Zadanie 5 - Wyszukiwanie wzorca

-

15.05.2023

Mikołaj Rożek, Aleksandra Szymańska

1. Algorytm naiwny

Algorytm naiwny to najprostsze podejście do problemu wyszukiwania wzorca w tekście. Działa on poprzez iterowanie po każdym znaku w tekście i sprawdzanie, czy kolejne znaki pasują do wzorca. W każdym kroku, algorytm porównuje znaki z tekstu i wzorca, jeżeli wszystkie znaki wzorca pasują, algorytm zwraca pozycję w tekście, na której rozpoczął się wzorzec.

Algorytm naiwny jest prosty do zrozumienia i implementacji, ale nie jest efektywny dla dużych wzorców, ponieważ w najgorszym przypadku, jego złożoność obliczeniowa wynosi O(n*m), gdzie n to długość tekstu, a m to długość wzorca.

2. Algorytm Knutha-Morrisa-Pratta (KMP)

Algorytm KMP to ulepszona wersja algorytmu naiwnego, która pozwala uniknąć niepotrzebnych porównań. Kluczem do działania algorytmu KMP jest wykorzystanie informacji, które algorytm zdobył podczas porównywania poprzednich znaków. Ta informacja jest przechowywana w tabeli, która jest tworzona na podstawie wzorca przed rozpoczęciem wyszukiwania. Tabela ta, nazywana jest tabelą KMP, i mówi algorytmowi, ile znaków można przesunąć wzorzec po nieudanym porównaniu.

Złożoność obliczeniowa algorytmu KMP wynosi O(n+m), co czyni go dużo bardziej efektywnym od algorytmu naiwnego, zwłaszcza dla dużych tekstów i wzorców.

3. Algorytm Karp-Rabina (KR)

Algorytm Karp-Rabina to kolejny algorytm wyszukiwania wzorca w tekście, który różni się od poprzednich dwóch algorytmów. Zamiast porównywać znaki bezpośrednio, algorytm KR oblicza wartość hashującą dla wzorca i dla każdego podciągu w tekście o długości wzorca. Jeżeli wartości hashujące są równe, algorytm wykonuje dodatkowe porównanie znak po znaku, aby potwierdzić, czy podciąg tekstu faktycznie pasuje do wzorca.

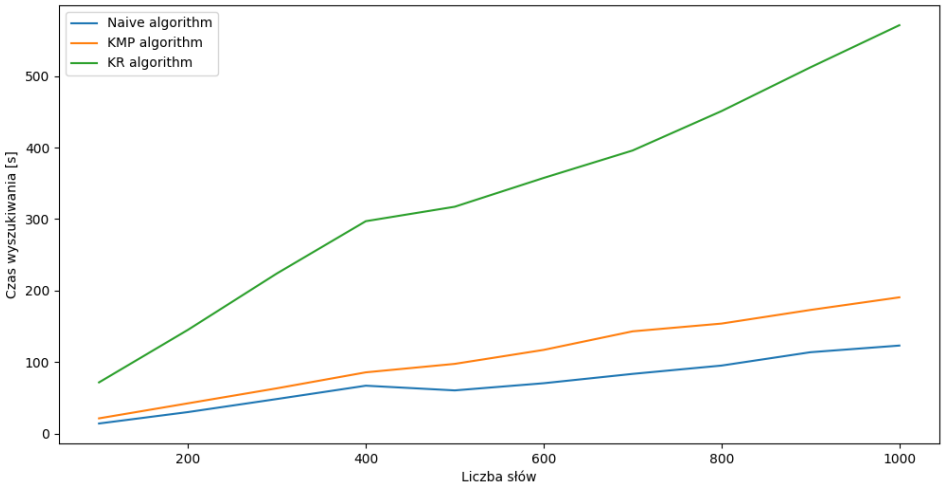
Algorytm Karp-Rabina jest efektywny, kiedy tekst składa się z dużych ilości powtarzających się fragmentów, ponieważ pozwala uniknąć niepotrzebnych porównań znak po znaku. Jednak złożoność obliczeniowa algorytmu w najgorszym przypadku wynosi O(n*m), co jest podobne do algorytmu naiwnego.

Kluczowym elementem algorytmu KR jest funkcja hashująca. Algorytm korzysta z tzw. "rolling hash", co oznacza, że gdy przesuwamy okno o jeden znak do przodu w tekście, możemy obliczyć nową wartość hasha na podstawie starej wartości, dodając informację o nowym znaku i usuwając informację o znaku, który opuszcza okno. To pozwala na szybkie obliczanie hashy dla kolejnych podciągów tekstu.

4. Testy

```
Text: UiUiUiiiiUUUiiiUUiUUUUUUiUUUUUUUUiiUiiUiUUiiiiiiUUUiiiUUUiUUiUUUUUiUUUUiUUi
Pattern: UiUii
KR algorithm returned: [2]

KMP algorithm returned: [2]

NAIVE algorithm returned: [2]

Text: UiiUUUUiUiUiUiiUiiiiUiiiiiUiiiUiUU
Pattern: iUUiiUU
KR algorithm returned: []

KMP algorithm returned: []

NAIVE algorithm returned: []

Text: UiUiiiiUUUiUiUiiUUUiUUUUUUiiiiUUUUiUUUUiiUiUUiUUUUU
Pattern: UiiiUiUiU
KR algorithm returned: []

KMP algorithm returned: []

NAIVE algorithm returned: []

Text: UUiiUUUiUiiiiUUUiUiiiUiiiiUiUUUUUUiiiiUUUiUUiUiUiUU
Pattern: iiiiUU
KR algorithm returned: [9, 34]

KMP algorithm returned: [9, 34]

NAIVE algorithm returned: [9, 34]

Text: UUUiiiUiiUUiiiUiUiUiUiiiUUiiiUiiiiUUUUiiUUiUUiiUiiUiUiUiUUUiiUUU
Pattern: iUiUiUUiU
KR algorithm returned: [52]

KMP algorithm returned: [52]

NAIVE algorithm returned: [52]
```

```
Text: vllvllllvvlvlvvlvvvlvvvvvvllvllvlvllvv
Pattern: lllvvllv
KR algorithm returned: []

KMP algorithm returned: []

NAIVE algorithm returned: []

Text: vvlvlvvvlvllvllllvllvvlvlvlvlllllvvllvlvvvvvvlvllllvvllllvlvvvvllvvv
Pattern: llvllv
KR algorithm returned: [14]

KMP algorithm returned: [14]

NAIVE algorithm returned: [14]

Text: lvlvvvlll
Pattern: ll
KR algorithm returned: [6, 7]

KMP algorithm returned: [6, 7]

NAIVE algorithm returned: [6, 7]

Text: vvvlllllllllvvllllvvvlvllllvlvllvllllvvvvvllllvlvvllllvvvvvvvlvvvvvvllvlvlvvllvvvlvllvlvvlllvll
Pattern: vlvvlv
KR algorithm returned: []

KMP algorithm returned: []

NAIVE algorithm returned: []

Text: vvvlllvvlvvvllvlllvllvvlllvvvvvllvvllllvvlvvvvlvvvvvvllvlllvvvvllvlvvvllvvvvvvllvvvvlvv
Pattern: lvllvvvvv
KR algorithm returned: []

KMP algorithm returned: []

NAIVE algorithm returned: []
```



Rysunek 1. Wykres porównujący algorytmy

5. Wnioski

W naszym przypadku algorytm naiwny zadziałał najszybciej i miał ku temu powód. Algorytm naiwny, pomimo swojej prostoty, może być efektywny, gdy szukamy krótkiego wzorca w dużym tekście. To dlatego, że nie wymaga on żadnych dodatkowych operacji przygotowawczych, takich jak obliczanie funkcji haszujących w Karpie - Rabinie czy tworzenie tablicy prefiksowej w KMP. W przypadku krótkiego, jednowyrazowego wzorca, dodatkowy czas potrzebny na przygotowanie przewyższył potencjalne korzyści pozostałych algorytmów. Algorytm Karpa - Rabina jest szczególnie efektywny, gdy tekst zawiera dużo powtarzających się fragmentów. "Pan Tadeusz"posiada unikalną strukturę, która prawdopodobnie nie zawiera tej cechy, co tłumaczy, dlaczego był najwolniejszy.