Hubert Makowski

# Sprawozdanie z 1 zadania laboratoryjnego z przedmiotu Teoria informacji i kodowania.

## Treść zadania:

Implementacja kodera/dekodera słownikowego. Metoda: LZ77

## Analiza zadania

Algorytm LZ77 jest to metoda strumieniowej słownikowej kompresji danych, polega na skracaniu zapisów tekstowych poprzez zastępowanie słów wskazanymi na odpowiednie hasła słownika.

Metoda LZ77 opiera się na założeniu, że słowa w kodowanym tekście powtarzają się. Słownik stanowi skończony ciąg ostatnio kodowanych symboli. Ciąg interpretujemy jako zbiór haseł, czyli dowolnych podciągów symboli słownika.

## Założenia

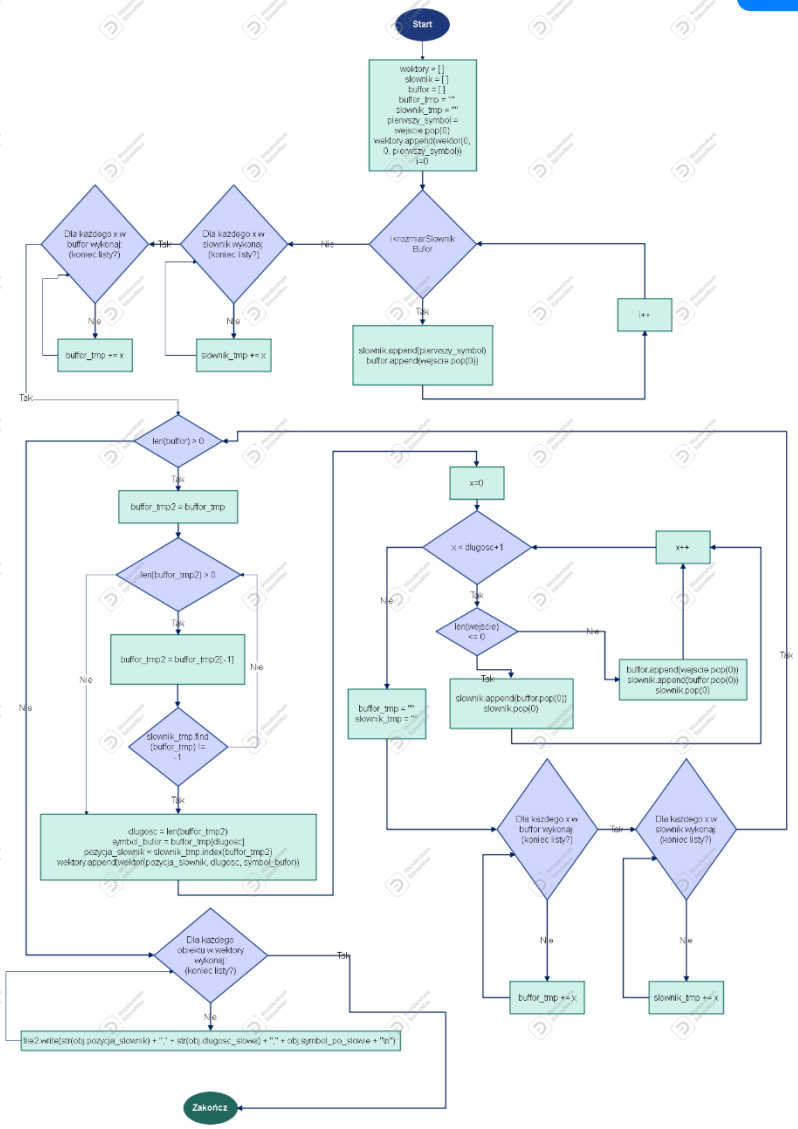
W zmiennej bufor zapisujemy aktualne kodowane symbole, zmienna wejście zawiera przeznaczona do kodowania sekwencje symboli wejściowych, słownik zawiera ostatnio kodowane symbole. Rozmiar bufora i słownika jest taki sam, a także jest on definiowany na ilość znaków w pliku wejściowym – 2.

Proces kodowania rozpoczyna się od inicjowania słownika. Następnie zmiennej słownik przypisuje się wartość początkowa sekwencje złożona z powtórzeń pierwszego kodowanego symbolu. Następnie symbol przesyłamy do dekodera, aby mógł on zainicjować słownik. W tym celu generujemy wektor kodowy w postaci <0, 0, s> gdzie s jest pierwszym elementem wejścia.

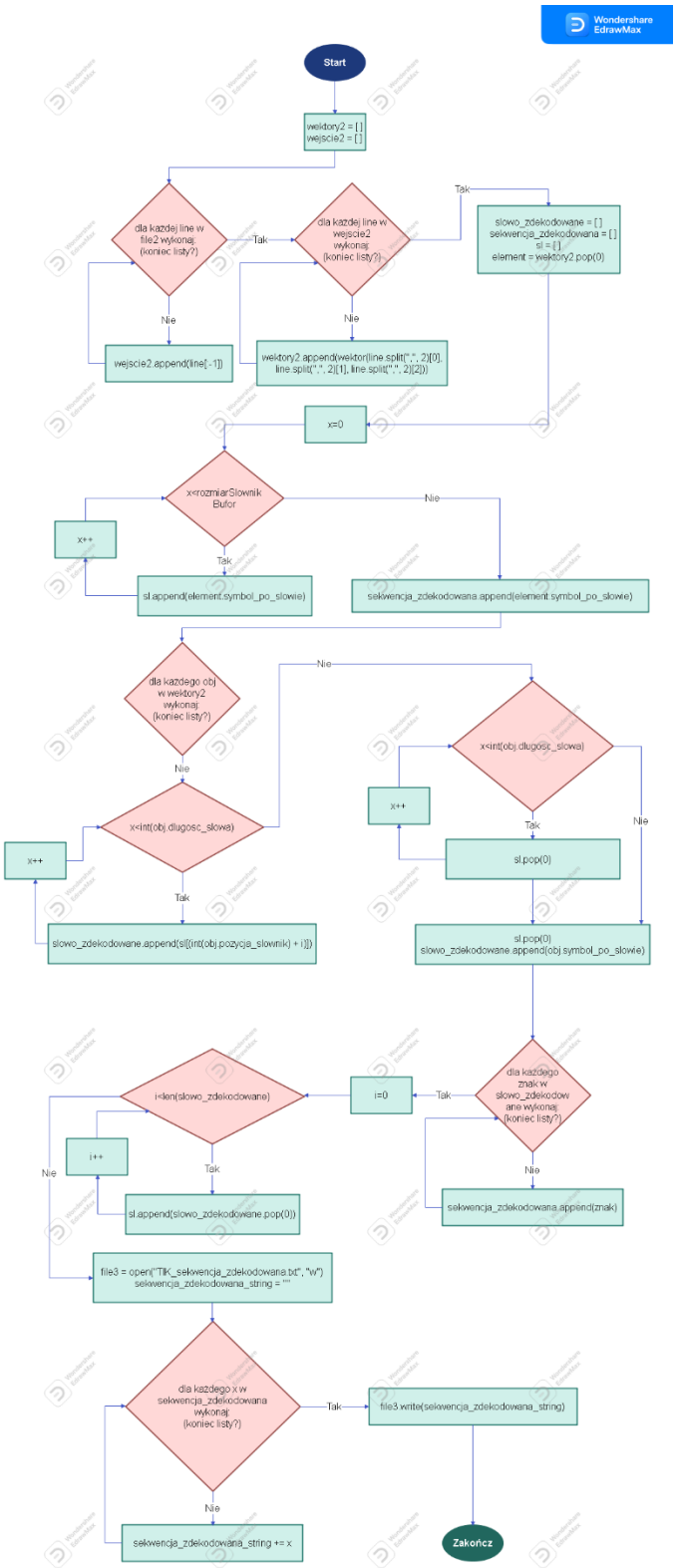
## Algorytm

Schematy blokowe wykonałem w programie WondershareEdrawMax (darmowa licencja próbna, znak wodny).

Schemat blokowy kodera



Schemat blokowy dekodera

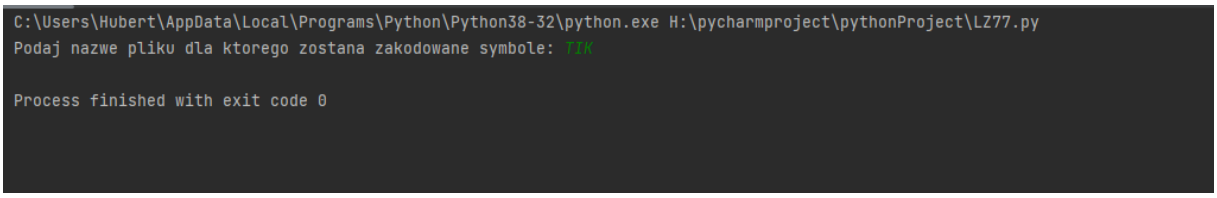


## Implementacja

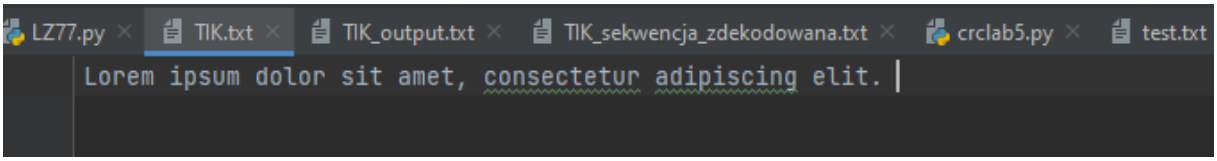
Początkowo algorytm chciałem implementować w języku c++, jednak ze względu na swoje ograniczenia związane z tym językiem, postanowiłem napisać kod w języku python. Korzystałem ze środowiska PyCharm community edition. Użytkownik podaje nazwę pliku wejściowego w terminalu (bez rozszerzenia .txt), nazwa pliku w którym zostaną zapisane wektory kodowe jest definiowana na TIK_output.txt, nazwa pliku do którego zostanie zapisana sekwencja zdekodowana jest zdefiniowana na TIK_sekwencja_zdekodowana.txt. Format zapisu wektorów kodowych ma postać -pozycja słowa w słowniku, długość słowa, symbol po słowie- (dane w pliku są oddzielone przecinkiem)

## Testowanie poprawności działania

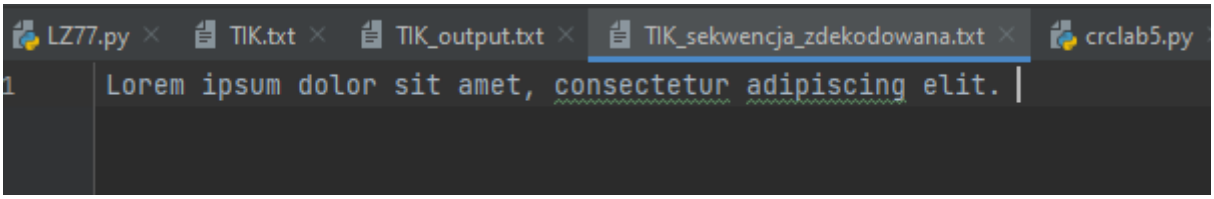
Podgląd terminala



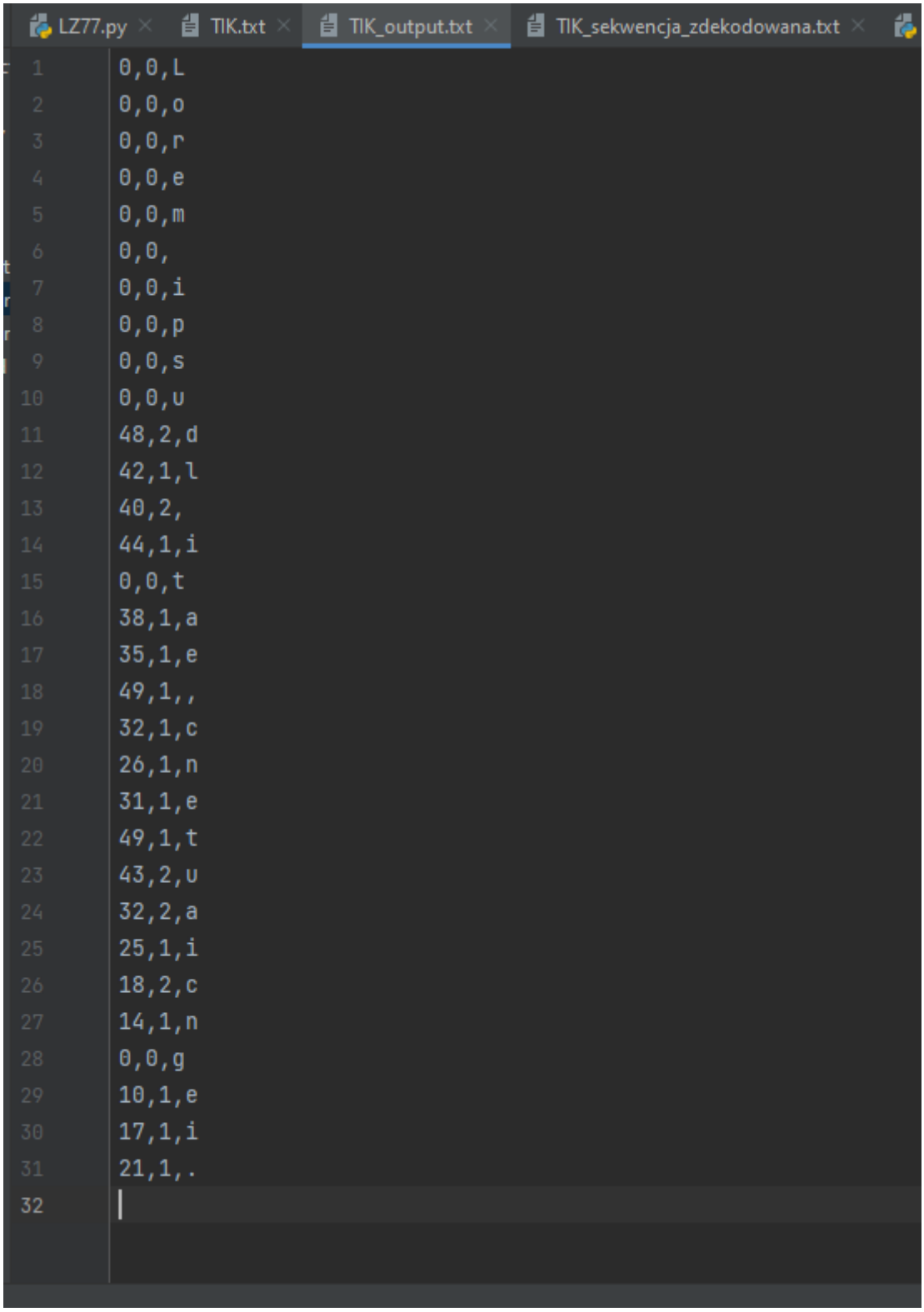

Plik wejściowy TIK.txt




Plik z sekwencja zdekodowana TIK_sekwencja_zdekodowana.txt



Plik z zapisanymi wektorami kodowymi TIK_output.txt

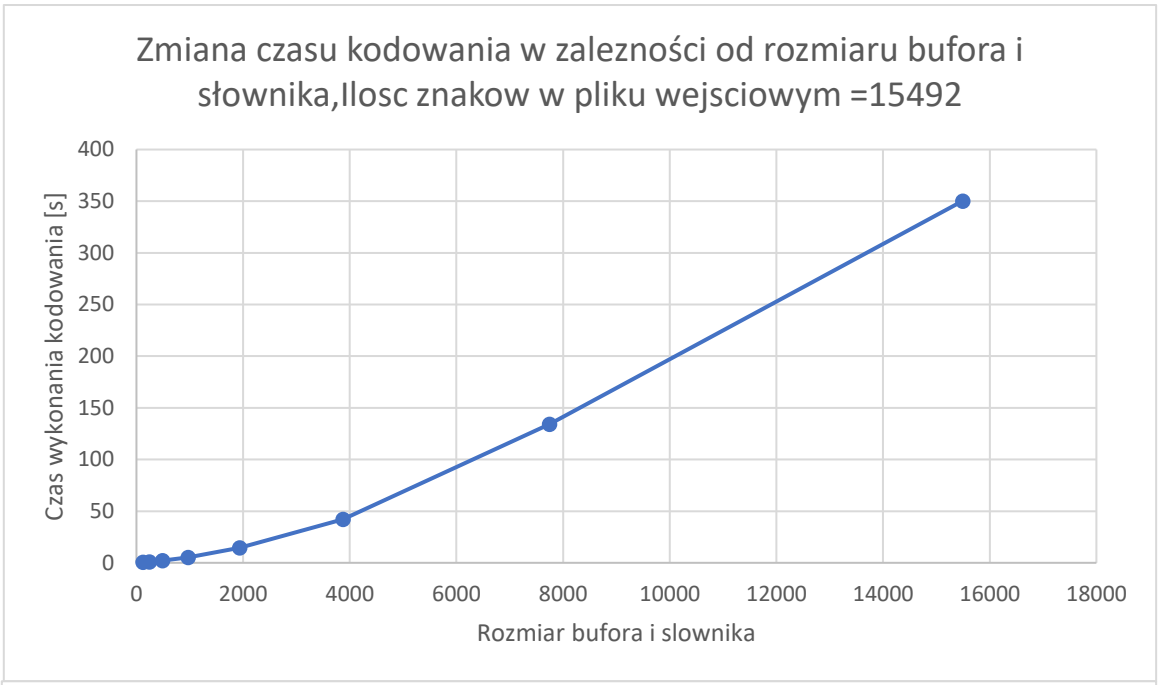


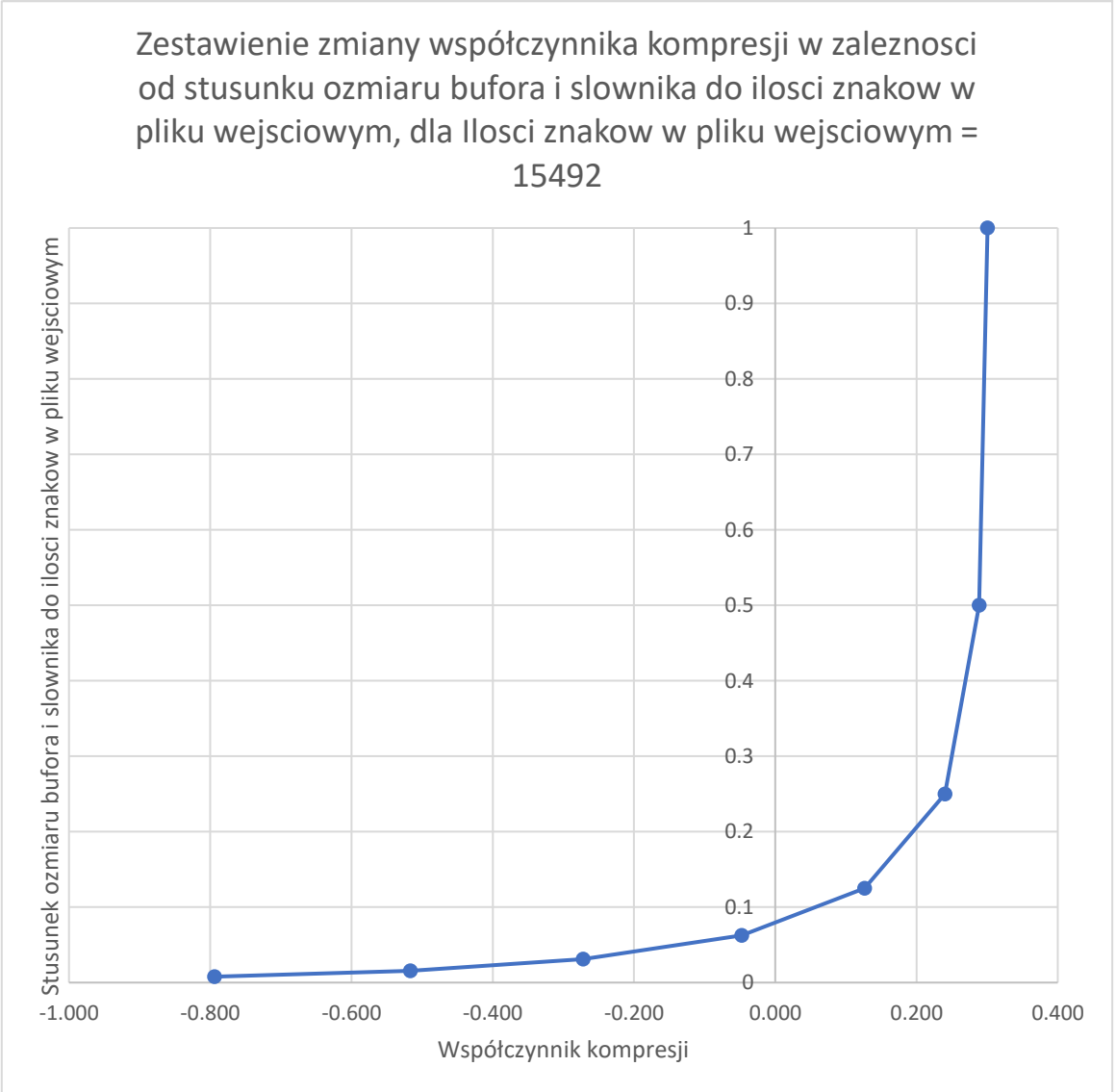
## Testowanie stopnia kompresji

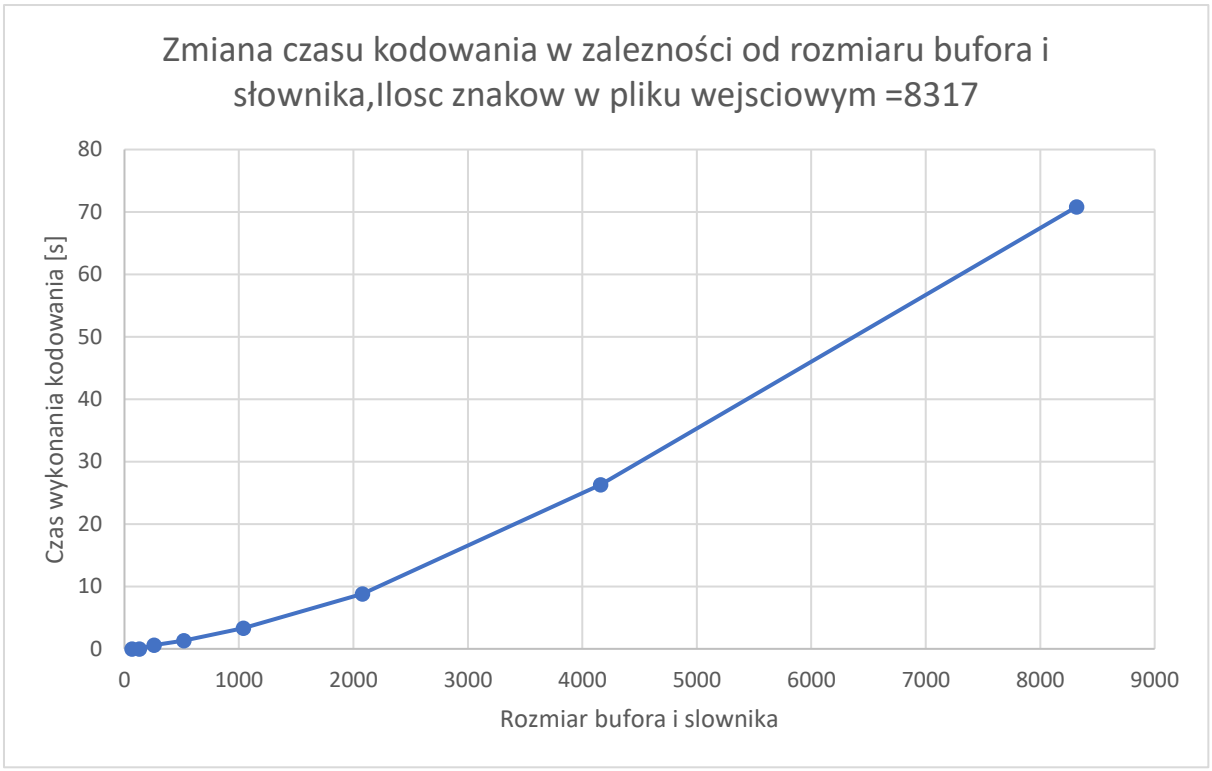
Wzór współczynnika kompresji-(( Ilość znaków w pliku wejściowym- Ilość wektorów kodowych*5)/ Ilość znaków w pliku wejściowym), przy Ilości wektorów kodowych pojawia się współczynnik 5 ponieważ, format zapisu wektorów kodowych ma postać -pozycja słowa w słowniku, długość słowa, symbol po słowie- na co składa się 5 znaków.

Tabela 1. Zestawienie wyników otrzymanych dla różnej ilości znaków w pliku wejściowym

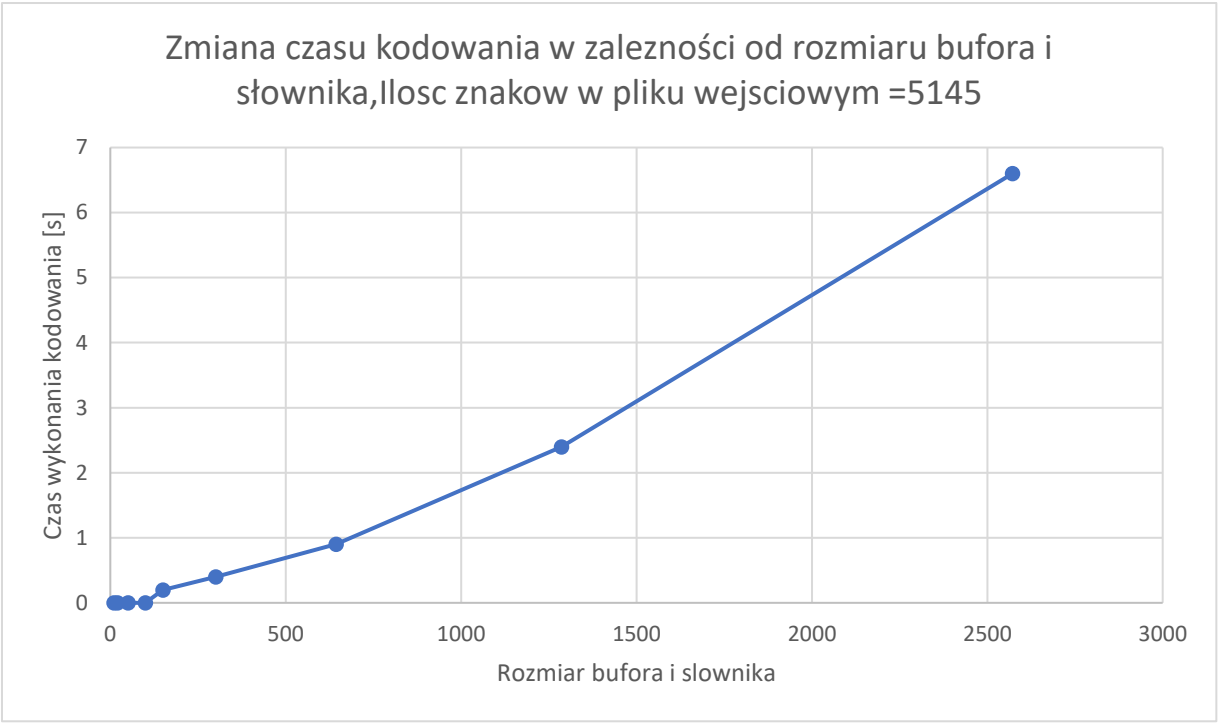
| Rozmiar bufora i słownika | Ilość znaków w pliku wejściowym | Ilość wektorów kodowych | Współczynnik kompresji | Czas wykonania kodowania [s] | Czas wykonania dekodowania [s] |
|---|---|---|---|---|---|
| 15491 | 15492 | 2167 | 0.301 | 350.2 | 0.03 |
| 7745 | 15492 | 2204 | 0.289 | 134.1 | 0.03 |
| 3872 | 15492 | 2354 | 0.240 | 42.1 | 0.03 |
| 1936 | 15492 | 2707 | 0.126 | 14.5 | 0.02 |
| 968 | 15492 | 3246 | -0.048 | 5.3 | 0.02 |
| 484 | 15492 | 3941 | -0.272 | 2.1 | 0.2 |
| 242 | 15492 | 4699 | -0.517 | 0.9 | 0.2 |
| 121 | 15492 | 5559 | -0.794 | 0.48 | 0.2 |
| 8316 | 8317 | 1130 | 0.321 | 70.8 | 0.02 |
| 4158 | 8317 | 1331 | 0.200 | 26.3 | 0.02 |
| 2079 | 8317 | 1465 | 0.119 | 8.8 | 0.02 |
| 1039 | 8317 | 1699 | -0.021 | 3.3 | 0.02 |
| 519 | 8317 | 2108 | -0.267 | 1.3 | 0.02 |
| 259 | 8317 | 2548 | -0.532 | 0.59 | 0.02 |
| 129 | 8317 | 2952 | -0.775 | 0 | 0 |
| 64 | 8317 | 3464 | -1.082 | 0 | 0 |
| 5144 | 5145 | 922 | 0.104 | 17 | 0.01 |
| 2572 | 5145 | 950 | 0.077 | 6.6 | 0.01 |
| 1286 | 5145 | 1089 | -0.058 | 2.4 | 0.01 |
| 643 | 5145 | 1273 | -0.237 | 0.9 | 0.01 |
| 300 | 5145 | 1514 | -0.471 | 0.4 | 0.01 |
| 150 | 5145 | 1787 | -0.737 | 0.2 | 0.01 |
| 100 | 5145 | 1948 | -0.893 | 0 | 0 |
| 50 | 5145 | 2270 | -1.206 | 0 | 0 |
| 20 | 5145 | 2790 | -1.711 | 0 | 0 |
| 10 | 5145 | 3330 | -2.236 | 0 | 0 |
| 1251 | 1252 | 335 | -0.338 | 0 | 0 |
| 625 | 1252 | 347 | -0.386 | 0 | 0 |
| 312 | 1252 | 379 | -0.514 | 0 | 0 |
| 156 | 1252 | 446 | -0.781 | 0 | 0 |
| 78 | 1252 | 499 | -0.993 | 0 | 0 |
| 39 | 1252 | 590 | -1.356 | 0 | 0 |
| 760 | 761 | 231 | -0.518 | 0 | 0 |
| 380 | 761 | 238 | -0.564 | 0 | 0 |
| 190 | 761 | 262 | -0.721 | 0 | 0 |
| 95 | 761 | 297 | -0.951 | 0 | 0 |
| 47 | 761 | 342 | -1.247 | 0 | 0 |
| 23 | 761 | 401 | -1.635 | 0 | 0 |
| 335 | 336 | 128 | -0.905 | 0 | 0 |
| 168 | 336 | 131 | -0.949 | 0 | 0 |
| 83 | 336 | 139 | -1.068 | 0 | 0 |
| 41 | 336 | 164 | -1.440 | 0 | 0 |
| 21 | 336 | 180 | -1.679 | 0 | 0 |


Zmiana czasu kodowania w zalezności od rozmiaru bufora i słownika,Ilosc znakow w pliku wejsciowym =15492
Czas wykonania kodowania [s]
400
350
300
250
200
150
100
50
0
0
2000
4000
6000
8000
10000
12000
14000
16000
18000
Rozmiar bufora i slownika


Zestawienie zmiany współczynnika kompresji w zaleznosci od stusunku ozmiaru bufora i slownika do ilosci znakow w pliku wejsciowym, dla Ilosci znakow w pliku wejsciowym = 15492
Stusunek ozmiaru bufora i slownika do ilosci znakow w pliku wejsciowym
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-1.000
-0.800
-0.600
-0.400
-0.200
0.000
0.200
0.400
Współczynnik kompresji


Zmiana czasu kodowania w zaleznośći od rozmiaru bufora i słownika,Ilosc znakow w pliku wejsciowym =8317
Czas wykonania kodowania [s]
80
70
60
50
40
30
20
10
0
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
Rozmiar bufora i slownika


Zestawienie zmiany współczynnika kompresji w zaleznosci od stusunku ozmiaru bufora i slownika do ilosci znakow w pliku wejsciowym, dla Ilosci znakow w pliku wejsciowym = 8317
Stusunek ozmiaru bufora i slownika do ilosci znakow w pliku wejsciowym
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-1.200
-1.000
-0.800
-0.600
-0.400
-0.200
0.000
0.200
0.400
Współczynnik kompresji


Zmiana czasu kodowania w zalezności od rozmiaru bufora i słownika,Ilosc znakow w pliku wejsciowym =5145
Czas wykonania kodowania [s]
0
1
2
3
4
5
6
7
0
500
1000
1500
2000
2500
3000
Rozmiar bufora i slownika


Zestawienie zmiany współczynnika kompresji w zaleznosci od stusunku ozmiaru bufora i slownika do ilosci znakow w pliku wejsciowym, dla Ilosci znakow w pliku wejsciowym = 5145
Stusunek ozmiaru bufora i slownika do ilosci znakow w pliku wejsciowym
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
-2.800
-2.300
-1.800
-1.300
-0.800
-0.300
0.200
0.700
Współczynnik kompresji

## Wnioski

Z tabeli a także wykresów wynika, że największy współczynnik kompresji występuję dla rozmiaru słownika i bufora bliskiego ilości znaków w pliku wejściowym. Co ważne współczynnik kompresji rośnie wraz z wzrostem wielkości rozmiaru słownika i bufora (jest proporcjonalny do wielkości rozmiaru słownika i bufora).

Z danych w tabeli, a także wykresów wynika również, że empiryczna maksymalna wartość współczynnika kompresji dla danego algorytmu wynosi około 0.3, współczynnik kompresji asymptotycznie dąży do wartości 0.3.

Warto również zwrócić uwagę na czas kodowania, który rośnie proporcjonalnie z rozmiarem słownika i bufora. Jest to spowodowane tym, iż algorytm poświęca więcej czasu na przeszukiwanie danej sekwencji w słowniku, ze względu na większą ilość znaków w ciągu, zarówno w słowniku jak i buforze (funkcja find w python).

Program nie radzi sobie z znakiem nowej linii w pliku wejściowym, ze względu na zapis wektorów kodowych, zatem postanowiłem usunąć znaki nowej linii z listy *wejście* i nie kodować znaków nowej linii.

Złożoność czasowa dekodowania szacuje na T(n)=**$O(n^2)$**

Złożoność czasowa kodowania  szacuje na T(n)=**$O(n^4)$** (funkcja find w python ma złożoność czasową T(n)=**$O(n^2)$**)

Wniosek końcowy – program nie sprawdzi się dla dużej ilości znaków w pliku wejściowym (czas wykonania programu jest nieakceptowalny).

## Kod źródłowy programu

```
#klasa wektora kodowego
class wektor:
    def __init__(self, pozycja_slownik, dlugosc_slowa, symbol_po_slowie):
        self.pozycja_slownik = pozycja_slownik
        self.dlugosc_slowa = dlugosc_slowa
        self.symbol_po_slowie = symbol_po_slowie

#funkcja dekodowania
def dekoduj(nazwaPliku, rozmiarSlownikBufor):
    try:
        file2 = open(nazwaPliku, "r") #plik z ktorego beda odczytywane
wektory kodowe
    except:
        print("Problem z otwarciem pliku")
        exit()
    wektory2 = [] #lista wektorow kodowych
    wejscie2 = [] #lista wejscia z pliku
    for line in file2: #czytaj sekwencje zakodowana z pliku
        wejscie2.append(line[:-1]) #usun znak nowej lini
    for line in wejscie2: #dodaj elementy do tablicy
        wektory2.append(wektor(line.split(",", 2)[0], line.split(",",
2)[1], line.split(",", 2)[2])) #rozgrupu po przecinku
    slowo_zdekodowane = []
    sekwencja_zdekodowana = []
    sl = [] #slownik
    element = wektory2.pop(0)  # wez pierwszy wektor do dekodowania
    for x in range(rozmiarSlownikBufor):  # wypelnij slownik
```

```
        sl.append(element.symbol_po_slowie)
    sekwencja_zdekodowana.append(element.symbol_po_slowie)
    for obj in wektory2:
        for i in range(int(obj.dlugosc_slowa)):
            slowo_zdekodowane.append(sl[(int(obj.pozycja_slownik) + i)])
#symbole ze slownika przenies do slowa zdekodowanego
        for i in range(int(obj.dlugosc_slowa)):  # usun elementy ze
slownika
            sl.pop(0)
        sl.pop(0)  # przesun slownik o 1 miejsce
        slowo_zdekodowane.append(obj.symbol_po_slowie) #dodaj do sekwencji
zdekodowanej symbol po slowie z wektora kodowego
        for znak in slowo_zdekodowane:
            sekwencja_zdekodowana.append(znak)  # dodaj elementy do
sekwencji zdekodowanej
        for i in range(len(slowo_zdekodowane)):
            sl.append(slowo_zdekodowane.pop(0))  # dodaj do slownika slowo
zdekodowane

    try:
        file3 = open("TIK_sekwencja_zdekodowana.txt", "w")  # zapisz
sekwencje zdekodowana do pliku
    except:
        print("Problem z otwarciem pliku")
        exit()
    sekwencja_zdekodowana_string = ""
    for x in sekwencja_zdekodowana:
        sekwencja_zdekodowana_string += x
    file3.write(sekwencja_zdekodowana_string)
    file2.close()
    file3.close()


#funkcja kodowania
def kodowanie(wejscie, rozmiarSlownikBufor):
    #dane podstawowe
    try:
        file2 = open("TIK_output.txt", "w")  # plik w ktorym zostana
zapisane wektory kodowe
    except:
        print("Problem z otwarciem pliku")
        exit()
    wektory = []
    slownik = []
    buffor = []
    buffor_tmp = ""
    slownik_tmp = ""
    pierwszy_symbol = wejscie.pop(0) #wez pierwszy element z wejscia
    wektory.append(wektor(0, 0, pierwszy_symbol)) #dodaj wektor kodowy do
listy
    for i in range(rozmiarSlownikBufor): #wypelnij slownik i bufor
pierwszym symbolem z pliku
        slownik.append(pierwszy_symbol)
        buffor.append(wejscie.pop(0))
    for x in slownik:
        slownik_tmp += x
    for x in buffor:
        buffor_tmp += x

    while len(buffor) > 0 : #wykonuj dopóki buffor > 0
        buffor_tmp2 = buffor_tmp
```

```
        while len(buffor_tmp2) > 0: #sprawdz czy istnieje sekwencja w
slowniku
            buffor_tmp2 = buffor_tmp2[:-1]  # zmniejsz rozmiar szukanej
sekwencji o 1 element
            if (slownik_tmp.find(buffor_tmp2)) != -1: #znaleziono sekwencje
w slowniku
                break
        dlugosc = len(buffor_tmp2) #dlugosc slowa
        symbol_bufor = buffor_tmp[dlugosc] #symbol po slowie
        pozycja_slownik = slownik_tmp.index(buffor_tmp2) #pozycja slowa w
slowniku
        wektory.append(wektor(pozycja_slownik, dlugosc, symbol_bufor))
#dodaj element do listy wektorow kodowych
        for x in range(dlugosc + 1): #przesun slownik bufor wejscie
            if (len(wejscie) <= 0):
                slownik.append(buffor.pop(0)) #usun element z bufora i
dodaj do slownika
                slownik.pop(0) #usun element ze slownika
            else:
                buffor.append(wejscie.pop(0)) #dodaj element na koniec
bufora
                slownik.append(buffor.pop(0)) #dodaj element na koniec
slownika z bufora
                slownik.pop(0) #usun element ze slownika
        buffor_tmp = ""
        slownik_tmp = ""
        for x in slownik: #wypelnij tymczasowny slownik
            slownik_tmp += x
        for y in buffor: #wypelnij tymczasowny bufor
            buffor_tmp += y
    for obj in wektory:
        file2.write(str(obj.pozycja_slownik) + "," + str(obj.dlugosc_slowa)
+ "," + obj.symbol_po_slowie + "\n")  # zapis do pliku
    file2.close()

#odczytywanie symboli z pliku
# 'main function'
file_name = input("Podaj nazwe pliku dla ktorego zostana zakodowane
symbole: ")
file_name += ".txt"
try:
    file = open(file_name, "r") #plik z symbolami do zakodowania
except:
    print("Problem z otwarciem pliku")
    exit()
wejscie = [] # lista w ktorej zostaną zapisane wektory kodowe
while True: #czytaj z pliku
    char = file.read(1)
    if not char:
        break
    wejscie.append(char) #dodaj znak do lisy
file.close()
ile_znakow = wejscie.count('\n')
for i in range(ile_znakow):# usun znaki nowej lini jesli wystepuja
    wejscie.remove('\n')
rozmiarSlownikBufor = int((len(wejscie) - 1)/32) #definiuj dlugosc slownika
i bufora
#kodowanie
kodowanie(wejscie, rozmiarSlownikBufor)
#dekodowanie
dekoduj("TIK_output.txt", rozmiarSlownikBufor)
```