Dokumentacja Egzaminu Praktycznego Algorytmy i Struktury Danych

Konrad Lubera, gr IB

21 maja 2020

Spis treści

1 Polecenie

Temat 9: Napisz program/aplikację który/która: Wyznaczy listę słów, które pojawiają się w tekście z jednego pliku, a nie pojawiają się w tekście z pliku drugiego przy pomocy algorytmu Boyer-Moore.

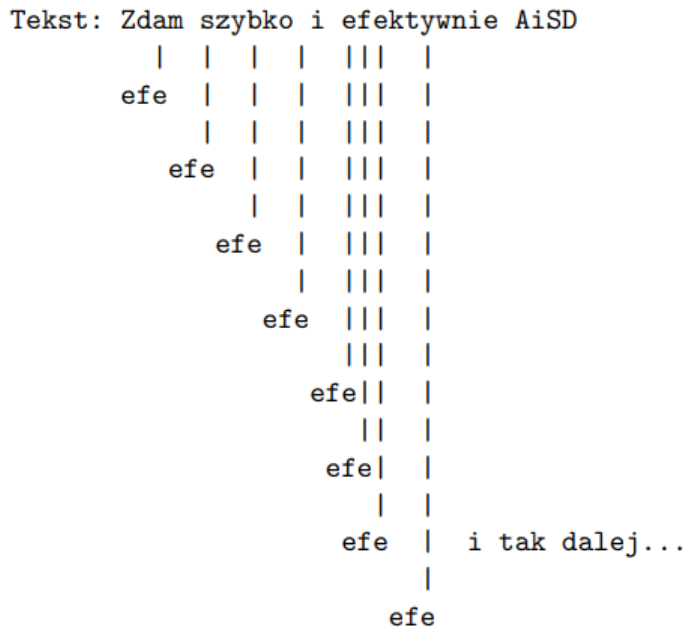
2 Algorytm

2.1 Opis

Do wykonania zadania użyto, zgodnie z poleceniem algorytmu Boyera-Moorea. Jest to wersja która korzysta z funkcji bad-character shift, bardzo podobna do tej która zastała zaprezentowana na wykładzie. Polega na porównywaniu z prawej strony wzorca, jeżeli wzorzec nie będzie pasował, to algorytm przeskakuje o długość wzorca. Jest to bardzo szybki i efektywny algorytm ponieważ ilość porównań jest relatywnie mała.

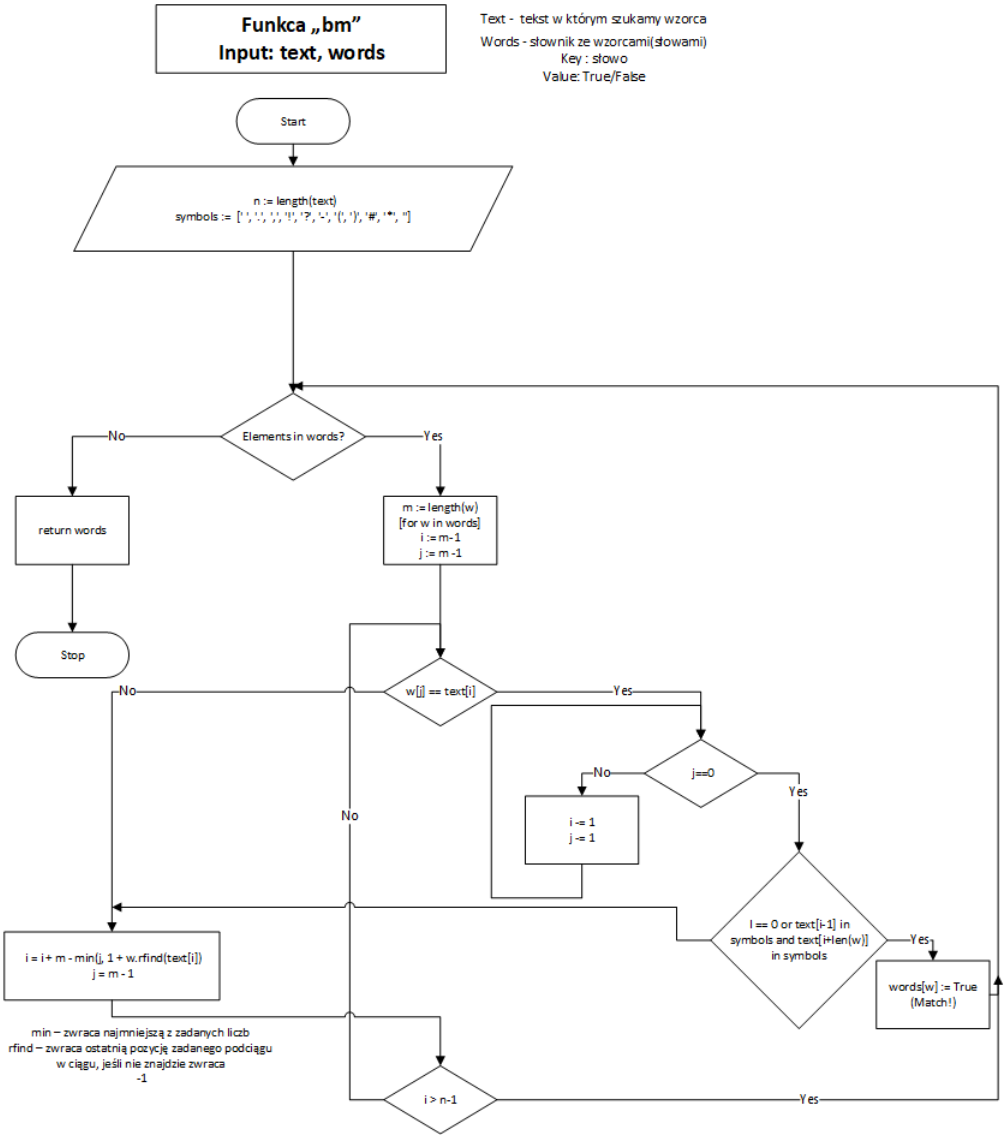
Implementację wykonano w języku Python w wersji 3.8.

2.2 Przykład działania



Rysunek 1: Przykład działania algorytmu z notatek z wykładów

2.3 Schemat blokowy



Rysunek 2: Zastosowany algorytm Boyera Moorea

3 Opis rozwiązania

3.1 Napotkane problemy

Podczas implementacji napotkano problem związany z tym, że algorytm służy, w jego oryginalnym wydaniu, do wyszukiwania fraz-wzorca co generuje problemy w wyszukiwaniu słów. Dla przykładu gdy naszym wzorcem jest słowo „dam" a tekście mamy słowo „zdam" algorytm uzna, że wzorzec się pojawił. Problem ten rozwiązano dojadając warunek który sprawdza czy znak przed i za dopasowaniem znajduje się na liście która zawiera znaki interpunkcyjne, często używane symbole oraz spację. Dodatkowo aby wyeliminować problem z dopasowanie słowa napisanego wielką literą i tego napisanego małą, wszystkie litery we wzorcu jak i tekście zostały zmienione na małe przy pomocy funkcji lower().

3.2 Pseudokod

input : *path*
output: string *text*

open file on *path* as $file$:
  $text = file$.read
$text$ = lower($text$)
$file$.close
return *text*

Algorithm 1: Funkcja read to string

input : *path*
output: dictionary *words*

words:=
symbols := '.,!-?():'
open file on *path* as $file$:
  foreach *line in* $file$ **do**
    foreach *word in line* **do**
      tmp:=*word*.strip(*symbols*)
      tmp = lower(*tmp*)
      if *tmp not in words and tmp!=''* **then**
        words[*tmp*] = False ; `/* przypisanie do słownika */`
      end
    end
end
return *words*

Algorithm 2: Funkcja read to dictionary

```
input  : string text, dictionary words
output: updated dictionary words

n := lenght(text)
symbols := [' ', '.', ',', '!', '?', '-', '(', ')', '*', '']
foreach w in words do
    m := lenght(w)
    j := m-1
    i := m-1
    repeat
        if w[j]==text[i] then
            if j==0 then
                if i==0 or text[i − 1]insymbols and
                 text[i + len(w)]insymbols then
                    words[w] = True
                    break
                else
                    i = i + m - min(j, 1+ w.find(text[i]))
                    j = m-1
                end
            else
                i -= 1
                j -= 1
            end
        else
            i = i + m - min(j, 1+ w.find(text[i]))
            j = m-1
        end
    until i>n-1;
end
return words
```

Algorithm 3: Funkcja BM

Pseudokod został napisany tylko dla funkcji które są ważne z punktu widzenia działania samego algorytmu.

4 Prezentacja wyników

Teksty które zostały użyte do przetestowania programu zostały tak dobrane żeby sprawdzić czy problemy o których napisano w sekcji Algorytm - Opis zostały rozwiązane w rezultacie teksty mają dużo znaków interpunkcyjnych oraz słów które zawierają w sobie inne słowa. Teksy nie jest też zbyt długie żeby nie generować trudności podczas sprawdzania wyników, jednak przetestowano program używając dłuższych tekstów i też działa :D.

Oto teksty:

- file1.txt

  > Byliśmy bardzo smutni, że nie mogliśmy wrócić na uczelnię, jednak najważniejsze było żeby przetrwać ten trudny czas pandemii i co najważniejsze dobrze napisać egzamin z algorytmów i struktur danych. Mam nadzieję ze to co zrobiłem wystarczy żebym zdał. Jeżeli nie będę bardzo zawiedziony! Kompletnie, nie, mam, pojęcia, jak, używać, znaków, interpunkcyjnych, więc, w, tym, zdaniu, przecinek, będzie, po, każdym, słowie. Umiejętności miękkie to nie jest moja mocna strona. Już nie wiem co tu napisać więc wkleję pierwszy akapit z Wikipedii o algorytmie Boyera Moore'a. Algorytm Boyera i Moore'a algorytm poszukiwania wzorca w tekście. Polega na porównywaniu, zaczynając od ostatniego elementu wzorca.

- file2.txt

  > Byliśmy bardzo zawiedzeni, że w tym semestrze nie wrócimy już na uczelnię, jednak pomimo tego ucze (specjalnie błąd) się pilnie i dam z siebie wszystko. Mam nadzieję, że taki stan rzeczy nie będzie trwać w nieskończoność. Są ważniejsze sprawy od spotkania z kolegami np. egzamin z aisd który mimo trudności się obył w bardzo przystępnej formie. Nie będę tu już więcej pisać jakichś głupot tylko wkleję znów akapit z Wikipedii i kilka słów z generatora losowych słów ów. Algorytm Boyera i Moore'a algorytm poszukiwania wzorca w tekście. Polega na porównywaniu, zaczynając od ostatniego elementu wzorca. Rzeczownik, przestrzeń, biblioteka, rosa parks, szubienica, ośmiornica, wzmacniacz, atletyczny, farmaceuta, jeżozwierz, frustracja, bezradność, anne frank, niedźwiedź, rękawiczki, kierownica, studzienka, dominikana, ciężarówka, sprzedawca.

To lista słów które pojawiają się w tekście z jednego pliku, a nie pojawiają się w tekście z pliku drugiego:

1. Zawiedzeni
2. Semestrze
3. Wrócimy
4. Pomimo
5. Tego
6. Ucze
7. Specjalnie
8. Błąd
9. Się
10. Pilnie
11. Dam
12. Siebie
13. Wszystko
14. Taki
15. Stan
16. Rzeczy
17. Trwać
18. Nieskończoność
19. Są
20. Ważniejsze
21. Sprawy
22. Spotkania
23. Kolegami
24. Np
25. Aisd
26. Który
27. Mimo
28. Trudności
29. Obył
30. Przystępnej
31. Formie
32. Więcej
33. Pisać
34. Jakichś
35. Głupot
36. Tylko
37. Znów
38. Kilka
39. Słów
40. Generatora
41. Losowych
42. Ów
43. Rzeczownik
44. Przestrzeń
45. Biblioteka
46. Rosa
47. Parks
48. Szubienica
49. Ośmiornica
50. Wzmacniacz
51. Atletyczny
52. Farmaceuta
53. Jeżozwierz
54. Frustracja
55. Bezradność
56. Anne
57. Frank
58. Niedźwiedź
59. Rękawiczki
60. Kierownica
61. Studzienka
62. Dominikana
63. Ciężarówka
64. Sprzedawca
65. Przeżyć
66. Samochód

Po wykonaniu programu otworzy się plik txt który zawiera ta listę oraz dodatkową statystykę. Jeżeli to nie nastąpi np gdy używamy systemu Linux wystarczy otworzyć plik „results.txt". Istnieje możliwość zmiany tekstów na których operuje program wystarczy wyedytować zawartość plików „file1.txt" i „file2.txt".

5 Kod programu

```
# -*- coding: utf-8 -*-
# Wyznaczy ęlist łósw, óktre ąpojawiaj ęsi w śtekcie z
    jednego pliku, a nie ąpojawiaj ęsi w śtekcie z pliku
    drugiego przy pomocy algorytmu Boyer-Moore
import os


def read_to_string(path):
    with open(path, 'r', encoding="utf-8") as file:
        text = file.read()
    text = text.lower()
    text += "␣" # dodatkowe zabezpieczenie
    #print(text)
    file.close()
    return text


def read_to_dict(path):
    words = {}
    symbols = '␣.,!-?()"":\n␣' # warunki dla - w
       środku ! - strip łzaatwia sprwawe
    with open(path, 'r', encoding="utf-8") as file:
        for line in file:
            for word in line.split():
                tmp = word.strip(symbols)
                tmp = tmp.lower()
                # TYMCZASOWE ĄROZWIZANIE!
                #t = ' ' + tmp + ' '
                if tmp not in words and tmp != "":
                    words[tmp] = False
    # print("śćIlo łósw bez ópowtrze: " + str(len(words
       )))
```

```
    #print(words.keys())
    file.close()
    return words


def bm(text, words):
    n = len(text)
    symbols = [' ', '.', ',', '!', '?', '-', '(', ')',
        '#', '*', '']
    for w in words:
        m = len(w)
        i = m - 1
        j = m - 1
        while True:
            if w[j] == text[i]:
                if j == 0:
                    if i == 0 or text[i-1] in symbols
                        and text[i+len(w)] in symbols:
                            words[w] = True
                            # print("Matched word: '" + w +
                                "' on position: " + str(i))
                            # if i == 0:  print("Before:
                                None")
                            # else: print("Before: " + text
                                [i-1])
                            # print("After: " + text[i +
                                len(w)])
                            break
                    else:
                        i = i + m - min(j, 1 + w.rfind(
                            text[i]))
                        #i = i + m - min(j, 1 + w.rfind
                            (text[i+len(w)]))
                        j = m - 1
                else:
                    i -= 1
                    j -= 1
            else:
                i = i + m - min(j, 1 + w.rfind(text[i])
                    )
                j = m - 1
```

```
            if i > n - 1:
                break
    return words


def count_words(path):
    words = []
    symbols = ' .,!-?()"":\n '
    with open(path, 'r', encoding="utf-8") as file:
        for line in file:
            for word in line.split():
                tmp = word.strip(symbols)
                tmp = tmp.lower()
                if tmp not in words and tmp != "":
                    words.append(tmp)
    file.close()
    return len(words)


def results(words_1, num_of_word_2):
    i = 0
    j = 0
    for w in words_1:
        if words_1[w]:
            j += 1
        elif not words_1[w]:
            # print(w)
            i += 1
    c = int(num_of_word_2) - j

    # print("ślIlo łósw w pliku 1. (bez ńpowtrze): " +
      str(len(words_1)))
    # print("ślIlo łósw w pliku 2. (bez ńpowtrze): " +
      str(num_of_word_2))
    # print("ślIlo łósw unikalne dla pliku 1: " + str(i)
      )
    # print("ślIlo łósw unikalne dla pliku 2: " + str(c)
      )
    # print("ślIlo łósw z pliku 1 znalezionych w pliku
      2: " + str(j))
```

```
    save_results(words_1, num_of_word_2, i, c, j)


def save_results(words, n2, unique_1, unique_2, matches
    ):
    file = open("results.txt", "w")

    file.write("Egzamin␣Praktyczny␣AiSD␣Konrad␣Lubera␣\
        n")
    file.write("Temat:␣Wyznaczy␣ęlist␣łósw,␣óktre␣
        ąpojawiaj␣ęsi␣w␣śtekcie␣z␣jednego␣pliku,␣a␣nie␣
        ąpojawiaj␣ęsi␣w␣śtekcie␣z␣pliku␣drugiego␣przy␣
        pomocy␣algorytmu␣Boyer-Moore\n")
    file.write("
        ----------\n")

    file.write("Wyniki:␣\n")
    file.write("---------\n")
    file.write("śćIlo␣łósw␣w␣pliku␣1.␣(bez␣ółpowtrze):␣"
        + str(len(words)) + "\n")
    file.write("śćIlo␣łósw␣w␣pliku␣2.␣(bez␣ółpowtrze):␣"
        + str(n2) + "\n")
    file.write("---------\n")
    file.write("łSowa␣unikalne␣dla␣pliku␣1:␣" + str(
        unique_1) + "\n")
    file.write("łSowa␣unikalne␣dla␣pliku␣2:␣" + str(
        unique_2) + "\n")
    file.write("---------\n")
    file.write("śćIlo␣łósw␣z␣pliku␣1␣znalezionych␣w␣
        pliku␣2:␣" + str(matches) + "\n")
    file.write("
        ----------\n")

    file.write(
        "łSowa␣óktre␣ąpojawiaj␣ęsi␣w␣śtekcie␣z␣jednego␣
            pliku,␣a␣nie␣ąpojawiaj␣ęsi␣w␣śtekcie␣z␣pliku␣
            drugiego:␣\n")
    i = 1
    for w in words:
        if not words[w]:
            file.write(str(i) + ".␣" + str(w.strip().
```

```
                capitalize()) + "\n")
            i += 1

    file.close()


results(bm(read_to_string("file1.txt"), read_to_dict("
    file2.txt")), count_words("file1.txt"))
os.system('start notepad results.txt')
```