|  |
| --- |
| Uniwersytet WSB Merito |
| Przetwarzanie Informacji Multimedialnej |
| Ćwiczenia 1 - zadania |

|  |
| --- |
| Autor: Lesław Pawlaczyk  Październik 2023 |

Spis treści

[Rozdział 1 – Wyszukiwanie wzorców 2](#_Toc93561521)

[Rozdział 2 - Szyfrowanie 4](#_Toc93561522)

[Rozdział 3 – Sprawozdania 5](#_Toc93561523)

[Bibliografia 6](#_Toc93561524)

[Załącznik 1 – Plik wyszukiwania Morissa – Pratta przed modyfikacją 7](#_Toc93561525)

[Załącznik 2 – Plik wyszukiwania Morissa – Pratta po modyfikacji 8](#_Toc93561526)

# Rozdział 1 – Wyszukiwanie wzorców

Celem zadań związanych z pierwszymi ćwiczeniami będzie przećwiczenie operacji wyszukiwania wzorców w tekście.

W wyszukiwaniu tekstu wyróżniamy przykładowe algorytmy:

1. Wyszukiwanie naiwne (Wałaszek, Algorytm naiwny 2021).
2. Wyszukiwanie Morrisa-Pratta (Wałaszek, Wyszukiwanie Morrisa-Pratta 2021).
3. Wyszukiwanie Knutha-Morrisa-Pratta (Wałaszek, Wyszukiwanie Morrisa-Pratta 2021).
4. Wyszukiwanie Boyera-Moora (Wałaszek, Wyszukiwanie Boyera-Moore'a 2021).

Zadania do wykonania:

1. Na podstawie wskazanych algorytmów użyć strony <https://www.onlinegdb.com/online_c++_compiler>. Zmodyfikować każdy z programów, aby dla każdego testować 5 różnych tekstów z dowolnej strony internetowej (po angielsku dla prostoty kodowania ASCII). Dla każdego z tekstów ustalić wzorzec do wyszukania (wzorzec nie musi się znajdować w tekście).

Przykład 1:

tekst = „She sells sea shells by the sea shore”

wzorzec = “sea shells”

Przykład 2:

tekst = „A quick brown fox jumped over a lazy dog”

wzorzec = “fox”

Przykład 3:

tekst = „ Yesterday all my troubles seemed so far away”

wzorzec = “away”

Przykład 4:

tekst = „ Abra cabra cadabra”

wzorzec = “Cabra”

**Ważne**: W załącznikach pokazane jak modyfikować algorytmy celem przeszukiwania proponowanego wzorca.

1. Dla każdego z czterech algorytmów i wygenerowanych 5 par „tekst / wzorzec” wykonać tabelkę

|  |  |  |  |
| --- | --- | --- | --- |
| Algorytm | Tekst | Wzorzec | Wynik z konsoli |
| Naiwny | She sells sea shells by the sea shore | sea shells |  |
| Naiwny | A quick brown fox jumped over a lazy dog | fox |  |
| … |  |  |  |
| … |  |  |  |
| Boyer’a-Moore’a | Yesterday all my troubles seemed so far away | away |  |

1. Przedstawić krótką dyskusję wybranego algorytmu z punktu widzenia złożoności obliczeniowej, pozytywnych oraz negatywnych cech.
2. Dla wybranego algorytmu przedstawić jego implementację w języku C# lub Python (praca powinna być oryginalna). Do tego celu można użyć strony: <https://dotnet.microsoft.com/platform/try-dotnet> dla C# albo <https://skulpt.org/> dla Python’a).

# Rozdział 2 - Szyfrowanie

Celem zadań będzie przećwiczenia operacji szyfrowania tekstu.

W szyfrowaniu tekstu wyróżniamy przykładowe algorytmy:

1. Szyfr Cezara (Wałaszek, Szyfr Cezara 2021).
2. Szyfrowanie z pseudolosowym dostępem (Wałaszek, Szyfrowanie z pseudolosowym dostępem 2021).
3. Szyfr przestawieniowy (Wałaszek, Szyfr przestawieniowy 2021).
4. Szyfr Enigmy (Wałaszek, Szyfr Enigmy 2021).
5. Szyfr RSA (Wałaszek, Szyfr RSA 2021).

Zadania do wykonania:

1. Uruchomić każdy z algorytmów z użyciem strony <https://www.onlinegdb.com/online_c++_compiler> dla co najmniej dwóch różnych kombinacji danych wejściowych.
2. Podać wynik działania konsolki programu dla uruchomionych kombinacji danych wejściowych.
3. Przedstawić krótką dyskusję wybranego algorytmu z punktu widzenia złożoności obliczeniowej, pozytywnych oraz negatywnych cech.
4. **Opcjonalnie**. Dla wybranej jednego algorytmu przedstawić jego implementację w języku C# lub Python (praca powinna być oryginalna). Do tego celu można użyć strony: <https://dotnet.microsoft.com/platform/try-dotnet> dla C# albo <https://skulpt.org/> dla Python’a.

# Rozdział 3 – Sprawozdania

Sprawozdania w formacie Word proszę wysyłać indywidualnie w terminie do 2023/11/21 na adres leslaw.pawlaczyk@chorzow.merito.pl.

W sprawozdaniu wskazać autorów pracy, numery albumów najlepiej podać link do **prywatnego** repozytorium na Github z implementacjami algorytmów.

Ocenianie:

1. dostateczny – zadania 1.1, 1.2, 1.3 razem z dyskusją rozwiązania.
2. dobry – zadanie na ocenę dostateczny oraz zadania 2.1, 2.2, 2.3 razem z dyskusją rozwiązania.
3. dobry plus – zadania na ocenę dobry oraz 1.4
4. bardzo dobry - zadania na ocenę dobry plus oraz 2.4

# Bibliografia

Jerzy Wałaszek. *Algorytm naiwny.* 2021. https://eduinf.waw.pl/inf/alg/001\_search/0046.php.

—. *Szyfr Cezara.* 2021. https://eduinf.waw.pl/inf/alg/001\_search/0063.php.

—. *Szyfr Enigmy.* 2021. https://eduinf.waw.pl/inf/alg/001\_search/0066.php.

—. *Szyfr przestawieniowy.* 2021. https://eduinf.waw.pl/inf/alg/001\_search/0065.php.

—. *Szyfr RSA.* 2021. https://eduinf.waw.pl/inf/alg/001\_search/0067.php.

—. *Szyfrowanie z pseudolosowym dostępem.* 2021. https://eduinf.waw.pl/inf/alg/001\_search/0064.php.

—. *Wyszukiwanie Boyera-Moore'a.* 2021. https://eduinf.waw.pl/inf/alg/001\_search/0051.php.

—. *Wyszukiwanie Karpa-Rabina.* 2021. https://eduinf.waw.pl/inf/alg/001\_search/0052.php.

—. *Wyszukiwanie Knutha-Morrisa-Pratta.* 2021. https://eduinf.waw.pl/inf/alg/001\_search/0049.php.

—. *Wyszukiwanie Morrisa-Pratta.* 2021. https://eduinf.waw.pl/inf/alg/001\_search/0048.php.

# Załącznik 1 – Plik wyszukiwania Morissa – Pratta przed modyfikacją

// Wyszukiwanie wzorca algorytmem MP

// Data: 3.06.2008

// (C)2020 mgr Jerzy Wałaszek

//-----------------------------

#include <iostream>

#include <string>

#include <cstdlib>

#include <time.h>

using namespace std;

const int N = 79; // długość łańcucha s

const int M = 5; // długość wzorca p

int main( )

{

string s, p;

int PI [ M + 1 ], i, b, pp;

srand ( ( unsigned )time ( NULL ) );

// generujemy łańcuch s

s = "";

for( i = 0; i < N; i++ ) s += 65 + rand( ) % 2;

// generujemy wzorzec

p = "";

for( i = 0; i < M; i++ ) p += 65 + rand( ) % 2;

// dla wzorca obliczamy tablicę PI [ ]

PI [ 0 ] = b = -1;

for( i = 1; i <= M; i++ )

{

while( ( b > -1 ) && ( p [ b ] != p [ i - 1 ] ) ) b = PI [ b ];

PI [ i ] = ++b;

}

// wypisujemy wzorzec

cout << p << endl;

// wypisujemy łańcuch s

cout << s;

// poszukujemy pozycji wzorca w łańcuchu

pp = b = 0;

for( i = 0; i < N; i++ )

{

while( ( b > -1 ) && ( p [ b ] != s [ i ] ) ) b = PI [ b ];

if( ++b == M )

{

while( pp < i - b + 1 )

{

cout << " "; pp++;

}

cout << "^"; pp++;

b = PI [ b ];

}

}

cout << endl;

return 0;

}

# Załącznik 2 – Plik wyszukiwania Morissa – Pratta po modyfikacji

// Wyszukiwanie wzorca algorytmem MP

// Data: 3.06.2008

// (C)2020 mgr Jerzy Wałaszek

//-----------------------------

#include <iostream>

#include <string>

#include <cstdlib>

#include <time.h>

using namespace std;

int main()

{

string s, p;

// generujemy łańcuch s

s = "She sells sea shells by the sea shore";

const int N = s.length();

// generujemy wzorzec

p = "sells";

const int M = p.length();

int i, b, pp;

int\* PI = new int[M + 1];

srand((unsigned)time(NULL));

// dla wzorca obliczamy tablicę PI [ ]

PI[0] = b = -1;

for (i = 1; i <= M; i++)

{

while ((b > -1) && (p[b] != p[i - 1]))

{

b = PI[b];

}

PI[i] = ++b;

}

// wypisujemy wzorzec

cout << p << endl;

// wypisujemy łańcuch s

cout << s;

// poszukujemy pozycji wzorca w łańcuchu

pp = b = 0;

for (i = 0; i < N; i++)

{

while ((b > -1) && (p[b] != s[i]))

{

b = PI[b];

}

if (++b == M)

{

while (pp < i - b + 1)

{

cout << " ";

pp++;

}

cout << "^";

pp++;

b = PI[b];

}

}

cout << endl << "Position found = " << pp << endl;

return 0;

}