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 algorytmie sprawdzania równoważności list cyklicznych Shiloacha(1979)[1] oraz szczegółowo opisano dowód jego poprawności i analizę złożoności obliczeniowej.

Spis treści

1 Definicja problemu

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

Definicja 1.1.

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

Dla wygody dalszego zapisu oznaczmy:

$$a_k \coloneqq a_{k \pmod n}, \; b_k \coloneqq b_{k \pmod n} \text{ dla wszystkich } k \geq n$$

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.[1]

$$A_k = [a_k, \ldots, a_n, a_0, \ldots, a_{k-1}]$$
$$B_k = [b_k, \ldots, b_n, b_0, \ldots, b_{k-1}]$$

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 ciagowi?”

[1] $A_0 = [a_0, \ldots, a_{n-1}]$, oraz $B_0 = [b_0, \ldots, b_{n-1}]$

2 Algorytm naiwny

2.1 Opis

Z Definicji 1.1 można łatwo zauważyć, że

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

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

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

Dowód. Oczywiste. ■

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

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

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

$$\forall_{k \in \{0,\ldots,n-1\}} \; a_{l+k} = b_k$$

Jeśli trafi na l spełniające warunek to mamy $k_0 = l$ i algorytm 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 l in range(n):
        for k in range(n):
            if a[(l + 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. 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.