)`
  - Create two Node pointer vector called `currentShell` (to save the nodes to be processed this level) and `nextShell` (to save the nodes going to be processed next level).
  - Print the network list
    1. [Additional Feature] Save every node and their neighbor nodes to the file.
  - Reset levels of all nodes
    1. Set the level of every node to 0.
  - Put starting node (node n) into `currentShell`
    1. Push `all_nodes[n]` to `currentShell`.
  - While there is nodes in `currentShell`, repeat below steps:
    1. Clear `nextShell`.
    2. Add 1 to `distance`.
    3. For each neighbor of a node in the `currentShell`, push back the neighbor to `nextShell` and assign distance to the level of the neighbor if the neighbor's level is zero and the neighbor isn't node n.
    4. Repeat 3. until every node in the `currentShell` has been processed, copy `nextShell` to `currentShell`.
  - Print and save [Additional Feature] the max distance.

Output Result

On screen

```
Enter Nodes: 25
max_distance = 12
max_distance = 12
max_distance = 9
max_distance = 9
max_distance = 6
max_distance = 6
max_distance = 6
max_distance = 6
max_distance = 5
max_distance = 5
max_distance = 4 (*14 times)
max_distance = 3 (*63 times)
max_distance = 2 (*177 times)
max_distance = 1 (*9 times)
```

In file

```
Node 0 has a neighbor of Node 1 24
Node 1 has a neighbor of Node 0 2
Node 2 has a neighbor of Node 1 3
Node 3 has a neighbor of Node 2 4
Node 4 has a neighbor of Node 3 5
Node 5 has a neighbor of Node 4 6
Node 6 has a neighbor of Node 5 7
Node 7 has a neighbor of Node 6 8
Node 8 has a neighbor of Node 7 9
Node 9 has a neighbor of Node 8 10
Node 10 has a neighbor of Node 9 11
Node 11 has a neighbor of Node 10 12
Node 12 has a neighbor of Node 11 13
Node 13 has a neighbor of Node 12 14
Node 14 has a neighbor of Node 13 15
Node 15 has a neighbor of Node 14 16
Node 16 has a neighbor of Node 15 17
Node 17 has a neighbor of Node 16 18
Node 18 has a neighbor of Node 17 19
Node 19 has a neighbor of Node 18 20
Node 20 has a neighbor of Node 19 21
Node 21 has a neighbor of Node 20 22
Node 22 has a neighbor of Node 21 23
Node 23 has a neighbor of Node 22 24
Node 24 has a neighbor of Node 23 0
max_distance = 12

Node 0 has a neighbor of Node 1 24
Node 1 has a neighbor of Node 0 2
Node 2 has a neighbor of Node 1 3
Node 3 has a neighbor of Node 2 4
Node 4 has a neighbor of Node 3 5
Node 5 has a neighbor of Node 4 6 22
Node 6 has a neighbor of Node 5 7
Node 7 has a neighbor of Node 6 8
Node 8 has a neighbor of Node 7 9
Node 9 has a neighbor of Node 8 10
Node 10 has a neighbor of Node 9 11
```

```
Node 11 has a neighbor of Node 10 12
Node 12 has a neighbor of Node 11 13
Node 13 has a neighbor of Node 12 14
Node 14 has a neighbor of Node 13 15
Node 15 has a neighbor of Node 14 16
Node 16 has a neighbor of Node 15 17
Node 17 has a neighbor of Node 16 18
Node 18 has a neighbor of Node 17 19
Node 19 has a neighbor of Node 18 20
Node 20 has a neighbor of Node 19 21
Node 21 has a neighbor of Node 20 22
Node 22 has a neighbor of Node 21 23
5
Node 23 has a neighbor of Node 22 24
Node 24 has a neighbor of Node 23 0
max_distance = 12

Node 0 has a neighbor of Node 1 24
Node 1 has a neighbor of Node 0 2 18
Node 2 has a neighbor of Node 1 3
Node 3 has a neighbor of Node 2 4
Node 4 has a neighbor of Node 3 5
Node 5 has a neighbor of Node 4 6 22
Node 6 has a neighbor of Node 5 7
Node 7 has a neighbor of Node 6 8
Node 8 has a neighbor of Node 7 9
Node 9 has a neighbor of Node 8 10
Node 10 has a neighbor of Node 9 11
Node 11 has a neighbor of Node 10 12
Node 12 has a neighbor of Node 11 13
Node 13 has a neighbor of Node 12 14
Node 14 has a neighbor of Node 13 15
Node 15 has a neighbor of Node 14 16
Node 16 has a neighbor of Node 15 17
Node 17 has a neighbor of Node 16 18
Node 18 has a neighbor of Node 17 19
1
Node 19 has a neighbor of Node 18 20
```

```
Node 20 has a neighbor of Node 19 21
Node 21 has a neighbor of Node 20 22
Node 22 has a neighbor of Node 21 23
5
Node 23 has a neighbor of Node 22 24
Node 24 has a neighbor of Node 23 0
max_distance = 9
```

(And so on.)