Algorytmy Tekstowe

Laboratorium 2 - Drzewo trie i sufiksów

24.03.2021

Magdalena Pastuła

Zadanie 3.

Implementacja struktury trie:

```
class TrieNode:
    list_of_children = []
    node_value = ''

    def __init__(self, value):
        self.node_value = value
        self.list_of_children = []

    def addChild(self, next_child):
        self.list_of_children.append(next_child)

    def addSuffix(self, text):
        if len(text) == 0:
            return
        new_child = TrieNode(text[0])
        self.list_of_children.append(new_child)
        new_child.addSuffix(text[1:])

    def findHead(self, text):
        if len(text) == 0:
            return None, text
        for child in self.list_of_children:
            if child.node_value == text[0]:
                return child.findHead(text[1:])
        return self, text

    def printChildren(self, level):
        for child in self.list_of_children:
            print("level ", level, " ", child.node_value)
            child.printChildren(level+1)


def build_trie(text):
    trie = TrieNode("root")

    # for every suffix
    for i in reversed(range(len(text))):
        suffix = text[i:]
        # check if trie has that suffix
        suffix_head, suffix_to_add = trie.findHead(suffix)
```

```
            if suffix_head is not None:
                suffix_head.addSuffix(suffix_to_add)

        return trie
```

Zadanie 6.1

Zmierzone czasy budowania drzewa trie dla poszczególnych łańcuchów znaków:

Łańcuch znaków	Zmierzony czas budowania drzewa trie
bbbd	38.5 ms
aabbabd	52.99 ms
ababcd	40.7 ms
abcbccd	52 ms
treść ustawy	31.459 s

przy czym w przypadku treści ustawy algorytm uruchomiono dla łańcucha złożonego z pierwszy 3 200 znaków pliku, dla większych łańcuchów program zwraca błąd związany z pamięcią (dokładniej -1073741571 (0xC00000FD)).