BLG 336E
Analysis of Algorithms 2
Homework 1 – DFS and BFS
CRN: 21160
Lecturer's Name : Zehra Çataltepe

Student's Name : Yusuf Utku GÜL
Number : 150150006
Date of Delivery : 06.04.2021

INTRODUCTION

In this assignment it is asked to solve crypt-arithmetic puzzles using Breadth-First Search (BFS) and Depth-First Search (DFS) graph traversal algorithms.

The code is tested at ITU SSH Server with using these command lines.

- g++ main.cpp -std=c++11
- ./a.out BFS WORD1 WORD2 WORD3 solution

IMPLEMENTATION

In my solution I used a tree-graph class which includes size of the tree, puzzle to solve, an adjacency list and node structure containing the number of the node (like a key) and also a solution table containing the number values using map structure.

In the node structure a key need in order to access data better. Each node has a different number which also hints the number of nodes used in the tree.

```
struct node{
    int num;
    map<char, int> sol;
};
```

Also the map structure containing a character as a key and a value is used. It seemed efficient because in map, one key takes only one value and it is easy to access.

The tree structure has one constructor and three methods. Tree constructor takes the maximum size of the tree and the crypt-arithmetic words to solve as input.

Connect method takes two nodes addresses as parameters and adds the connection to the adjacency list.

```
class Tree
{
    int size;    // max size of the tree
    list<node> *adj_list;   // adj list for stack or queue
    string word1;    //input words
    string word2;
    string word3;
public:
    Tree(string word1,string word2,string word3,int size);
    void connect(node &root, node &leaf);   //adds edge to
    void DFS(node a);   //traverse with dfs
    void BFS(node b);   //traverse with bfs
};
```

Also DFS and BFS methods which are used the solve the puzzle takes the starting node as input.

In this program iterative DFS algorithm is used.

- First a list to save if a node has been visited is created with all set to false.
- Initial node is pushed into the stack.
- Then until the stack is empty.
- Top of the stack is popped.
- Check if solution of the node is complete.
  - If so return.
- Discovered is set to true for the node.
- All the adjacent and not visited nodes are pushed into the stack.
- Proceed to the child node if it is a possible solution.

While implementing BFS algorithm, queue is used.

- Like in the DFS, a list to save if a node has been visited is created with all set to false.
- Initial node is added to the queue.
- Then until the queue is empty.
- Front of the queue is enqueued
- Check if solution of the node is complete.
  - If so return.
- Discovered is set to true for the node.
- All the adjacent and not visited nodes are pushed into the queue.
- Proceeded to the neighbour node if it is a possible solution.

ANALYSIS

Some results from the code:
TWO + TWO = FOUR

```
Algorithm:BFS
Word 1:TWO
Word 2:TWO
Word 3:FOUR
Solved
Algorithm: BFS
Number of the visited nodes: 5319
Maximum number of nodes kept in memory: 32
Running time: 0.046 seconds.
Solution:
F       1
O       8
R       6
T       9
U       5
W       2
```

```
Algorithm:DFS
Word 1:TWO
Word 2:TWO
Word 3:FOUR
Solved
Algorithm: DFS
Number of the visited nodes: 1876
Maximum number of nodes kept in memory: 30
Running time: 0.017 seconds.
Solution:
F       1
O       7
R       4
T       8
U       3
W       6
```

DOWN + WWW = ERROR

```
Algorithm:BFS
Word 1:DOWN
Word 2:WWW
Word 3:ERROR
Solved
Algorithm: BFS
Number of the visited nodes: 766
Maximum number of nodes kept in memory: 23
Running time: 0.008 seconds.
Solution:
D       9
E       1
N       4
O       3
R       0
W       6
```

```
Algorithm:DFS
Word 1:DOWN
Word 2:WWW
Word 3:ERROR
Solved
Algorithm: DFS
Number of the visited nodes: 347
Maximum number of nodes kept in memory: 24
Running time: 0.004 seconds.
Solution:
D       9
E       1
N       4
O       3
R       0
W       6
```

SEND + MORE = MONEY

```
Algorithm:BFS
Word 1:SEND
Word 2:MORE
Word 3:MONEY
Solved
Algorithm: BFS
Number of the visited nodes: 637293
Maximum number of nodes kept in memory: 43
Running time: 6.88 seconds.
Solution:
D       7
E       5
M       1
N       6
O       0
R       8
S       9
Y       2
```

```
Algorithm:DFS
Word 1:SEND
Word 2:MORE
Word 3:MONEY
Solved
Algorithm: DFS
Number of the visited nodes: 434911
Maximum number of nodes kept in memory: 41
Running time: 4.465 seconds.
Solution:
D       7
E       5
M       1
N       6
O       0
R       8
S       9
Y       2
```

Even thogh both BFS and DFS has the same time complexity, run time of BFS generally took more time due to the structure of the graph. Since this graph is in the shape of a tree and the runtime completed when reached to the final layer, DFS gets that layer quicker while BFS searches widely.

Maximum number of nodes kept in the memory for BFS is more than DFS. Because it keeps nodes layer by layer and those layers are getting larger as it proceeds.

Graph is undirected so, If such a list is not kept it could cause a loop in the graph. So, discovered nodes list should be kept for not to visit the discovered nodes again.