Algorytmy Tekstowe - Lab2 - Łukasz Sochacki

Zadanie 1

In [3]:

```
teststrings = ["bbbd", "aabbabd", "ababcd", "abcbccd"] #Testowe dane
```

In [4]:

```
with open('1997_714_head.txt', 'r', encoding='utf-8') as f:
    data = f.read()
```

In [5]:

```
data #Dane zaladowane z pliku
```

Out[5]:

```
'\n\n\n\nDz.U. z 1998 r. Nr 144, poz. 930\n                                                    \n
\n                                        \n                                                    \n
USTAWA\n                              z dnia 20 listopada 1998 r.\n
\n          o zryczałtowanym podatku dochodowym od niektórych przychodów\n
osiąganych przez osoby fizyczne\n                                                  \n
Rozdział 1\n                                    Przepisy ogólne\n
\n                                    Art. 1.\nUstawa reguluje opodatkowanie zryczałtowa
nym podatkiem dochodowym niektórych\nprzychodów (dochodów) osiąganych przez osoby fizyczn
e prowadzące pozarolniczą\ndziałalność gospodarczą oraz przez osoby duchowne.\n
\n                                    Art. 2.\n1. Osoby fizyczne osiągające przychody z
pozarolniczej działalności\n  gospodarczej opłacają zryczałtowany podatek dochodowy w for
mie:\n   1) ryczałtu od przychodów ewidencjonowanych,\n   2) karty podatkowej.\n2. Osoby
duchowne, prawnie uznanych wyznań, opłacają zryczałtowany podatek\n  dochodowy od przycho
dów osób duchownych.\n3. Wpływy z podatku dochodowego opłacanego w formie ryczałtu od prz
ychodów\n  ewidencjonowanych oraz zryczałtowanego podatku dochodowego od przychodów\n  os
ób duchownych stanowią dochód budżetu państwa.\n4. Wpływy z karty podatkowej stanowią doc
hody gmin.\n                                                  \n
Art. 3.\nPrzychodów (dochodów) opodatkowanych w formach zryczałtowanych nie łączy się z\n
przychodami (dochodami) z innych źródeł podlegającymi opodatkowaniu na\npodstawie ustawy
z dnia 26 lipca 1991 r. o podatku dochodowym od osób\nfizycznych (Dz. U. z 1993 r. Nr 90,
poz. 416 i Nr 134, poz. 646, z 1994 r. Nr\n43, poz. 163, Nr 90, poz. 419, Nr 113, poz. 54
7, Nr 123, poz. 602 i Nr 126,\npoz. 626, z 1995 r. Nr 5, poz. 25 i Nr 133, poz. 654, z 19
96 r. Nr 25, poz.\n113, Nr 87, poz. 395, Nr 137, poz. 638, Nr 147, poz. 686 i Nr 156, poz
. 776, z\n1997 r. Nr 28, poz. 153, Nr 30, poz. 164, Nr 71, poz. 449, Nr 85, poz. 538, Nr\
n96, poz. 592, Nr 121, poz. 770, Nr 123, poz. 776, Nr 137, poz. 926, Nr 139,\npoz. 932-93
4 i Nr 141, poz. 943 i 945 oraz z 1998 r. Nr 66, poz. 430, Nr 74,\npoz. 471, Nr 108, poz.
685 i Nr 117, poz. 756), zwanej dalej "ustawą o podatku\ndochodowym".\n'
```

Zadanie 2

In [6]:

```
def uniqueSign(text): #Funkcja okreslajaca, czy dany tekst konczy sie na unikalnym znaku.
Jesli nie, to dodaje taki, by zapewnic unikalnosc
    letters = set()
    n = len(text)
    for i in range(n-1):
        letters.add(text[i])

    if not text[n-1] in letters:
        return text
    else:
        return text + '\0'
```

Zadanie 3

In [7]:

```
from queue import LifoQueue
```

In [8]:

```
class Node:

    def __init__(self):
        self.letter = None
        self.parent = None
        self.link = None
        self.children = dict()
        self.depth = 0

    def graft(self, text, sibling=None):
        for current_letter in text:
            if current_letter not in self.children:
                self.children[current_letter] = Node()
            self = self.children[current_letter]
            if sibling:
                sibling = sibling.children[current_letter]
                sibling.link = self
        return self

    def hasItem(self, item):
        if isinstance(item, str):
            while item != "":
                if item[0] in self.children.keys():
                    self, item = self.children[item[0]], item[1:]
                else:
                    return False
            return True
        return False
```

In [9]:

```
class Trie:
    def __init__(self, text):
        self.root = Node()
        leaf = self.root.graft(text)
        self.root.children[text[0]].link = self.root
        for i in range(len(text)):
            if i == 0:
                continue
            else:
                (head, sibling) = self.up_link_down(leaf)
                if not head:
                    head = self.root
                    sibling = head.children[text[i - 1]]
                    sibling.link = head
                leaf = head.graft(text[(i + head.depth):], sibling)

    @staticmethod
    def up_link_down(sibling: Node):
        letters = LifoQueue()
        while sibling and not sibling.link:
            letters.put(sibling.letter)
            sibling = sibling.parent
        if not sibling:
            return None, None
        node = sibling.link
        while not letters.empty():
            current_letter = letters.get()
            if current_letter in node.children.keys():
                node = node.children[current_letter]
                sibling = sibling.children[current_letter]
                sibling.link = node
```

```
            else:
                break
        return node, sibling

    def hasItem(self, item):
        if isinstance(item, str):
            if self.root.hasItem(item):
                return True
            else:
                return False
        return False
```

Zadanie 5

In [10]:

```
def isCorrect(algorithm, text): #Funkcja sprawdzajaca, czy struktura jest poprawnie utwor
zona
    for i in range(len(text)):
        if not algorithm.hasItem(text[i:]):
            return False
    return True
```

In [11]:

```
isCorrect(Trie(teststrings[0]),teststrings[0])
```

Out[11]:

```
True
```

In [12]:

```
isCorrect(Trie(teststrings[1]),teststrings[1])
```

Out[12]:

```
True
```

In [13]:

```
isCorrect(Trie(teststrings[2]),teststrings[2])
```

Out[13]:

```
True
```

In [14]:

```
isCorrect(Trie(teststrings[3]),teststrings[3])
```

Out[14]:

```
True
```

In [15]:

```
data = uniqueSign(data)
isCorrect(Trie(data),data)
```

Out[15]:

```
True
```

Zadanie 6/7

In [16]:

```
import time
```

In [17]:

```
def getTime(algorithm, text): #Funkcja do mierzenia czasów
    start = time.time()
    algorithm(text)
    end = time.time()
    return end - start
```

In [30]:

```
print(str(getTime(Trie, teststrings[0])) + ' s')
```

```
0.0010051727294921875 s
```

In [40]:

```
print(str(getTime(Trie, teststrings[1])) + ' s')
```

```
0.00099945068359375 s
```

In [42]:

```
print(str(getTime(Trie, teststrings[2])) + ' s')
```

```
0.0010027885437011719 s
```

In [41]:

```
print(str(getTime(Trie, teststrings[3])) + ' s')
```

```
0.0010285377502441406 s
```

In [45]:

```
print(str(getTime(Trie, data)) + ' s')
```

```
9.011034727096558 s
```

In [ ]:

```

```