LZ77 (Lempel-Ziv 77) – metoda strumieniowej słownikowej kompresji danych. Metoda wykorzystuje fakt iż w danych przeznaczonych do zakodowania powtarzają się pewne ciągi bajtów/znaków (zależy jaką formą danych wejściowych się posługujemy).

Przykład 1:

Zakodujmy string -> **ababcbababaaaaaa**

reszta	słownik[0:3]	bufor	pozostałe	wyjście	komentarz
	_ _ _ _	abab	cbababaaaaaa	0 0 a	nie znaleziono dopasowania, przesuwamy o 1 w lewo
	_ _ _a	babc	bababaaaaaa	0 0 b	nie znaleziono dopasowania, przesuwamy o 1 w lewo
	_ _ ab	abcb	ababaaaaaa	2 2 c	dopasowano 'ab' na indeksie 2, przesuwamy o 3 w lewo
a	babc	baba	baaaaaa	0 3 a	dopasowano 'bab' na indeksie 0, przesuwam o 4 w lewo
ababc	baba	baaa	aaa	0 2 a	dopasowano 'ba' na indeksie 0, przesuwam o 3 w lewo
ababcbab	abaa	aaaa		2 2 a	dopasowano 'aa' na indeksie 2, przesuwam o 3 w lewo
ababcbababa	aaaa	a		0 1 _	dopasowano 'a' na indeksie 0, przesuwam o 2
					koniec - bufor pusty

Wynikiem pracy algorytmu jest słownik oraz lista elementów kodujących. Dla powyższych danych będzie to (0,0,a), (0,0,b), (2,2,c), (0,3,a), (0,2,a), (2,2,a), (0,1,_).

Przyjmujemy, że _ oznacza koniec/brak znaku.

Algorytm:

1. Przyjmijmy rozmiar bufora na n oraz rozmiar słownika na m.
2. Wypełnijmy bufor pierwszymi n znakami.
3. W słowniku szukamy najdłuższego możliwego prefiksowego ciągu znaków, który znajduje się w słowniku.
4. Jeżeli nie znajdziemy dopasowania to zapisujemy wartość **(i, j, k)** gdzie **i** oraz **j** to 0 a **k** to pierwszy symbol w buforze. Następnie przesuwamy całość o **1** w lewo.
5. Jeżeli znajdziemy dopasowanie to zapisujemy wartość **(i, j, k)** gdzie **i** to indeks znalezionego ciągu, **j** to rozmiar znalezionego podciągu, **k** to pierwszy znak za znalezionym ciągiem. Następnie przesuwamy się o **j+1** w lewo.
6. Wracamy do punktu 3 jeżeli w buforze znajdują się jakieś dane.

Algorytm dekodujący:

1. Dla każdej wartości **(i, j, k):**
    a. wyprowadź ciąg o długości **j** ze słownika zaczynając od indeksu **i**
    b. wyprowadź **k**

Słownik:

indeks	słownik[0:3]
(0,0,a)	
(0,0,b)	a
(2,2,c)	ab
(0,3,a)	babc
(0,2,a)	baba
(2,2,a)	abaa
(0,1,_)	aaaa

Ciąg kodujący:

(0,0,a), (0,0,b), (2,2,c), (0,3,a), (0,2,a), (2,2,a), (0,1,_)

wynik =””

1. Dla (0, 0, a) wyprowadzamy od indeksu 0 o długości 0. Następnie doklejamy „a”.

    wynik=”a”

2. Dla (0,0,b) wyprowadzamy ciąg od indeksu 0 o długości 0. Następnie doklejamy „b”.

    wynik=”ab”

3. Dla (2,2,c) wyprowadzamy ciąg od indeksu 2 o długości 2. Następnie doklejamy „c”.

    wynik=”ababc”

4. Dla (0,3,a) wyprowadzamy ciąg od indeksu 0 o długości 3. Następnie doklejamy „a”.

    wynik=”ababcbaba”

5. Dla (0,2,a) wyprowadzamy ciąg od indeksu 0 o długości 2. Następnie doklejamy „a”.

    wynik=”ababcbababaa”

6. Dla (2,2,a) wyprowadzamy ciąg od indeksu 2 o długości 2. Następnie doklejamy „a”

    wynik=”ababcbababaaaaa”

7. Dla (0,1,_) wyprowadzamy ciąg od indeksu 0 o długości 1. Nic już nie doklejamy.

   wynik=”ababcbababaaaaaa”

Zadanie:

Zaimplementuj algorytm kodowania LZ77

Na ocenę dst:

- zaimplementuj dekoder LZ77 (program przyjmuje słownik oraz kod w dowolnej formie np. z pliku txt)

Na ocenę db:

- zaimplementuj algorytm LZ77

Na ocenę bdb:

- zaimplementuj pełny algorytm LZ77

-zaimplementuj dekoder LZ77 (oczywiście wymaga to zapamiętania w jakiejś formie wcześniej wygenerowanego słownika oraz kodu)