Design Analysis of Algorithm Assignment-2

Kaushal - IIT2019030,
Sarvesh - IIT2019031,
Aarushi - IIT2019032

Abstract

In this paper we have prepared a matrix of random characters of given size 50×50 to check for valid English words diagonally. Here we have also discussed about the time complexity of the algorithm used.

1 Keywords

Dictionary, Matrix, String, Array.

2 Introduction

According to the given problem statement, we had to check if any word in the diagonals are valid or not, therefore we have made use of PyEnchant library from python. Enchant is used to check the spelling of words and suggest corrections for words that are miss-spelled. It can use many popular spellchecking packages to perform this task, including Ispell, Aspell and MySpell. It is quite flexible at handling multiple dictionaries and multiple languages. Dictionaries are created using a language tag which specifies the language to be checked The language tag used here signifies American English. If the language tag is not specified, an attempt is made to determine the language currently in use. This is not always possible, in which case an Error is raised.

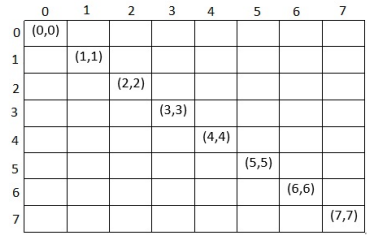
3 Application

Python's great library system made it easier to check for valid English words available in the diagonal of the randomly generated matrix. PyEnchant is a spellchecking library for Python, based on the excellent Enchant library. It combines all the functionality of the underlying Enchant library with the flexibility of Python and a nice "Pythonic" object-oriented interface. Used mainly for dictionary related purposes, and spelling checking domains or other work related to that.

4 Algorithm

Our algorithm iterates the elements and check if it is present on the main diagonal then push it on the string. Then we are checking whether the string is present in the dictionary or not with the help of enchant library. If the string is present the it append to the array of string and in last it print all the strings

which is present in the dictionary.



Running through the matrix it searches for valid words along the diagonal, as mentioned in the above figure. This algorithm runs for each column of the matrix hence takes 50x50 time.

5 Pseudo code

```
import string
import random
import enchant
d = enchant.Dict("en_US")
rows, cols = (50, 50)
arr = []
for i in range(cols):
col = []
for j in range(rows):
col.append(random_character)
arr.append(col)
ans = []
for i in range(0,cols):
s = ""
for j in range(i,rows):
s+ = arr[j][j]
if(len(s) == 1 and s.upper() != 'A'):
continue
if(d.check(s)):
ans.append(s)
print(ans)
```

6 Complexity analysis

6.1 Time Complexity:

In case of time complexity the loops are running completely through the entire fixed matrix for twice. So, after accordingly analysing the pseudo code we can infer that time complexity will be the size of matrix.

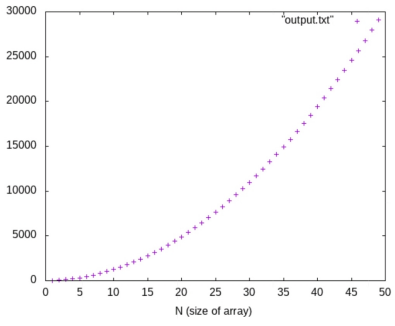
6.2 Space Complexity:

As the size for the matrix is fixed according to the problem statement thus, the space used by any of the algorithm will be constant and equal to that fixed value. Here the space complexity is 50x50.

7 Experimental Study

7.1 Apriori Analysis:

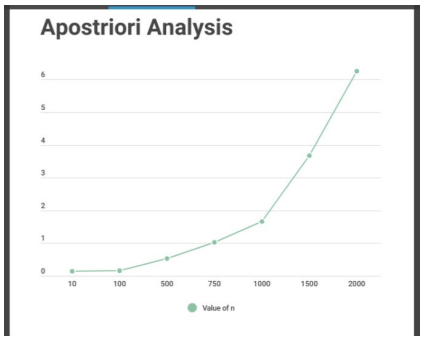
Apriori analysis means,analysis is performed prior to running it on a specific system.This analysis is a stage where a function is defined using some theoretical model.Hence,we determine the time and space complexity of an algorithm by just looking at the algorithm rather than running it on a particular system with a different memory,processor, and compiler.So,as we discussed under the heading complexity analysis we arrived at the conclusion that time complexity is O(nxn).



7.2 Aposteriori Analysis:

Aposteriori analysis of an algorithm means we perform analysis of an algorithm only after running it on a system.It directly depends on the system and changes from system to system.So for the a aposteriori analysis of the algorithm,we have run our code on the compiler and get values of the time by specifying the value of n and m and different value of the time had occur.

Size of Array	Time Taken
10	0.15
100	0.16
500	0.53
750	1.02
1000	1.66
1500	3.67
2000	6.25



8 Conclusion

Through this assignment we learnt about the PyEnchant library in python, and applied it in calculating valid words in the randomly generated matrix of characters.

9 References

https://pyenchant.github.io/pyenchant/tutorial.html
https://pypi.org/project/pyenchant/