CSE 4094 PROJECT 1 REPORT

Celal Bayrak 150144044

Selection of Data Structure:

I think the main challenge of this project is searching through data structure. I decided to choose the simplest one for ease of search. Suffix tree and compress trie are more complex compared to standart trie. So I decided to use standart trie.

Explanatition of Code:

```
10 class TrieNode:
11     def __init__(self):
12         self.children = [None]*300
13         self.isEndOfWord = False
14         self.indexes=[]
15         self.filenames=[]
16 
17 class Trie:
18     def __init__(self):
19         self.root = self.getNode()
20 
21     def getNode(self):
22         return TrieNode()
```

This is structure of struct which I used. Each struct has at most 300 children struct and each struct has a boolean variable which named “isEndOfWord” and 2 lists. “indexes” list used for first query, keeping indexes of inserted words. “filenames” list used for second query, keeping file names of inserted words.

```
def charToIndex(self,ch):
    if 48<=ord(ch) and 57>=ord(ch):
        return (ord(ch)-22)
    else:
        return (ord(ch)-ord('a'))
```

“charToIndex” function returns index for each letter, then I used this index for inserting trie. For example if letter is ‘b’, index value is 1. Then I inserted ‘b’ to children[1].

```
def insert(self,key,word_index,file_name):
    pCrawl = self.root
    length = len(key)
    for level in range(length):
        index = self.charToIndex(key[level])
        if not pCrawl.children[index]:
            pCrawl.children[index] = self.getNode()
        pCrawl = pCrawl.children[index]
    pCrawl.isEndOfWord = True
    pCrawl.indexes.append(word_index)
    pCrawl.filenames.append(file_name)
    pCrawl.filenames=list(set(pCrawl.filenames))
```

“insert” function inserts letters to trie. When word finishes “isEndOfWord” variable is True. Also inserts file names and index to struct.

```
def search(self, key):
    arr=[]
    pCrawl = self.root
    length = len(key)
    for level in range(length):
        index = self.charToIndex(key[level])
        if not pCrawl.children[index]:
            return False
        pCrawl = pCrawl.children[index] #key found

    self.recursive(pCrawl,arr,key)    #searches under key
```

“search” function searches given string in the trie. When it founds the key then it calls a recursive function which named as “recursive”.

```
def recursive(self,pCrawl,arr,key):
    chars=""
    if pCrawl is None:
        return
    if pCrawl.isEndOfWord:
        for i in arr:
            chars=chars+i
        print(key+chars)
        print("index: "+str(pCrawl.indexes))
        print("file name: " + str(pCrawl.filenames))
        print("----------------------------------------")
    for i in range(0,300):
        if pCrawl.children[i]:
            if i>=26:
                char=chr(i+22)
            else:
                char=chr(i+ord('a'))
            arr.append(char)
            self.recursive(pCrawl.children[i],arr,key)
            arr.pop()
```

"recursive" function searches the trie for given word. In "search" function we found the key and we are looking for continuation of word. I give the node, which search stayed, as parameter to "recursive" function. When we reach the end of word by cheching "isEndOfWord" variable, function prints the complete word, index and file names. Also "arr" keeps the incurred letters and when we are returning back to branching node, "arr" discharges its elements.

For first query above functions are enough.

```
def recursive_common(self,pCrawl,arr,file_len):
    chars=""
    if pCrawl is None:
        return
    if pCrawl.isEndOfWord:
        if len(pCrawl.filenames)==file_len:
            for i in arr:
                chars=chars+i
            print(chars)
    for i in range(0,300):
        if pCrawl.children[i]:
            if i>=26:
                char=chr(i+22)
            else:
                char=chr(i+ord('a'))
            arr.append(char)
            self.recursive_common(pCrawl.children[i],arr,file_len)
            arr.pop()
```

For second query, "recursive_common" function prints the common words. The difference from "recursive" is this function checks the length of "filenames" list. If the length is equal to given file number, it means the word is common. Then it prints the word.

```
def read(path):
    f = open(path,"r")
    text=f.read()
    return text

def text_to_word(text):
    punctuations = '''!()-[]{};:'"\,<>./?@#$%^&*_~'''
    no_punct = ""
    for char in text:
       if char not in punctuations:
           no_punct = no_punct + char
       else:
           no_punct=no_punct+" "
    no_punct=no_punct.lower()
    words=no_punct.split()
    return words
```

For preprocessing I used above functions.

```
def main():

    opt=input("If you want to find words and their indexes which are starting with given input, please en
    if opt=="1":
        path1=input("Please enter path:")
        if os.path.isdir(path1):
            files=os.listdir(path1)
            text_paths=[]

            for file in files:
                if file.endswith("txt"):
                    text_path=os.path.join(path1,file)
                    text_paths.append(text_path)
            while True:
                inpu=input("If you want to finish entering please type exit. Enter the prefix: ")
                inpu=inpu.lower()

                if inpu!="exit":
                    for text_file in text_paths:
                        word_index=0
                        text=read(text_file)
                        keys=text_to_word(text)
                        file_name=os.path.split(text_file)[1]
                        t = Trie()
                        for key in keys:
                            t.insert(key,word_index,file_name)
                            word_index=word_index+len(key)
                        t.search(inpu)
                else:
                    break
        else:
            print("Directory does not exist")
```

This is the main function which drives the program.

Firstly I get input from user as “1” or ”2” . If input is “1”, program runs the first query. Then I get the directory from user as input. Then I keep txt files under given directory, in “text_paths”. Then I get the prefix from user as input. Then I insert each word of txt file to tries. I create different tries for different files. Then “search” function prints the found words.

```
    elif opt=="2":
        #path al
        path2=input("Please enter path:")
        if os.path.isdir(path2):
            files2=[]
            while True:
                inp=input("If you want to finish entering please type exit. Enter file name: ")
                if inp != "exit":
                    files2.append(inp)
                else:
                    break

            t = Trie()
            word_index=0
            arr=[]
            text_paths2=[]
            for file in files2:
                if file.endswith("txt"):
                    text_path=os.path.join(path2,file)
                    text_paths2.append(text_path)
            for text_file in text_paths2:
                text=read(text_file)
                keys=text_to_word(text)
                file_name=os.path.split(text_file)[1]
                for key in keys:
                    t.insert(key,word_index,file_name)
                    word_index=word_index+len(key)
            t.recursive_common(t.root,arr,len(text_paths2))
        else:
            print("Directory does not exist")
    else:
        print("Invalid input")
if __name__ == '__main__':
    main()
```

If input is “2”, the program runs second query. The difference from first query there is one trie created. And all of files inserted to this trie.

Sample Inputs and Outputs:

Inputs are sequntial.

First Query:

Inputs: “1”, “C:\Users\Celal\Desktop\advance data”, “ips”

```
In [7]: runfile('C:/Users/Celal/Desktop/advance data/hw1_2.py', wdir='C:/Users/Celal/Desktop/advance data')

If you want to find words and their indexes which are starting with given input, please enter 1. If you want to find common words, please enter 2:
1

Please enter path:C:\Users\Celal\Desktop\advance data

If you want to finish entering please type exit. Enter the prefix: ips
ipsum
index: [5, 270, 497]
file name: ['file1.txt']
------------------------------------
ipsumsum
index: [585]
file name: ['file1.txt']
------------------------------------
ipsumsummmmm
index: [593]
file name: ['file1.txt']
------------------------------------
ipsum
index: [133]
file name: ['file2.txt']
------------------------------------
ipsum
index: [132]
file name: ['file3.txt']
------------------------------------
ipsum
index: [736, 741, 746]
file name: ['file4.txt']
------------------------------------
```


Inputs: “11”

```
If you want to finish entering please type exit. Enter the prefix: 11
111
index: [107]
file name: ['file10.txt']
------------------------------------
112
index: [220]
file name: ['file10.txt']
------------------------------------
113
index: [439]
file name: ['file10.txt']
------------------------------------
114
index: [620]
file name: ['file10.txt']
------------------------------------
115
index: [750]
file name: ['file10.txt']
------------------------------------
116
index: [772]
file name: ['file10.txt']
------------------------------------
111
index: [107]
file name: ['file9.txt']
------------------------------------
112
index: [220]
file name: ['file9.txt']
------------------------------------
```

Inputs: "LORE"

```
If you want to finish entering please type exit. Enter the prefix: LORE
lorem
index: [0, 58]
file name: ['file1.txt']
-------------------------------------
lorem
index: [214, 432, 505]
file name: ['file2.txt']
-------------------------------------
lorem
index: [240]
file name: ['file3.txt']
-------------------------------------
lorem
index: [615]
file name: ['file4.txt']
-------------------------------------
lorem
index: [0, 416, 454, 777, 943, 1401, 1854, 1967, 3757, 4127, 4977, 5194, 5278, 5423, 6215, 6229, 6900, 6964, 7409, 7491, 7509]
file name: ['file5.txt']
-------------------------------------
lorem
index: [25, 264, 821]
file name: ['file7.txt']
-------------------------------------
lorem
index: [421, 481, 517, 812]
file name: ['file8.txt']
-------------------------------------
```

Inputs: "Ğ", "ğ", "Ü", "ü", "Ç", "ç", "ş", "Ş"

```
If you want to finish entering please type exit. Enter the prefix: Ğ

If you want to finish entering please type exit. Enter the prefix: ğ

If you want to finish entering please type exit. Enter the prefix: Ü

If you want to finish entering please type exit. Enter the prefix: ü

If you want to finish entering please type exit. Enter the prefix: Ç

If you want to finish entering please type exit. Enter the prefix: ç

If you want to finish entering please type exit. Enter the prefix: ş

If you want to finish entering please type exit. Enter the prefix: Ş

If you want to finish entering please type exit. Enter the prefix:
```

Second Query:

Inputs: “2”, “C:\Users\Celal\Desktop\advance data”, “file1.txt”, “file2.txt”, “file3.txt”

```
If you want to find words and their indexes which are starting with given input, please enter 1. If you want to find common words, please enter 2:
2

Please enter path:C:\Users\Celal\Desktop\advance data

If you want to finish entering please type exit. Enter file name: file1.txt

If you want to finish entering please type exit. Enter file name: file2.txt

If you want to finish entering please type exit. Enter file name: file3.txt

If you want to finish entering please type exit. Enter file name: exit
ac
amet
consequat
dapibus
donec
egestas
eget
et
id
in
ipsum
lacus
lorem
malesuada
mattis
nisl
nulla
purus
sed
sit
```


Inputs: “2”, “C:\Users\Celal\Desktop\advance data”, “file10.txt”, “file9.txt”, “file8.txt” , “file7.txt” , “file5.txt”

```
If you want to find words and their indexes which are starting with given input, please enter 1. If you want to find common words, please enter 2:
2

Please enter path:C:\Users\Celal\Desktop\advance data

If you want to finish entering please type exit. Enter file name: file10.txt

If you want to finish entering please type exit. Enter file name: file9.txt

If you want to finish entering please type exit. Enter file name: file8.txt

If you want to finish entering please type exit. Enter file name: file7.txt

If you want to finish entering please type exit. Enter file name: file5.txt

If you want to finish entering please type exit. Enter file name: exit
a
at
in

In [6]:
```

Inputs: “2”, “C:\Users\Celal\Desktop\advance data”, “file1.txt”, “file2.txt”, “file3.txt” , “file4.txt” , “file5.txt”, “file6.txt”, “file7.txt”, “file8.txt”, “file9.txt”, “file10.txt”

```
If you want to find words and their indexes which are starting with given input, please enter 1. If you want to find common words, please enter 2:
2

Please enter path:C:\Users\Celal\Desktop\advance data

If you want to finish entering please type exit. Enter file name: file1.txt

If you want to finish entering please type exit. Enter file name: file2.txt

If you want to finish entering please type exit. Enter file name: file3.txt

If you want to finish entering please type exit. Enter file name: file4.txt

If you want to finish entering please type exit. Enter file name: file5.txt

If you want to finish entering please type exit. Enter file name: file6.txt

If you want to finish entering please type exit. Enter file name: file7.txt

If you want to finish entering please type exit. Enter file name: file8.txt

If you want to finish entering please type exit. Enter file name: file9.txt

If you want to finish entering please type exit. Enter file name: file10.txt

If you want to finish entering please type exit. Enter file name: exit
in

In [2]:
```

<u>Assumptions:</u>

First query reads all .txt files under given directory. I assume these files are the target files.

Second query reads .txt files only. And I assume given files are under the given directory and file names are correct.

When finding word indexes, I assume spaces and punctuations are none.