11. Wyszukiwanie wzorca w tekście

Problem wyszukiwania wzorca w tekście jest niezwykle istotny w zastosowaniach informatyki. Praktycznie wszystkie edytory tekstu są wyposażone w narzędzie, które pozwala znaleźć dany ciąg znaków w dokumencie. Szukanie informacji w internecie też często sprowadza się do wpisania w wyszukiwarce fragmentu tekstu. Oczekujemy wtedy, że wyniki nie tylko będą poprawne, lecz także pojawią się szybko. W tym temacie poznasz algorytmy związane z wyszukiwaniem wzorca w tekście.

Cele lekcji

- Wyszukasz wzorzec w tekście z wykorzystaniem algorytmu naiwnego.
- Dowiesz się, na czym polega metoda haszowania.
- Poznasz podstawy arytmetyki modularnej.
- Zastosujesz funkcję haszującą do wyszukiwania wzorca w tekście.
- Poznasz algorytm Karpa–Rabina, służący do znajdowania wzorca w tekście.

Problem wyszukiwania wzorca w tekście

Problem wyszukiwania wzorca w tekście można sformułować w następujący sposób. Dane są dwa napisy: jeden nazwiemy tekstem, a drugi wzorcem. Celem jest określenie, w którym miejscu w tekście znajduje się wzorzec, o ile w ogóle w nim występuje.

Przyjmiemy, że długość wzorca (rozumiana jako liczba znaków) jest nie większa niż długość tekstu, a w praktyce jest ona wielokrotnie mniejsza. Ograniczymy znaki, które mogą występować w tekście i we wzorcu, do małych liter alfabetu łacińskiego.

Oto specyfikacja problemu:

Dobra rada

Biblioteki większości języków programowania oferują funkcję do poszukiwania wzorca w tekście. W języku C++ możesz skorzystać z biblioteki `string` i metody `find` dla klasy `string`.

Specyfikacja

Dane: `t`, `w` – niepuste napisy złożone z małych liter alfabetu łacińskiego, długość tekstu `t` jest większa lub równa długości wzorca `w`.

Wynik: `p` – liczba całkowita określająca pozycję (indeks) pierwszego wystąpienia wzorca `w` w tekście `t` – lub `-1`, gdy wzorzec nie występuje w tekście.

11.1. Algorytm naiwny

Algorytm naiwny wyszukiwania wzorca w tekście

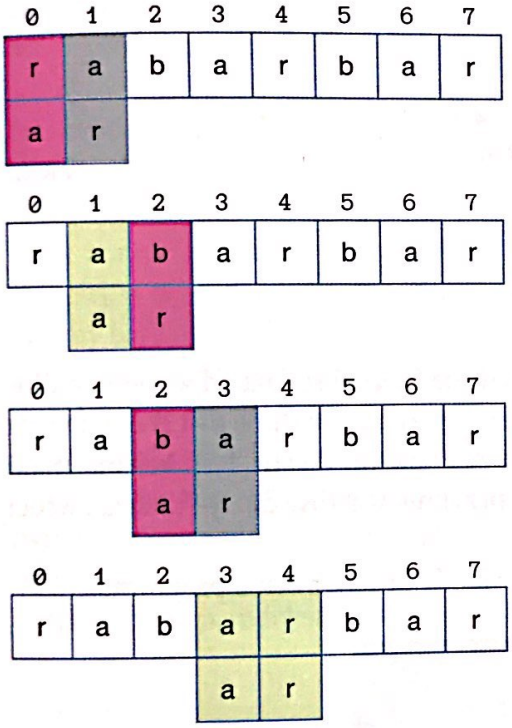
Jednym z najprostszych rozwiązań jest tzw. **algorytm naiwny wyszukiwania wzorca w tekście**, który polega na porównywaniu znak po znaku wzorca z tekstem.

W algorytmie naiwnym ustawiamy wzorzec pod tekstem tak, aby pierwszy znak wzorca znajdował się pod pierwszym znakiem tekstu i porównujemy kolejne pary znaków obu napisów.

Jeśli wszystkie znaki wzorca będą odpowiadały znakom fragmentu tekstu, to znaleźliśmy położenie wzorca w tekście. Jeśli jednak znajdziemy pierwszą parę różniących się znaków, to przesuwamy wzorzec o jeden znak w prawo względem tekstu i ponownie rozpoczynamy porównywanie kolejnych par znaków.

Algorytm kończy działanie po znalezieniu pozycji wzorca w tekście lub wysunięciu wzorca poza tekst, czyli gdy pozycja początkowa wzorca jest większa od długości tekstu pomniejszonej o długość wzorca.

Opisany algorytm pokażemy na przykładzie. Poszukamy wzorca *ar* w tekście *rabarbar*. Przedstawia to rysunek 11.1.



Rys. 11.1. Naiwne wyszukiwanie wzorca w tekście

Oto algorytm naiwny w postaci funkcji, zapisany w pseudokodzie:

```
funkcja Znajdz(w,t)
    p ← 0
    dopóki p ≤ długość t - długość w wykonuj
        i ← 0
        dopóki i < długość w oraz w[i] = t[p+i] wykonuj
            i ← i + 1
        jeśli i = długość w to zwróć p i zakończ
        w przeciwnym przypadku p ← p + 1
    zwróć -1 i zakończ
```

Dobra rada

Jeśli chcesz znaleźć wszystkie wystąpienia wzorca w tekście, kontynuuj działanie algorytmu, aż pozycja początkowa wzorca będzie większa od długości tekstu pomniejszonej o długość wzorca.

Zmienna pomocnicza i określa indeks porównywanego znaku wzorca. W zależności od przesunięcia wzorca względem tekstu (wartości zmiennej p) porównujemy znaki w[i] z t[p+i] dla i w zakresie od 0 do długości wzorca minus 1, chyba że wcześniej pojawi się para różnych znaków. Wówczas zwiększamy wartość zmiennej p, czyli przesuwamy wzorzec o jeden znak w prawo względem tekstu.

Oto kod źródłowy funkcji Znajdz, realizującej algorytm naiwny wyszukiwania wzorca w tekście:

Kod źródłowy funkcji Znajdz, realizującej algorytm naiwny wyszukiwania wzorca w tekście

```
1.  int Znajdz(string w, string t)
2.  {
3.      int dw=w.size();
4.      int dt=t.size();
5.      int i, p=0;
6.      while (p<=dt-dw)
7.      {
8.          i=0;
9.          while (i<dw && w[i]==t[p+i]) i++;
10.         if (i==dw) return p;
11.         else p++;
12.     }
13.     return -1;
14. }
```

Zmienne pomocnicze dw i dt (linie 3–4) określają odpowiednio długość wzorca i długość tekstu. Zewnętrzna pętla (linie 6–12) odpowiada za sprawdzenie znaków wzorca ze znakami tekstu. Jeśli kolejne znaki będą sobie równe, a zmienna i przyjmie wartość dw, to funkcja zwróci pozycję wzorca (linia 10).

Rysunek 11.2 przedstawia przykłady uruchomienia programu.

```
Tekst: rabarbar
Szukany wzorzec: ar
Pozycja wzorca w tekscie: 3
```

```
Tekst: rabarbar
Szukany wzorzec: rak
Wzorzec nie wystepuje w tekscie
```

Rys. 11.2. Efekt działania programu poszukującego wzorców *ar* i *rak* w tekście *rabarbar*

Ćwiczenie 1

Napisz program, który wczyta tekst oraz wzorzec, zapisane małymi literami alfabetu łacińskiego, a następnie wyszuka wzorzec w tekście, stosując algorytm naiwny. W programie wykorzystaj kod źródłowy funkcji Znajdz.