Temat: Wyszukiwanie wzorca w tekście metodą naiwną.

Wprowadzenie do wyszukiwania wzorca w tekście

Wyszukiwanie wzorca w tekście to problem polegający na znalezieniu wszystkich miejsc wystąpień danego ciągu znaków (tzw. **wzorca**) w innym ciągu znaków (tzw. **tekście**). Istnieje wiele algorytmów do tego zadania, a jednym z najprostszych jest **metoda naiwna**.
Metoda naiwną jest łatwa do zrozumienia i zaimplementowania, ale nie jest najwydajniejsza – zwłaszcza przy dużych tekstach i długich wzorcach. Mimo to, jest dobrym punktem wyjścia do nauki bardziej zaawansowanych metod.

Na czym polega metoda naiwna?

Metoda naiwna polega na porównywaniu wzorca z każdą możliwą pozycją w tekście, krok po kroku, od lewej do prawej. Algorytm próbuje dopasować wzorzec w każdej pozycji, przesuwając go o jedno miejsce, aż do końca tekstu. Jeśli wzorzec zostanie dopasowany, algorytm zapisuje pozycję, a następnie przesuwa wzorzec o jedno miejsce dalej i kontynuuje wyszukiwanie.

Przykład działania metody naiwniej

Załóżmy, że mamy tekst:
`tekst: "ABABABABCABABABABAB"`
`wzorzec: "ABAB"`
Metoda naiwna sprawdzi wzorzec w każdej możliwej pozycji tekstu:

1. Porównaj wzorzec z pozycją 0 tekstu ("ABAB" == "ABAB") – znaleziono dopasowanie.
2. Przesuń wzorzec o jedno miejsce dalej i porównaj z pozycją 1 tekstu ("BAB" != "ABAB") – brak dopasowania.
3. Powtórz proces aż do końca tekstu.

Poniżej znajduje się lista kroków algorytmu:

1. Wczytaj *tekst* i *wzorzec*
2. Przypisz m długość *tekst* i n długość *wzorzec*
3. Dla każdego indeksu *i* z *[0, n - m + 1]:*
    - Wybierz ciąg [i, i + m] z *tekstu* i porównaj z *wzorzec*
    - Jeśli są identyczne zwróć, że wzorzec istnieje, w przeciwnym razie kontynuuj pętle
4. Jeśli program wcześniej nie zwrócił wyniku to zwróć, że *wzorzec* nie występuje

Przykład działania

Weźmy tekst "**INFORMACJA**" i wyszukajmy tekst "**MA**" kolejno należy porównywać następujące fragmenty:

Fragment	Szukany fragment
IN	Nie
NF	Nie
FO	Nie
OR	Nie
RM	Nie
MA	Tak
AC	Nie
CJ	Nie
JA	Nie

Szukany wzorzec został odnaleziony. W zależności od przeznaczenia algorytmu można go kontynuować w celu zliczenia wszystkich wystąpień wzorca, albo przerwać po znalezieniu pierwszego wystąpienia wzorca.

KOD PROGRAMU

```
def wyszukaj_wzorzec(tekst, wzorzec):
    # Pobierz długości tekstu i wzorca
    dl_tekst = len(tekst)
    dl_wzorzec = len(wzorzec)

    # Przechowuj listę pozycji, gdzie znaleziono wzorzec
    pozycje = []

    # Przejdź przez każdą możliwą pozycję w tekście
    for i in range(dl_tekst - dl_wzorzec + 1):
        # Sprawdź, czy wzorzec pasuje do tekstu w bieżącej pozycji
        if tekst[i:i + dl_wzorzec] == wzorzec:
            # Jeśli pasuje, dodaj bieżącą pozycję do wyników
            pozycje.append(i)

    # Zwróć listę pozycji, gdzie wzorzec został znaleziony
    return pozycje

# Przykład użycia:
tekst = "ABABABABCABABABABAB"
wzorzec = "ABAB"
wynik = wyszukaj_wzorzec(tekst, wzorzec)

if wynik:
    print(f"Wzorzec '{wzorzec}' znaleziono w pozycjach: {wynik}")
else:
    print("Wzorzec nie został znaleziony.")
```

Wyjaśnienie kodu:

1. **Pętla for** przechodzi przez każdą możliwą pozycję w tekście, od której wzorzec może się zaczynać.
2. **Porównanie tekst[i:i + dl_wzorzec] == wzorzec** sprawdza, czy wycinek tekstu o długości wzorca odpowiada wzorcowi.
3. **Lista pozycje** przechowuje wszystkie indeksy, w których wzorzec został znaleziony.
4. Jeśli wzorzec został znaleziony, program wyświetla jego pozycje.

Zastosowanie metody naiwnej

Metoda naiwna jest prosta do zaimplementowania i skuteczna przy małych tekstach lub krótkich wzorcach. Można ją stosować np. do:

- wyszukiwania słów w krótkich dokumentach,
- analizy tekstów o niewielkich rozmiarach,
- podstawowych aplikacji edukacyjnych.

Podsumowanie

Metoda naiwna jest efektywna do prostych zastosowań, ale jej wydajność spada przy dużych danych. Stanowi jednak doskonały punkt wyjścia do nauki bardziej zaawansowanych technik wyszukiwania wzorców, takich jak algorytmy o lepszej złożoności czasowej.

ZADANIA DO WYKONANIA

ZAD. 1 **Wyszukiwanie słowa w zdaniu**

Napisz program, który wyszuka wszystkie wystąpienia danego słowa w zdaniu. Użytkownik podaje zdanie oraz szukane słowo. Program powinien zwrócić listę pozycji, na których to słowo występuje.

Dane wejściowe:

- Zdanie (np. "Ala ma kota, a kot ma Alę").
- Wzorzec do wyszukania (np. "ma").

Oczekiwany wynik:

- Lista pozycji (indeksów), w których słowo występuje.

ZAD.2 **Wyszukiwanie podciągu w DNA**

Napisz program, który przeszuka sekwencję DNA pod kątem wystąpienia określonego podciągu nukleotydów. Użytkownik podaje sekwencję DNA oraz wzorzec (podciąg) do wyszukania.

Dane wejściowe:

- Sekwencja DNA (np. "ACTGACTGCTAGCTAGACTG").
- Wzorzec (np. "CTG").

Oczekiwany wynik:

- Liczba wystąpień wzorca w sekwencji DNA.

ZAD.3 **Wyszukiwanie imienia na liście**

Napisz program, który znajdzie wszystkie wystąpienia imienia w liście tekstowej. Użytkownik podaje listę (ciąg znaków z nazwiskami oddzielonymi spacjami) oraz imię, którego szuka.

Dane wejściowe:

- Lista imion (np. "Anna Tomasz Anna Jan Maria Anna").
- Wzorzec do wyszukania (np. "Anna").

Oczekiwany wynik:

- Liczba wystąpień wzorca na liście oraz ich indeksy.

ZAD.4 **Wyszukiwanie frazy w pliku tekstowym**

Napisz program, który przeszuka plik tekstowy pod kątem wystąpienia danej frazy. Użytkownik podaje nazwę pliku oraz frazę do wyszukania. Program powinien wyświetlić numery linii, w których wystąpiło dopasowanie.

Dane wejściowe:

- Nazwa pliku tekstowego (np. "dokument.txt").
- Fraza do wyszukania (np. "Python").

Oczekiwany wynik:

- Numery linii, w których fraza została znaleziona.

ZAD.5 **Cenzurowanie słowa w tekście**

Napisz program, który przeszuka dany tekst pod kątem konkretnego słowa i zastąpi je gwiazdkami (*). Użytkownik podaje tekst oraz słowo do ocenzurowania.

Dane wejściowe:

- Tekst (np. "To jest sekret, który musi zostać sekretem").
- Wzorzec do ocenzurowania (np. "sekret").

Oczekiwany wynik:

- Tekst, w którym wzorzec został zastąpiony znakami * (np. "To jest ******, który musi zostać ******").