

# Równoważność cykliczna ciągów

## Definicja problemu i przedstawienie rozwiązań

Mikołaj Juda

2023

W referacie przedstawiono problem równoważności cyklicznej ciągów oraz różne algorytmy do jego rozwiązania razem z implementacją w języku Python. Pokrótce omówiono algorytm naiwny oraz algorytm korzystający z wyszukiwania wzorca. Przedstawiono również szybki algorytm sprawdzania równoważności list cyklicznych Shiloacha(1979)[1] oraz szczegółowo opisano dowód jego poprawności i analizę złożoności obliczeniowej.

## Spis treści

<b>1</b>	<b>Definicja problemu</b>	<b>2</b>
<b>2</b>	<b>Algorytm naiwny</b>	<b>3</b>
2.1	Opis . . . . .	3
2.2	Implementacja . . . . .	3
	<b>Bibliografia</b>	<b>4</b>

## 1 Definicja problemu

Dane są dwa ciągi  $A = (a_0, \dots, a_{n-1})$  oraz  $B = (b_0, \dots, b_{n-1})$  długości  $n$ .  $A$  i  $B$  są *równoważne cyklicznie* ( $A \equiv B$ ), gdy są równe w sensie list cyklicznych tzn.

### Definicja 1.1.

$$A \equiv B \iff \exists_{k_0 \in \mathbb{Z}} \forall_{k \in \{0, \dots, n-1\}} a_{(k_0+k) \pmod n} = b_k$$

Dla wygody dalszego zapisu oznaczmy:

$$a_k := a_{k \pmod n}, \quad b_k := b_{k \pmod n} \quad \text{dla wszystkich } k \geq n$$

Zdefiniujmy  $A_k$  jako listę powstałą z przesunięcia cyklicznego ciągu  $A$  takiego, że  $a_k$  jest pierwszym elementem ciągu  $A_k$ . Analogicznie dla  $B_k$ .<sup>1</sup>

$$\begin{aligned} A_k &= [a_k, \dots, a_n, a_0, \dots, a_{k-1}] \\ B_k &= [b_k, \dots, b_n, b_0, \dots, b_{k-1}] \end{aligned}$$

Definicję Definicja 1.1 można przedstawić równoważnie jako:

### Definicja 1.2.

$$A \equiv B \iff \exists_{k_0 \in \mathbb{Z}} A_{k_0} = B_0$$

Podsumowując, problem brzmi: „Czy istnieje takie przesunięcie cykliczne jednego ciągu, że jest po nim równy drugiemu ciągowi?”

---

<sup>1</sup>  $A_0 = [a_0, \dots, a_{n-1}]$ , oraz  $B_0 = [b_0, \dots, b_{n-1}]$

## 2 Algorytm naiwny

### 2.1 Opis

Z **Definicji 1.1** można łatwo zauważyć, że

**Lemat 2.1.** Jeżeli nie istnieje  $k_0 \in \{0, \dots, n-1\}$  spełniające warunek:

$$\forall_{k \in \{0, \dots, n-1\}} a_{k_0+k} = b_k$$

to nie istnieje  $k_0 \in \mathbb{Z}$  spełniające ten warunek.

*Dowód.* Oczywiście. ■

**Wniosek 2.2.** Żeby ustalić istnienie  $k_0$  z **Definicji 1.1** wystarczy sprawdzić czy

$$\exists_{k_0 \in \{0, \dots, n-1\}} \forall_{k \in \{0, \dots, n-1\}} a_{k_0+k} = b_k$$

Algorytm naiwny sprawdza dla każdego  $c \in \{0, \dots, n-1\}$  czy

$$\forall_{k \in \{0, \dots, n-1\}} a_{c+k} = b_k$$

Jeśli trafi na  $c$  spełniające warunek to zwraca True, w przeciwnym wypadku zwraca False. Algorytm ma złożoność kwadratową.[\[2\]](#)

### 2.2 Implementacja

```
def rownowazne_cyklicznie(a: list, b: list) -> bool:
    if len(a) != len(b):
        return False
    n = len(a)
    for c in range(n):
        for k in range(n):
            if a[(c + k) % n] != b[k]:
                break
        else:
            return True
    return False
```

## Bibliografia

- [1] Yossi Shiloach. „A fast equivalence-checking algorithm for circular lists”. W: *Information Processing Letters* 8.5 (1979), s. 236–238. ISSN: 0020-0190. DOI: [https://doi.org/10.1016/0020-0190\(79\)90114-5](https://doi.org/10.1016/0020-0190(79)90114-5). URL: <https://www.sciencedirect.com/science/article/pii/0020019079901145>.
- [2] *Algorytmy i struktury danych/Wstęp: poprawność i złożoność algorytmu*. 2020. URL: [https://wazniak.mimuw.edu.pl/index.php?title=Algorytmy\\_i\\_struktury\\_danych/Wst%C4%99p:\\_poprawno%C5%9B%C4%87\\_i\\_z%C5%82o%C5%BCono%C5%9B%C4%87\\_algorytmu#Algorytm\\_8.\\_R.C3.B3wnowa.C5.BCno.C5.9B.C4.87\\_cykliczna\\_ci.C4.85g.C3.B3w](https://wazniak.mimuw.edu.pl/index.php?title=Algorytmy_i_struktury_danych/Wst%C4%99p:_poprawno%C5%9B%C4%87_i_z%C5%82o%C5%BCono%C5%9B%C4%87_algorytmu#Algorytm_8._R.C3.B3wnowa.C5.BCno.C5.9B.C4.87_cykliczna_ci.C4.85g.C3.B3w).