ISTANBUL TECHNICAL UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

BLG 336E

ANALYSIS OF ALGORITHMS II

PROJECT I REPORT

DATE: 03.04.2021

STUDENT NAME: ABDULKADİR PAZAR

STUDENT NO: 150180028

SPRING 2020

**Project I Report**

**1.)**

1. **Node Structure and Tree Structure**

My nodes consist of a vector<pair<char,int>> to store letter number pairs and a node array of size ten to store child nodes. createTree() function I wrote produces the tree structure requested from us using recursion.

1. **Pseudocode**

**createTree(node, layer)**

if (layer < # of unique letters)

for (digits from 0 to 9)

copy data from node

add {unique letters[layer], j} pair to copied data

create new node

node->children[digit] = new node

createTree(new node, layer + 1)

**BFS(node)**

if(solution not found)

create queue q

enqueue node to queue

while (q is not empty)

if (queue size exceeds maximum number of nodes)

max nodes = queue[size]

# of visited nodes++

current node = dequeue from q

if (current node has duplicate numbers)

continue;

if (if one of the last digits is 0)

continue;

if (solution not found and current node is leaf and sum check is true)

result = current node[data]

solution found = true

return;

for (digits from 0 to 9)

enqueue(current node[children[digit]]

**DFS(node, layer)**

if(solution is not found and there are remaining nodes)

# of visited nodes++;

current # of nodes++

if (current nodes exceeds max nodes in memory)

max nodes = current # of nodes

if (duplicate number exists)

current # of nodes--

return

if (one of last digits is 0)

current # of nodes—

return

if (solution not found and node is leaf and sum check is true)

result = node[data]

solution found = true

current # of nodes--

return

for (digits from 0 to 9)

DFS(node->children[digit], depth + 1)

current # of nodes—

**c)**

The time complexities of both DFS and BFS are O(10n) as the solution is guaranteed to be in a leaf and there are 10n leaves where n is the number of unique letters.

**2.)**

TWO + TWO = FOUR

|  |  |  |
| --- | --- | --- |
|  | DFS | BFS |
| The number of visited nodes | 205556 | 219410 |
| Max. number of nodes in memory | 7 | 928704 |
| The running time (seconds) | 0.103 | 0.223 |

DOWN + WWW = ERROR

|  |  |  |
| --- | --- | --- |
|  | DFS | BFS |
| The number of visited nodes | 242090 | 245152 |
| Max. number of nodes in memory | 7 | 1004322 |
| The running time | 0.146 | 0.242 |

The program runs out of memory and throws an instance of bad\_alloc() while trying to solve SEND + MORE = MONEY as creating a tree with 11111111 nodes consumes all of memory. This way of approaching the problem is very slow and quickly runs out of memory as creating the tree has exponential time and space complexity. To solve the puzzle efficiently, I would not create a tree data structure. Instead, I would use a recursive permutation function to try all possible values. This would produce a result much faster and would have a lesser chance of running out of memory.

**3.)**

We do not keep a list of discovered nodes as our algorithms traverse on a tree (there are no loops). This does not affect the outcome of the algorithm as we do not need to check if we visited a certain node before in any of the algorithms.