MIDSEM LAB EXAM

Design and Analysis of Algorithm

IDAA432C

Prepare a matrix of random characters of size 50 50, check for valid English words diagonally.

Submitted By:
Raagini Mandal **(IIT2017077)**
Naman Deept **(IIT2017507)**

April 6, 2019

Generate a 50×50 matrix of random characters and check for valid english words diagonally

Raagini Mandal IIT2017077,   Naman Deept IIT2017507

Dept. of Information Technology
Indian Institute of Information Technology Allahabad

April 6, 2019

1 Abstract

This paper introduces an efficient algorithm for Generating a 50×50 matrix of random characters and check for valid english words diagonally
Keywords: *Matrix*, *left diagonal*, *Right diagonal*, *english words*, *Validity*

2 Introduction

In mathematics, a matrix is a rectangular array of numbers, symbols, or expressions, arranged in rows and columns.

3 Proposed Method

Input : we need to generate a matrix of size 50× and check for valid diagonal elements. We check both for left as well as right diagonals. Therefore for all elements with indexes i×i and with indexes i×(n-i).

3.1 Checking valid words

we generate a character array to store our diagonal words and then check if any substring matches with the words we have already included in a text file.

3.1.1 Algorithm

Here, we generate the matrix of size 50×50 and then fill up with the random characters from 'a' to 'z' in the matrix.

```
procedure GENERATEMATRIX($size$)
    $matrix \leftarrow null$
    for $i \leftarrow 1\, to\, n$
        for $j \leftarrow 1\, to\, n$
            $matrix[i][j] \leftarrow randomcharacter$
        end for

```

Generating diagonals: Here we generate the diagonal of a matrix from the given matrix of random characters obtained in the above.

```
procedure $generateDiagonals(matrix)$
    $ldiagonal \leftarrow null$
    for $i \leftarrow 1\, to\, n$
        $ldiadonal_i \leftarrow matrix_{i,i}$
    end for
    $rdiagonal \leftarrow null$
    for $i \leftarrow 1\, to\, n$
        $rdiadonal_i \leftarrow matrix_{i,size-i}$
    end for
end procedure=0
```

Extracting all possible substrings: Here from the generated left and right diagonal , we extract all the possible substrings from the matrix ,via the naive approach which will be of the order of n^3 time complexity where n is the size of the matrix.

```
procedure SUBSTRING($ArrayDiagonal$,$size$)
    $DiagStringArray \leftarrow null$
    for $i \leftarrow 0\ to\ n-1$
        for $i \leftarrow i+1\ to\ n-1$
            $String \leftarrow EmptyString$
            for $k \leftarrow i\ to\ j$
                $String \leftarrow String + Diagonal_k$
            end for
            $DiagStringArray \leftarrow DiagStringArray + String$
        end for
    end for
end procedure
```

```
procedure (num)
    $Result \leftarrow emptyString$
    for $i \leftarrow 1$ to $num$
        $Result \leftarrow Result + 0$
    end for
    return $Result$
```

3.1.2 Time Complexity Analysis

Average Case Analysis : average case occurs at a time complexity of order of n^3. Also our best case and worst case is same as our average case.

4 Analysis & Experimentation

For the best case scenario, it occurs when the word we are searching is found right in the begining of the txt file .Worst case occurs if its an invalid word in that case we need to scan through the whole file till we reach the end.

```
procedure RETURNBACK(num)
    $str \leftarrow emptyString$
    $res \leftarrow emptyString$
    $str \leftarrow num$ ToString
    for $i \leftarrow 1$ to $length(str)$
        if $temp.charAt(i)\ equals$ '.'
            continue
            $res \leftarrow res + str.charAt(i)$
        end if
    end for
    return $res$ To Integer
```

Table 1: Size of character matrix vs Time Taken for execution

Size	Time (in secs)
5	0.062
10	0.125
15	0.375
20	0.89
25	1.75
30	2.99
35	4.765
40	7.234
45	10.593
50	14.53
55	19.187
60	24.64

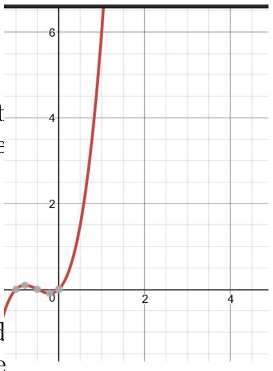

Figure 1: Figure: Graph showing the experimental analysis

analysis.jpg
Figure: graph shows best ,average and worst case in time complexities

This graph shows analysis of our time complexities.

5 Discussion and Future Work

Here we used random character generator as we need to store random characters in our created matrix. Then we extracted the left and right diagonals as character arrays. We included a seperate file that contains the valid words and we use it for making comparisions with the substrings that we generated from the left and right diagonals for checking validity .

6 Conclusion

The given problem was solved using random character generator and by including a file which contains the words and we use it for checking the words.

7 References

1. IDAA432C (Design and Analysis of Algorithm) class lecture

2. Introduction to Algorithms by Cormen