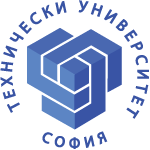
**Курсов проект по дисциплина Синтез и анализ на алгоритми за студенти ФКСТ**



Студент: Димитър Иванов Иванов ИТИ Група 36 Фак. Номер: 501219035

Ръководител: Константин Костадинов

**Тема на заданието:**

Алгоритми за търсене в стринг. Алгоритъм на Boyer-Moore

**Текст на заданието:**

Да се напише програма, която разменя местата на редовете, съдържащи най-големия и най-малкия елемент в масив с n реда и m стълба

**Начин на работа на алгоритма**: За правилното действие на алгоритма се нуждаем от масив в който отчитаме наличието на символи от шаблона който търсим. Създаваме масив с размер номер на символа ‘z’ в ASCII таблици и отбелязваме на кой индекс си намират символите.

**Пример:**

**Текст: GCAATGCCTATGTGACC**

**Шаблон: TATGTG**

**Масив[‘T’] = 4**

**Масив[‘A’] = 1**

**Масив[‘G’] = 5**

Забележете как някой символи се повтарят, в този случай Т се повтаря 3 пъти и 3 пъти презаписваме стойността на индекса където се намира. Първо записваме 0, после 2 и накрая 4.

Същото нещо важи и за G.

След записването на индексите, започваме да търсим шаблона като започнем от началото на текста в този случай сравняваме **: GCAATG**

**TATGTG**

Започваме проверката от края на шаблона. Сравняваме първите 2 символа които са равни, но 3-тия симвл не е и в този случай няма нужда да проверяваме за остатъка от шаблона и го местим напред в текста.

**Колко по-напред местим шаблона? Тук се налага да използваме масива за наличие който в началото сме попълнили.**

Преместваме шаблона напред с индекса на обхождане който е 3 минус наличието на символа от текста който е предизвикал неравенството => ‘A’ . Премества с 3 – 1 = 2 индекса напред в текста, получаваме **GCAATGCC**

**TATGTG**

Продължаваме да сравняваме шаблона с текста и да местим шаблона при неравенство докато стигнем до**: GCAATGCCTATGTG**

**TATGTG**

**В този случай има съвпадение и алгоритма приключва.**

**Общо описание:**

Задача обхваща вход на матрица заедно с вход на най-големия и малкия елемент от тази матрица.

След това обхождаме матрицата като проверяваме дали всеки елемент е или най-малкия или най-големия елемент, ако намерим най-големи и най-малкия приключваме обхождането и разменяме редовете където сме ги намерили.

Накрая принтираме матрицата независимо дали е имало размяна на редове.

Нужна памет: О(n\*m) – матрица

Нужно време: О(n\*m) – за вход

О(n\*m) – за обхождане – в най-лошия случай

О(n\*m) – за изход

**Обяснение на файловете:**

**ogranicheniq.h** – файл който съдържа всички константи свързани с ограничения на масиви.

**saobshteniq.h** – файл който съдържа всички константи съобщения които ще насочват потребителя към изпълнение на програмата.

**main.h –** header файл който съдържа всички нужни библиотеки и декларира функциите.

**main.cpp –** файл който съдържа деклариране на основните промениви за редове,колони и максимален и минимален елемент, входа на тези данни става чрез извикване на съответните фунцкии, файла приключва с извикване на фунцкия за изпълнение на алгоритма и изход на матриата за да покажем че успешно сме сменили редовете.

**vhod.cpp** – файл който съдържа функции за въвеждане на число, текст и матрицата.

**Izhod.cpp** – файл който съдържа фунцкия за принтиране на матрицата.

**Obhojdane.cpp** – файл за търсене на макс и мин елемент чрез прилагане на алгоритма върху всеки елемент от матрицата.

**booyer-moore.cpp** – главния файл на програмата, съдържа търсене елемент чрез алгоритма на booyer-moore.

**Обяснение на константите:**

Използваме константи за съобщения и поставяне на граници на масиви.

Запазваме тази информация в константи, защото няма да я променяме по време на изпълнение на програмата и защото ако искаме да променим дадено съобщение, няма да има нужда да го търсим в целия проект, защото константите са в отделния файлове.

#define **MAX\_REDOVE** 50 – максималните редове които може да побере матрицата

#define **MAX\_KOLONI** 50 – максималните колони които може да побере матрицата

#define **MIN\_DIMENSII** 1 – минимални дименсии, подсигуряваме че потребителя няма да въведе

стойност за ред или колона която е по-малка от 1.

#define **NO\_OF\_CHARS** 256 – броя на символи в аски таблицата, използва се за отбелязване на наличние на символи от шаблона.

#define **VAVEDETERED** "Vavedete redovete na matricata: " – Съобщение което уведомява потребителя за въвеждане на редове на матрицата.

#define **VAVEDETEKOLONA** "Vavedete kolonite na matricata: " – Съобщение което уведомява потребителя за въвеждане на колони на матрицата.

#define **VAVEDETESTOINOSTZAKLETKA** "Vavedete stoinost za kletka (%d,%d) : " – Съобщение за въвеждане на клетка в матрицата.

#define **VAVEDENOCHISLOGOLQMO** "Vavedenoto chislo previshava ogranicheniqta, ne moje da e poveche ot " – Предупреждава потребителя че данните които е въвел не са валидни.

#define **VAVEDENOCHISLONEMALKO** "Vavedenoto chislo e prekaleno malko, ne moje da e po-malko ot" – Предупреждава потребителя че данните които е въвел не са валидни

#define **VAVEDENIQTEXTEPRAZEN** "Vavedniq text e prazen." – Предупреждава потребителя че данните които е въвел не са валидни

#define **VAVEDETEMAXSTRING** "Vavedete nai-golemiq element: " – Съобщение което уведомява потребителя за въвеждане на максималния елемент на матрицата.

#define **VAVEDETEMINSTRING** "Vavedete nai-malkiq element: " – Съобщение което уведомява потребителя за въвеждане на минималния елемент на матрицата.

**Обяснение на функциите:**

**string vavedeteText(string saobshtenie); -** въвеждане на текст за максималния, минималния елемент и клетките на матрицата, съдържа валидация в случай че се въведе празен стринг.

**int vavedeteChislo(string saobshtenie,int minGranica,int maxGranica); -** въвеждане на число за граници на матрицата, съдържа валидация за числото дали се намира в подадените в функцията граници.

**void vhodMatrica(string matrica[MAX\_REDOVE][MAX\_KOLONI],int redove,int koloni); -** обхождаме матрицата и приемаме стойност за всякя клетка като преизползваме функцията vavedeteText;

**void izhodMatrica(string matrica[MAX\_REDOVE][MAX\_KOLONI],int redove,int koloni); -** обхождаме матрицата и принтираме всяка клетка.

**void markiraneNaNalichie(string str, int size,int nalichie[NO\_OF\_CHARS]); -** макрираме символите от шаблона с индекса им.

**int booyerMoore(string text, string shablon);** - фунцкия за прилагането на алгоритма, обхождаме текста и проверяваме дали съвпада с шаблона.

**void proveriZaBoyerMoore(string matrica[MAX\_REDOVE][MAX\_KOLONI],int redove,int koloni,string maxStr,string minStr); -** обхождаме матрицата и проверяваме дали всяка клетка е или минимални или максималния елемент, ако намерим 2-та елемент приключваме обхождането и разменяме техните редове.

**Източници на информация:**

**https://www.geeksforgeeks.org/boyer-moore-algorithm-for-pattern-searching/**

Boyer-Moore Substring Search - Part 1

**-https://www.youtube.com/watch?v=3Ft3HmizsCk**

# Boyer-Moore Substring Search - Part 2

**-https://www.youtube.com/watch?v=Tbj8iH9UkSA**

# Boyer Moore Pattern Matching Algorithm

**-** **https://www.youtube.com/watch?v=4Oj\_ESzSNCk&t=542s**

**Код**:

**main.cpp**

#include "main.h"

int main() {

int redove,koloni;

string maxStr,minStr,temp;

/\* Вход на размери на матрицата \*/

redove = vavedeteChislo(VAVEDETERED,MIN\_DIMENSII,MAX\_REDOVE);

koloni = vavedeteChislo(VAVEDETEKOLONA,MIN\_DIMENSII,MAX\_KOLONI);

cin.ignore(1,'\n'); //Игнорираме \n тъй като getline го обхваща

string matrica[MAX\_REDOVE][MAX\_KOLONI];

vhodMatrica(matrica,redove,koloni);

/\* Вход на най-голям елемент \*/

maxStr = vavedeteText(VAVEDETEMAXSTRING);

minStr = vavedeteText(VAVEDETEMINSTRING);

proveriZaBoyerMoore(matrica,redove,koloni,maxStr,minStr);

izhodMatrica(matrica,redove,koloni);

return 0;

}

**vhod.cpp**

#include "main.h"

void vhodMatrica(string matrica[MAX\_REDOVE][MAX\_KOLONI],int redove,int koloni) {

//Въвеждане на матрицата

for(int red=0; red<redove; red++) {

for(int kolona = 0; kolona < koloni; kolona++) {

char str[39];

sprintf(str,VAVEDETESTOINOSTZAKLETKA, red,kolona);

matrica[red][kolona] = vavedeteText(str);

}

}

}

int vavedeteChislo(string saobshtenie,int minGranica,int maxGranica) {

int num;

cout << saobshtenie;

cin >> num;

while(num < minGranica || num > maxGranica) {

if(num < minGranica) {

cout << VAVEDENOCHISLONEMALKO << minGranica << endl;

}

if(num > maxGranica) {

cout << VAVEDENOCHISLOGOLQMO << maxGranica << endl;

}

cout << saobshtenie;

cin >> num;

}

return num;

}

string vavedeteText(string saobshtenie) {

string str;

cout << saobshtenie;

getline(cin,str);

while(str.empty()) {

cout << VAVEDENIQTEXTEPRAZEN << endl;

cout << saobshtenie;

getline(cin,str);

}

return str;

}

**izhod.cpp**

#include "main.h"

void izhodMatrica(string matrica[MAX\_REDOVE][MAX\_KOLONI],int redove,int koloni) {

//Принтиране на матрицата

for(int red=0; red<redove; red++) {

cout << "Red " << red + 1 << " : ";

for(int kolona = 0; kolona < koloni; kolona++) {

if(kolona == koloni - 1) {

cout << matrica[red][kolona] << endl;

} else {

cout << matrica[red][kolona] + ", ";

}

}

}

}

**obhojdane.cpp**

#include "main.h"

/\* Обхождаме всички елементи в матрица и при намиране на макс и мин приключваме обхождане и разместваме тяхните редове \*/

void proveriZaBoyerMoore(string matrica[MAX\_REDOVE][MAX\_KOLONI],int redove,int koloni,string maxStr,string minStr) {

int red,kolona,maxProverka,minProverka,namereniSa,redaNaMax,redaNaMin;

maxProverka = -1;

minProverka = -1;

namereniSa = -1;

redaNaMax = -1;

redaNaMin = -1;

//Прилагаме алгоритъма за всеки елемент, проверяваме дали е макс или мин елемент

for(red=0; red<redove; red++) {

for(kolona = 0; kolona < koloni; kolona++) {

//Ако един път сме намерили най-големия стринг няма смисъл отново да го търсим

if(maxProverka == -1) {

maxProverka = booyerMoore(matrica[red][kolona],maxStr);

}

//Записваме реда на максималния елемент, правим проверка дали вече не сме го записали

if(maxProverka == 1 && redaNaMax == -1) {

redaNaMax = red;

}

//Ако един път сме намерили най-малкия стринг няма смисъл отново да го търсим

if(minProverka == -1) {

minProverka = booyerMoore(matrica[red][kolona],minStr);

}

//Записваме реда на минималния елемент, правим проверка дали вече не сме го записали

if(minProverka == 1 && redaNaMin == -1) {

redaNaMin = red;

}

//Намерили сме и най-големия и най-малкия стринг

if(maxProverka == 1 && minProverka == 1) {

namereniSa = 1;

break;

}

}

//Ако сме намерили редовете на максималния и минималния няма нужда да продължаваме обхождането

if(namereniSa == 1) {

break;

}

}

//Разместване на редовете съдърщаши макс и мин елемент

if(redaNaMax != -1 && redaNaMin != -1) {

string temp;

for(kolona = 0; kolona < koloni; kolona++) {

temp = matrica[redaNaMax][kolona];

matrica[redaNaMax][kolona] = matrica[redaNaMin][kolona];

matrica[redaNaMin][kolona] = temp;

}

}

}

**booyer-moore.cpp**

#include "main.h"

/\* Търсене на стринг в текст с използване на Boyer Moore алгоритъм \*/

int booyerMoore(string text, string shablon) {

int shablonDaljina = shablon.size();

int textDaljina = text.size();

/\* Проверяваме за равенство между 2-те дължини защото шаблона

ще отчета ако имаме текст който вътрешн съдържа шаблона.

В този случай грешно че отчетем местоположението на максималния и минималния елемент

\*/

if(shablonDaljina != textDaljina){

return -1;

}

int nalichie[NO\_OF\_CHARS];

/\* Пълним масива с -1 за неналичие и с позиция на символа в шаблона ако има наличие \*/

markiraneNaNalichie(shablon, shablonDaljina, nalichie);

/\* Индекс за показване докъде сме стигнали в текста\*/

int indexText = 0;

while(indexText <= (textDaljina - shablonDaljina)) {

int j = shablonDaljina - 1;

/\* Намаляваме j докато текста и шаблона са еднакви \*/

while(j >= 0 && shablon[j] == text[indexText + j]) {

j--;

}

/\* Ако текста съдържа шаблона тогава j ще стане -1\*/

if (j < 0) {

/\* Шаблона е намерен връщаме 1 за да потвърдим \*/

return 1;

} else {

/\* Местим шаблона толкова знака напред колкото е наличието

на символа от текста където сме имали неравенство

от масива за наличие на символи.

Макс функция е сложена за да осигурен преместване надясно в текста.\*/

indexText += max(1, j - nalichie[text[indexText + j]]);

}

}

return -1;

}

// Предварителна фунцкия за Boyer Moore,маркиране на наличните символи

void markiraneNaNalichie(string str, int size,int nalichie[NO\_OF\_CHARS]) {

int i;

// Иниализираме всички символи с -1 за неналичие

for (i = 0; i < NO\_OF\_CHARS; i++)

nalichie[i] = -1;

/\* Поставяме индекса за да потвърдим наличие

Ако имаме 2 еднакви символа ще се пренапише

\*/

for (i = 0; i < size; i++) {

nalichie[(int) str[i]] = i;

}

}