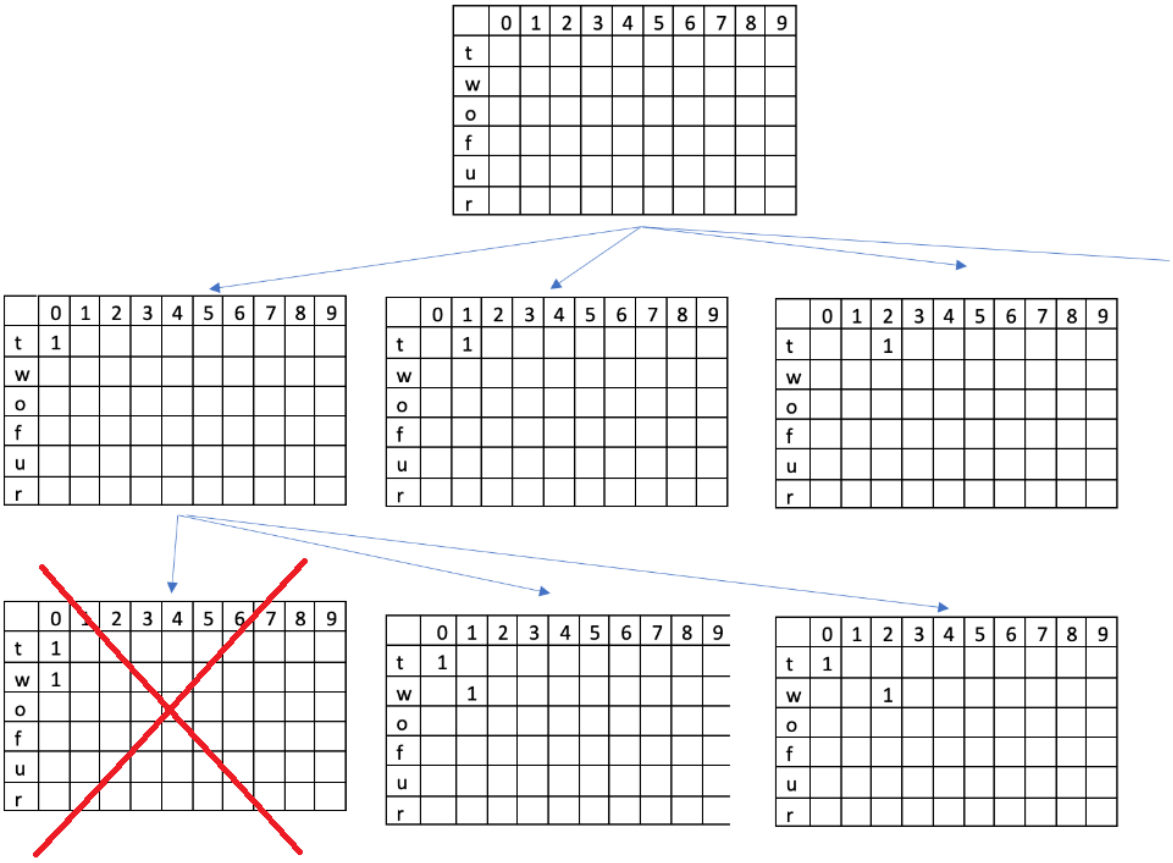
İlya Mustafa Nuhi
150180710

HOMEWORK1 REPORT

1-)

a-) Describe node and assignment representations in detail.

Every node stores a matrix which has 10 columns and amount of rows equal to given total distinct letters. Our Tree starts with the root node which is a blank matrix and it has 10 child. And going deeper in the childs, every child has 1 less child because same letters can not assign with the same numbers. So as you can see in the figure below since the node which t=0 can not have node with w=0 because t already assigned to 0; every child have 1 less child and layers have following pattern of nodes: first layer(root):1 , second layer:1.10=10, third layer: 10.9=90, fourth layer: 90.8=720 and so on.



İlya Mustafa Nuhi
150180710

b-) Write your pseudo-code.

DFS function pseudo code

```
create answer pointer
set answer pointer to result of the DFS_helper function(returns the answer node pointer)

if answer pointer is NULL
        print "There is no solution for this puzzle"
else
        return answer
```

DFS_helper function pseudo code

```
node_visited++
create tmp pointer and set it NULL
create iterator scope char vector “it”
if current searched node is a leaf node {                    // checks if the searched node is the asnwer
        initialize numb1, numb2, sum to 0
        initialize big_num_size and set it to longest adding number
        for ( int i =0 ; i < big_num_size ; i++ ){
                initialize numeric1, numeric2, numeric3 to 0
                if numb1 is smaller than numb2 and i==big_num_size-1
                        set numeric1 to 1
                else{
                        find the address of the number1’s letter in the letters vector and set it to “it” vector
                        initialize index_num1 and set it to index of the letter in the letters vector

                        while(not seeing 1 in the matrix of the searched letter)
                                numeric1++
                }
                if numb2 is smaller than numb1 and i==big_num_size-1
                                        set numeric2 to 1
```

```
        else{
                find the address of the number2's letter in the letters vector and set it to "it" vector
                initialize index_num2 and set it to index of the letter in the letters vector

                while(not seeing 1 in the matrix of the searched letter)
                        numeric2++
        }
        find the address of the sum's letter in the letters vector and set it to "it" vector
        initialize index_num2 and set it to index of the letter in the letters vector

        while(not seeing 1 in the matrix of the searched letter)
                numeric3++
        set numb1 += numeric1 * 10^i
        set numb2 += numeric2 * 10^i
        set sum += numeric3 * 10^i
}
Set sum += 10^(big_num_size)
İf numb1 + numb2 == sum then return the answer node temp else return null

For (int i=0; i< child_numb; i++){
        İf first letter of the number1, number 2 or result is 0 return null
        İf first letter of the result is not 1 return null
        Set tmp to DFS_helper function and send child_numb-1 and i as parameters
        İf answer is found return tmp
}
İf above for loop is finished and couldn't find the solution return NULL
```

İlya Mustafa Nuhi
150180710

BFS function pseudo code

```
create answer pointer
set answer pointer to result of the BFS_helper function(returns the answer node pointer)

if answer pointer is NULL
        print "There is no solution for this puzzle"
else
        return answer
```

BFS_helper function pseudo code

```
İnitialize queue
Push the root node of the tree to queue
While queue is not empty{
   Create searching_node node pointer
   Set searching_node to first node of the queue
   Pop the first element from queue
   İf(searching_node is leaf node){          // checks if searching_node is the answer
        initialize numb1, numb2, sum to 0
        initialize big_num_size and set it to longest adding number
        for ( int i =0 ; i < big_num_size ; i++ ){
            initialize numeric1, numeric2, numeric3 to 0
            if numb1 is smaller than numb2 and i==big_num_size-1
                set numeric1 to 1
            else{
                find the address of the number1's letter in the letters vector and set it to "it" vector
                initialize index_num1 and set it to index of the letter in the letters vector

                while(not seeing 1 in the matrix of the searched letter)
                    numeric1++
            }
            if numb2 is smaller than numb1 and i==big_num_size-1
                set numeric2 to 1
```

```
                else{
                        find the address of the number2's letter in the letters vector and set it to "it" vector
                        initialize index_num2 and set it to index of the letter in the letters vector

                        while(not seeing 1 in the matrix of the searched letter)
                                numeric2++
                }
                find the address of the sum's letter in the letters vector and set it to "it" vector
                initialize index_num2 and set it to index of the letter in the letters vector

                while(not seeing 1 in the matrix of the searched letter)
                        numeric3++
                set numb1 += numeric1 * 10^i
                set numb2 += numeric2 * 10^i
                set sum += numeric3 * 10^i
        }
        Set sum += 10^(big_num_size)
        İf numb1 + numb2 == sum then return searching node
    }
    İf(searching node has childs){
        for (int i =0; i< searching_node's child number; i++){
                İf first letter of the child[i]'s number1, number 2 or result is 0 continue the for loop
                İf first letter of the child[i]'s result is not 1 continue the for loop
                İf child[i] doesn't face with any constarints pushback the child to the queue

        }
     }
}
Return NULL if can't find the solution
```

İlya Mustafa Nuhi
150180710

c-) Show the complexity of your algorithm on the pseudo-code

in DFS_helper function:

Explenation:

I am doing the recursive part in this for loop by calling the DFS_helper function and sending parameters child[i] node, child_numb-1, and i. So i can search the tree in depth search first method. Also if current searched node doesn't match with the constraints recursive function will not be called for this node.

```
For (int i=0; i< child_numb; i++){
        İf first letter of the number1, number 2 or result is 0 return null
        İf first letter of the result is not 1 return null
        Set tmp to DFS_helper function and send child_numb-1 and i as parameters
        İf answer is found return tmp
}
```

Complexity of DFS:

We are actually doing stack implementation on the DFS algorithm but since it is a Tree structure its complexity is O(n), n is the number of nodes.

İn BFS_helper function:

Explenation:

My BFS search algorithm works with iterative method because in breadth first search we have to search the layers first. So my function works in while loop which works while queue is not empty. And at the end of my while loop, in the for loop i add the searched node's childs to queue so that while loop will work until solution is found or every node has checked. Also if current searched node's childs doesn't match with the constraints they are not added to queue.

İlya Mustafa Nuhi
150180710

```
While queue is not empty{


    İf(searching node has childs){
        for (int i =0; i< searching_node's child number; i++){
            İf first letter of the child[i]'s number1, number 2 or result is 0 continue the for loop
            İf first letter of the child[i]'s result is not 1 continue the for loop
            İf child[i] doesn't face with any constarints pushback the child to the queue


        }
```

Complexity of BFS:

We are actually doing BFS implementation using queue so the complexity is

O(n) , n is the number of nodes.

İlya Mustafa Nuhi
150180710

2-) Analyze and compare the algorithm results in terms of:

```
ilya@DESKTOP-OUTI5O2:/mnt/c/users/ilyan/source/repos/algo2_hw1/algo2_hw1$ ./hw1 BFS TWO TWO FOUR output.txt
Algorithm: BFS
Maximum number of nodes kept in the memory: 187300
Number of the visited nodes: 17314
Running time: 0.0167835
Solution: T: 7, W: 3, O: 4, F: 1, U: 6, R: 8
ilya@DESKTOP-OUTI5O2:/mnt/c/users/ilyan/source/repos/algo2_hw1/algo2_hw1$ g++ algo2_hw1.cpp -o hw1
ilya@DESKTOP-OUTI5O2:/mnt/c/users/ilyan/source/repos/algo2_hw1/algo2_hw1$ ./hw1 DFS TWO TWO FOUR output.txt
Algorithm: DFS
Maximum number of nodes kept in the memory: 187300
Number of the visited nodes: 14421
Running time: 0.0203343
Solution: T: 7, W: 3, O: 4, F: 1, U: 6, R: 8
ilya@DESKTOP-OUTI5O2:/mnt/c/users/ilyan/source/repos/algo2_hw1/algo2_hw1$ g++ algo2_hw1.cpp -o hw1
ilya@DESKTOP-OUTI5O2:/mnt/c/users/ilyan/source/repos/algo2_hw1/algo2_hw1$ ./hw1 BFS SEND MORE MONEY output.txt
Algorithm: BFS
Maximum number of nodes kept in the memory: 2606500
Number of the visited nodes: 236442
Running time: 0.317412
Solution: S: 9, E: 5, N: 6, D: 7, M: 1, O: 0, R: 8, Y: 2
ilya@DESKTOP-OUTI5O2:/mnt/c/users/ilyan/source/repos/algo2_hw1/algo2_hw1$ g++ algo2_hw1.cpp -o hw1
ilya@DESKTOP-OUTI5O2:/mnt/c/users/ilyan/source/repos/algo2_hw1/algo2_hw1$ ./hw1 DFS SEND MORE MONEY output.txt
Algorithm: DFS
Maximum number of nodes kept in the memory: 2606500
Number of the visited nodes: 247773
Running time: 0.427776
Solution: S: 9, E: 5, N: 6, D: 7, M: 1, O: 0, R: 8, Y: 2
ilya@DESKTOP-OUTI5O2:/mnt/c/users/ilyan/source/repos/algo2_hw1/algo2_hw1$ g++ algo2_hw1.cpp -o hw1
ilya@DESKTOP-OUTI5O2:/mnt/c/users/ilyan/source/repos/algo2_hw1/algo2_hw1$ ./hw1 BFS DOWN WWW ERROR output.txt
Algorithm: BFS
Maximum number of nodes kept in the memory: 187300
Number of the visited nodes: 18810
Running time: 0.0330824
Solution: D: 9, O: 3, W: 6, N: 4, E: 1, R: 0
ilya@DESKTOP-OUTI5O2:/mnt/c/users/ilyan/source/repos/algo2_hw1/algo2_hw1$ g++ algo2_hw1.cpp -o hw1
ilya@DESKTOP-OUTI5O2:/mnt/c/users/ilyan/source/repos/algo2_hw1/algo2_hw1$ ./hw1 DFS DOWN WWW ERROR output.txt
Algorithm: DFS
Maximum number of nodes kept in the memory: 187300
Number of the visited nodes: 37860
Running time: 0.0457417
Solution: D: 9, O: 3, W: 6, N: 4, E: 1, R: 0
```

Results of my BFS and DFS algorithm

a-) the number of visited nodes
As you can see in my results

For the example BFS TWO TWO FOUR: Visited nodes: **17314**

For the example DFS TWO TWO FOUR: Visited nodes: **14421**

For the example BFS SEND MORE MONEY: Visited nodes: **236442**

For the example DFS SEND MORE MONEY: Visited nodes: **247773**

For the example BFS DOWN WWW ERROR: Visited nodes: **18810**

For the example DFS DOWN WWW ERROR: Visited nodes: **37860**

İlya Mustafa Nuhi
150180710

b-) the maximum number of nodes kept in the memory
As you can see in my results

TWO TWO FOUR and DOWN WWW ERROR have 6 distinct letters.
Because we can't assign same letters with same numbers, every child should has 1 less child so for 6 distinct letters:

=1.10 + 10.9 + 90.8 + 720.7 + 5040.6 + 30240.5
=10 + 90 + 720 + 5040 + 30240 + 151200 = **187300**

SEND MORE MONEY have 8 distinct letters.
So for 8 distinct letters:

=1.10 + 10.9 + 90.8 + 720.7 + 5040.6 + 30240.5 + 151200.4 + 604800.3
=10 + 90 + 720 + 5040 + 30240 + 151200 + 604800 + 1814400 = **2606500**

c-)the running time
As you can see in my results

For the example BFS TWO TWO FOUR: Running Time: **0.016783 seconds**

For the example DFS TWO TWO FOUR: Running Time: **0.020334 seconds**

For the example BFS SEND MORE MONEY: Running Time: **0.317412 seconds**

For the example DFS SEND MORE MONEY: Running Time: **0.427776 seconds**

For the example BFS DOWN WWW ERROR: Running Time: **0.033082 seconds**

For the example DFS DOWN WWW ERROR: Running Time: **0.045741 seconds**

3-) Discuss why we should maintain a list of discovered nodes? How this affects the outcome of the algorithms?

İf we would implement a graph for this homework, we should maintain a list of discovered nodes because of preventing unnecessary node checking but since we are using tree implementation we don't have to maintain list of discovered nodes.