DAA Assignment 2

Vidushi Pathak (IIT2019027)
Shubham Netam (IIT2019028)
Puneet Singhwaiya (IIT2019029).
B.Tech. 4th semester, Department of Information Technology.
*Indian Institute Of Information Technology, Allahabad

***Abstract*—In this paper we have discussed the algorithm to generate a 50x50 matrix of random character and to check for valid English words reverse diagonally. Here we have also discussed the time complexity of the algorithm used.**

I. Introduction

In this algorithm we have used a python library PyEnchant to check if a word is valid english word or not. PyEnchant is a spellchecking library for Python, based on the excellent Enchant library. PyEnchant combines all the functionality of the underlying Enchant library with the flexibility of Python and a nice "Pythonic" object-oriented interface.
For generating random characters we have used Python random.choice() function.Python random module's random.choice() function returns a random element from the non-empty sequence. we can use the random.choice() function for selecting a random password from word-list, Selecting a random item from the available data.

II. Algorithm

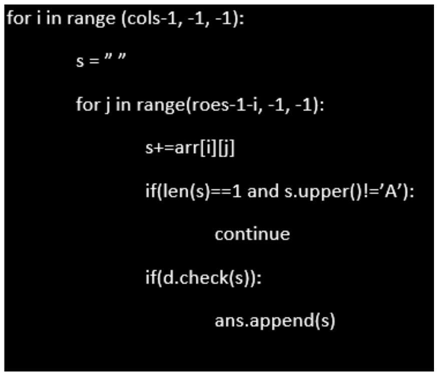
Our algorithm iterates the elements and check if it is present on the main diagonal in reverse direction then push it in the string. Then it checks whether the string is present in the dictionary or not with the help of enchant library. If the string is present then it append to the array of string and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

III. Pseudo Code

i.First we will create a matrix of size 50x50 of random characters by using random.choice function.
ii.Then we run a loop for 50 times in reverse diagonal direction for just getting a string from these random characters one by one.
iii.Then we check that if the size of the string is greater than one or if the string value is 'A', then only it is possible that the string is present in the dictionary.
iv.If present, then we directly print that string.

IV. Time Complexity

As we can see in figure 1. that the outer for loop will run n times and the inner for loop will run n(n+1)/2 times. Therefore the time complexity of the algorithm will be O(n^2).



```
for i in range (cols-1, -1, -1):
    s = " "
    for j in range(roes-1-i, -1, -1):
        s+=arr[i][j]
        if(len(s)==1 and s.upper()!='A'):
            continue
        if(d.check(s)):
            ans.append(s)
```

Fig. 1. code

V. 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. Hence the space complexity is O(1).

VI. Experimental Study

A. Apriori Analysis:

Apriori analysis means,analysis is performed prior to running it on a specfic 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).

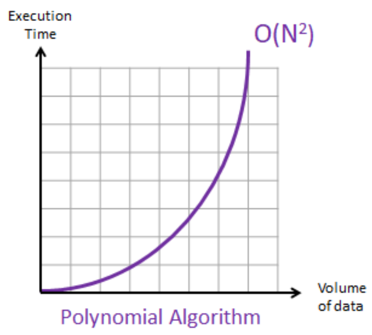
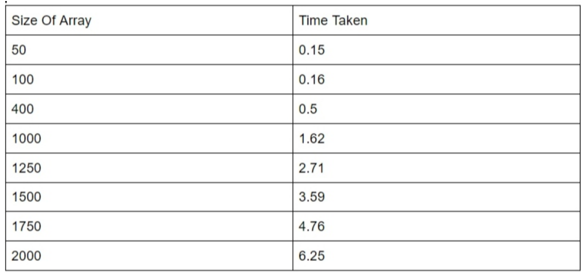


Fig. 2.

B. 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
50	0.15
100	0.16
400	0.5
1000	1.62
1250	2.71
1500	3.59
1750	4.76
2000	6.25

Fig. 3.

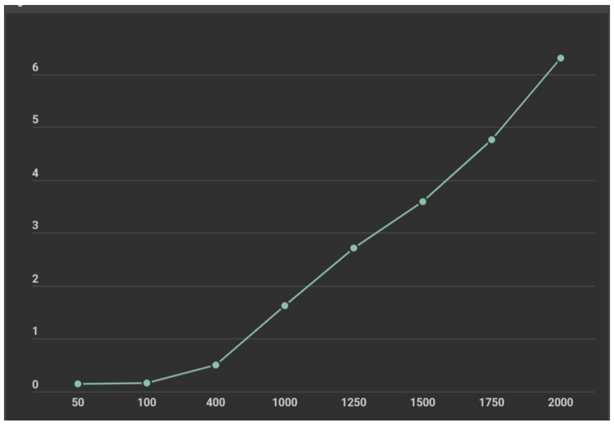


Fig. 4.

VII. 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

VIII. References

https://pyenchant.github.io/pyenchant/
https://pyenchant.github.io/pyenchant/tutorial.html
https://pynative.com/python-random-choice/