

# Kompresja bezstratna algorytmem LZ77<sup>1</sup>

Algorytm LZ77 koduje tekst w postaci ciągu trójek  $(p_i, c_i, s_i)$ , gdzie  $p_i$  i  $c_i$  są nieujemnymi liczbami całkowitymi, a  $s_i$  pojedynczym znakiem. Trójki te stanowią instrukcję, jak można odtworzyć oryginalny tekst: w  $i$ -tym kroku należy przepisać z dotychczas odkodowanego fragmentu tekstu  $c_i$  kolejnych znaków, poczynając od znaku znajdującego się  $p_i$  pozycji przed ostatnim odkodowanym znakiem, a następnie dopisać jeszcze znak  $s_i$ . Zauważmy, że  $p_i + 1$  może być mniejsze niż  $c_i$ , wtedy należy przepisać również znaki dopisane w tym samym kroku.

## Przykład:

Ciąg  $(0,0,a), (0,1,b), (1,3,c)$  koduje tekst aababac (po pierwszym kroku mamy słowo a; w drugim kroku przepisujemy ostatni znak i dopisujemy b otrzymując aab, w trzecim kroku przepisujemy trzy znaki poczynając od drugiego wystąpienia litery a i dopisujemy c).

Pierwsza część zadania polega na zdekodowaniu tekstu zadokowanego w taki sposób, jak opisano powyżej.

Celem drugiej części zadania, dla danego tekstu  $t$  oraz stałej  $p_{max}$ , jest znalezienie kodowania  $t$ , w którym dla każdego  $i$  zachodzi  $p_i \leq p_{max}$  oraz liczba użytych trójek jest minimalna. Można to robić odwracając powyższą procedurę dekodowania i działając zachłannie, tzn. w każdym kroku dodawać do wyniku trójkę  $(p_i, c_i, s_i)$  z największym możliwym  $c_i$ .

Dokładniej, algorytm kodujący w każdym kroku powinien wykonywać następujące operacje:

1. Niech  $w$  = ostatnie  $p_{max} + 1$  zakodowanych znaków z  $t$
2. Niech  $r$  = wszystkie niezadokowane znaki z  $t$
3. Znajdź w słowie  $wr$  najdłuższe możliwe wystąpienie właściwego prefiksu słowa  $r$  zaczynające się wśród pierwszych  $|w|$  pozycji. Niech  $c$  będzie długością znalezionej prefiksu a  $j$  indeksem pierwszego znaku znalezionej prefiksu liczoną od 1.
4. Dodaj do wyniku trójkę  $(|w| - j, c, r[c + 1])$
5. Oznacz pierwsze  $c + 1$  znaków z  $w$  jako zakodowane

Na maksymalną ocenę z części laboratoryjnej wymagana jest implementacja, w której każdy krok będzie realizowany w czasie liniowym (tzn.  $O(n)$ , gdzie  $n$  jest liczbą znaków w kodowanym napisie). Można otrzymać zmniejszoną liczbę punktów za implementację naiwną.

W części domowej wymagana jest efektywniejsza implementacja, która w kroku  $i$  będzie wykonywała  $O(p_{max} + c_i)$  operacji (zauważmy, że może to być dużo mniej niż  $O(n)$ ).

## Punktacja:

Dekodowanie – 1p

Kodowanie naiwne – 1p

Kodowanie, w którym każdy krok wykonuje się w czasie  $O(t)$  – 3p

## Wskazówki:

- Krok 3. można zrealizować modyfikując algorytm KMP. Wystarczy traktować  $r$  jako wzorzec i zapamiętywać, kiedy udało się dopasować najdłuższy fragment wzorca.
- Najlepszą złożoność  $O(p_{max} + c)$  można osiągnąć wyliczając tablicę  $P$  w algorytmie KMP w sposób leniwy (wyznaczać kolejną wartość dopiero, kiedy jest potrzebna) i odwołując się do słów  $w$  i  $r$  przez odpowiednie indeksowanie oryginalnego tekstu, bez zbędnego kopiowania.

---

<sup>1</sup> W zadaniu opisana jest pewna modyfikacja algorytmu – pomijamy górne ograniczenie na długość dopasowywanego wzorca. W praktycznych zastosowaniach (na przykład w formacie .zip) to ograniczenie jest istotne i, zwykle, dużo mniejsze niż długość fragmentu, w którym tego wzorca szukamy.