**Министерство цифрового развития, связи**

**и массовых коммуникаций Российской Федерации**

**Ордена Трудового Красного Знамени**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский технический университет связи и информатики»**

Кафедра «Математическая кибернетика и информационные технологии»

**Отчет по лабораторной работе №3**

по дисциплине «Структуры и алгоритмы обработки данных»

на тему «Методы поиска подстроки в строке»

Выполнил: студент группы БВТ1901

Перевозчиков С. В.

Руководитель:

Мелехин А. А.

Москва 2021

Цель работы: изучить методы поиска подстроки в строке и написать их реализацию на одном языке программирования.

Техническое задание:

Задание 1.

Реализовать методы поиска подстроки в строке. Добавить возможность ввода строки и подстроки с клавиатуры. Предусмотреть возможность существования пробела. Реализовать возможность выбора опции чувствительности или нечувствительности к регистру. Оценить время работы каждого алгоритма поиска и сравнить его со временем работы стандартной функции поиска, используемой в выбранном языке программирования.

Алгоритмы:

1. Кнута-Морриса-Пратта
2. Упрощенный Бойера-Мура

Задание 2.

Написать программу, определяющую, является ли данное расположение «Пятнашек» решаемым, то есть можно ли из него за конечное число шагов перейти к правильному. Если это возможно, то необходимо найти хотя бы одно решение - последовательность движений, после которой числа будут расположены в правильном порядке.

Выполнение задания:

Задание 1.

Алгоритм Кнута-Морриса-Пратта.

Код:

public static int KMP(String line, String forSearch)

{

int index = -1, compare = 0;

line = forSearch + "^" + line;

System.out.println(line);

for(int i = forSearch.length() + 1; i < line.length(); i++)

{

String start = "";

String end = "";

for(int j = i; j > 0; j--)

{

if(line.charAt(j) == '^')

{

break;

}

start += line.charAt(i - j);

end = line.charAt(j) + end;

if(start.equals(end))

{

compare = start.length();

}

if(compare == forSearch.length())

{

index = j - forSearch.length() - 1;

return index;

}

}

}

return index;

}

Упрощенный алгоритм Бойера-Мура.

Код:

public static int BM(String line, String forSearch)

{

int index = -1;

int[] d = new int[256];

for(int i = 0; i < d.length; i++)

{

d[i] = forSearch.length();

}

for(int i = forSearch.length() - 1; i >= 0; i--)

{

d[forSearch.charAt(i)] = forSearch.length() - 1 - i;

}

for(int i = forSearch.length() - 1; i < line.length();)

{

if(line.charAt(i) == forSearch.charAt(forSearch.length() - 1))

{

boolean isFounded = true;

int shift = forSearch.length() - 1;

for(int j = 0; j < forSearch.length(); j++)

{

if(forSearch.charAt(forSearch.length() - 1 - j) != line.charAt(i - j))

{

shift = forSearch.length() - 1 - j;

isFounded = false;

break;

}

}

if(isFounded)

{

index = i - (forSearch.length() - 1);

break;

}

else

{

i+= d[forSearch.charAt(shift)];

}

}

i+= d[line.charAt(i)];

}

return index;

}

Задание 2.

Код реализации доски для пятнашек, проверки «решаемости» начальной позиции и нахождения решения:

public class TagGame

{

public static int[][] board = new int[4][4];

public static boolean[][] accessBoard = new boolean[4][4];

public static int zeroX;

public static int zeroY;

public static int numberX;

public static int numberY;

public static int moves = 0;

public static void main(String[] args)

{

try

{

Scanner input = new Scanner(System.in);

while(true)

{

System.out.println("Input start position of board: ");

String line = input.nextLine();

String[] numbers = line.split(" ");

if(numbers.length == 16)

{

int index = 0;

boolean correct = true, hasEmpty = false;

for(int i = 0; i < 4; i++)

{

for(int j = 0; j < 4; j++)

{

int number = Integer.parseInt(numbers[index]);

if(number > 15 || number < 0)

{

correct = false;

break;

}

if(number == 0)

{

hasEmpty = true;

zeroX = j;

zeroY = i;

}

board[i][j] = number;

index++;

}

if(!correct)

{

break;

}

}

if(correct)

{

if(hasEmpty)

{

break;

}

else

{

System.out.println("Start position of board has no empty space");

}

}

else

{

System.out.println("Start position of board has wrong number");

}

}

else

{

System.out.println("Start position of board has wrong count of numbers");

}

}

System.out.println(Arrays.deepToString(board));

System.out.println(canSolve());

solve();

}

catch(Exception ex)

{

ex.printStackTrace();

}

}

public static boolean canSolve()

{

int N = 0;

for(int k = 0; k < 16; k++)

{

int number = board[k / 4][k % 4];

if(number == 0)

{

N += (k / 4) + 1;

}

for(int i = k / 4, j = k % 4; j < 4; j++)

{

if(board[i][j] < number && board[i][j] != 0)

{

N++;

}

}

for(int i = (k / 4) + 1; i < 4; i++)

{

for(int j = 0; j < 4; j++)

{

if(board[i][j] < number && board[i][j] != 0)

{

N++;

}

}

}

}

if(N % 2 == 1)

{

return false;

}

System.out.println(N);

return true;

}

public static void moveUp()

{

board[zeroY][zeroX] = board[zeroY-1][zeroX];

board[zeroY-1][zeroX] = 0;

zeroY--;

moves++;

}

public static void moveDown()

{

board[zeroY][zeroX] = board[zeroY+1][zeroX];

board[zeroY+1][zeroX] = 0;

zeroY++;

moves++;

}

public static void moveLeft()

{

board[zeroY][zeroX] = board[zeroY][zeroX-1];

board[zeroY][zeroX-1] = 0;

zeroX--;

moves++;

}

public static void moveRight()

{

board[zeroY][zeroX] = board[zeroY][zeroX+1];

board[zeroY][zeroX+1] = 0;

zeroX++;

moves++;

}

public static void stepUp()

{

boolean changed = false;

if(zeroY < numberY && !changed)

{

moveDown();

numberY--;

changed = true;

}

if(zeroX < numberX && !changed)

{

if(!accessBoard[zeroY-1][zeroX] && !accessBoard[zeroY-1][zeroX+1])

{

moveUp();

moveRight();

moveDown();

numberY--;

changed = true;

}

}

if(zeroX > numberX && !changed)

{

if(!accessBoard[zeroY-1][zeroX] && !accessBoard[zeroY-1][zeroX-1])

{

moveUp();

moveLeft();

moveDown();

numberY--;

changed = true;

}

}

if(zeroY > numberY && !changed)

{

if(zeroX + 1 < 4)

{

moveRight();

moveUp();

moveUp();

moveLeft();

moveDown();

numberY--;

changed = true;

}

else

{

if(!accessBoard[zeroY][zeroX-1] && !accessBoard[zeroY-1][zeroX-1] && !accessBoard[zeroY-2][zeroX-1] && !accessBoard[zeroY-2][zeroX])

{

moveLeft();

moveUp();

moveUp();

moveRight();

moveDown();

numberY--;

changed = true;

}

}

}

}

public static void stepDown()

{

boolean changed = false;

if(zeroY > numberY && !changed)

{

moveUp();

numberY++;

changed = true;

}

if(zeroX < numberX && !changed)

{

moveDown();

moveRight();

moveUp();

numberY++;

changed = true;

}

if(zeroX > numberX && !changed)

{

moveDown();

moveLeft();

moveUp();

numberY++;

changed = true;

}

if(zeroY < numberY && !changed)

{

if(zeroX + 1 < 4)

{

moveRight();

moveDown();

moveDown();

moveLeft();

moveUp();

numberY++;

changed = true;

}

else

{

if(!accessBoard[zeroY][zeroX-1] && !accessBoard[zeroY+1][zeroX-1] && !accessBoard[zeroY+2][zeroX-1])

{

moveLeft();

moveDown();

moveDown();

moveRight();

moveUp();

numberY++;

changed = true;

}

}

}

}

public static void stepRight()

{

boolean changed = false;

if(zeroX > numberX && !changed)

{

moveLeft();

numberX++;

changed = true;

}

if(zeroY < numberY && !changed)

{

moveRight();

moveDown();

moveLeft();

numberX++;

changed = true;

}

if(zeroY > numberY && !changed)

{

moveRight();

moveUp();

moveLeft();

numberX++;

changed = true;

}

if(zeroX < numberX && !changed)

{

if(zeroY + 1 < 4)

{

moveDown();

moveRight();

moveRight();

moveUp();

moveLeft();

numberX++;

changed = true;

}

else

{

if(!accessBoard[zeroY-1][zeroX] && !accessBoard[zeroY-1][zeroX+1] && !accessBoard[zeroY-1][zeroX+2])

{

moveUp();

moveRight();

moveRight();

moveDown();

moveLeft();

numberX++;

changed = true;

}

}

}

}

public static void stepLeft()

{

boolean changed = false;

if(zeroX < numberX && !changed)

{

moveRight();

numberX--;

changed = true;

}

if(zeroY < numberY && !changed)

{

if(!accessBoard[zeroY+1][zeroX-1] && !accessBoard[zeroY][zeroX-1])

{

moveLeft();

moveDown();

moveRight();

numberX--;

changed = true;

}

}

if(zeroY > numberY && !changed)

{

if(!accessBoard[zeroY-1][zeroX-1] && !accessBoard[zeroY][zeroX-1])

{

moveLeft();

moveUp();

moveRight();

numberX--;

changed = true;

}

}

if(zeroX > numberX && !changed)

{

if(zeroY + 1 < 4)

{

moveDown();

moveLeft();

moveLeft();

moveUp();

moveRight();

numberX--;

changed = true;

}

else

{

if(!accessBoard[zeroY][zeroX-2] && !accessBoard[zeroY-1][zeroX-2] && !accessBoard[zeroY-1][zeroX-1] && !accessBoard[zeroY-1][zeroX])

{

moveUp();

moveLeft();

moveLeft();

moveDown();

moveRight();

numberX--;

changed = true;

}

}

}

}

public static void solve()

{

int i0 = 0, j0 = 0, number = 1;

boolean isSolved = false;

boolean isLineFinnished = false;

while(!isSolved)

{

//ведем поиск плитки с числом number

boolean isFound = false;

for(int i = i0; i < 4; i++)

{

for(int j = j0; j < 4; j++)

{

if(board[i][j] == number)

{

isFound = true;

numberX = j;

numberY = i;

break;

}

}

if(isFound)

{

break;

}

}

//вывод координат искомого числа

System.out.print("Coords of number: ");

System.out.print(numberX);

System.out.print(" , ");

System.out.println(numberY);

//перемещение нулевой плитки к искомой

while(true)

{

if(zeroY+1 < 4)

{

if(board[zeroY+1][zeroX] == number)

{

System.out.println("on down");

System.out.println(numberX);

System.out.println(numberY);

break;

}

}

if(zeroY-1 >= 0)

{

if(board[zeroY-1][zeroX] == number)

{

System.out.println("on up");

System.out.println(numberX);

System.out.println(numberY);

break;

}

}

if(zeroX+1 < 4)

{

if(board[zeroY][zeroX+1] == number)

{

System.out.println("on right");

System.out.println(numberX);

System.out.println(numberY);

break;

}

}

if(zeroX-1 >= 0)

{

if(board[zeroY][zeroX-1] == number)

{

System.out.println("on left");

System.out.println(numberX);

System.out.println(numberY);

break;

}

}

boolean isMoved = false;

if(numberX > zeroX && !isMoved)

{

moveRight();

isMoved = true;

}

if(numberX < zeroX && !isMoved && !accessBoard[zeroY][zeroX-1])

{

moveLeft();

isMoved = true;

}

if(numberY > zeroY && !isMoved)

{

moveDown();

isMoved = true;

}

if(numberY < zeroY && !isMoved && !accessBoard[zeroY-1] [zeroX])

{

moveUp();

isMoved = true;

}

}

System.out.println("zero has been moved to required number");

int requiredX = (number - 1) % 4, requiredY = (number - 1) / 4;

System.out.print("Req coords x - ");

System.out.print(requiredX);

System.out.print(" , y - ");

System.out.println(requiredY);

//перемещение плитки на нужную позицию

System.out.print("Start moving - ");

System.out.println(Arrays.deepToString(board));

boolean isSet = false;

while((numberX != requiredX || numberY != requiredY) && moves < 500)

{

if(!isLineFinnished && (number == 4 || number == 8))

{

while(true)

{

if(numberY-1 >= 0)

{

if(board[numberY-1][numberX] == number-1)

{

break;

}

}

if((number-2) % 4 > numberX)

{

stepRight();

}

if((number-2) % 4 < numberX)

{

stepLeft();

}

if((number-2) / 4 < numberY)

{

stepUp();

}

if(numberX == requiredX && numberY == requiredY)

{

isSet = true;

break;

}

}

if(isSet)

{

break;

}

if(zeroX < numberX)

{

moveDown();

moveRight();

}

if(zeroX > numberX)

{

moveDown();

moveLeft();

}

accessBoard[(number-2) / 4][(number-2) % 4] = false;

moveUp();

moveRight();

moveDown();

moveLeft();

moveUp();

moveRight();

moveUp();

moveLeft();

moveDown();

moveRight();

moveDown();

moveLeft();

moveUp();

moveUp();

moveRight();

moveDown();

accessBoard[(number-2) / 4][(number-2) % 4] = true;

break;

}

if(isLineFinnished && (number == 13 || number == 14))

{

while(true)

{

if(numberX-1 >= 0)

{

if(board[numberY][numberX-1] == number-4)

{

break;

}

}

if((number-5) % 4 > numberX)

{

stepRight();

}

if((number-5) % 4 < numberX)

{

stepLeft();

}

if((number-5) / 4 < numberY)

{

stepUp();

}

if((number-5) / 4 > numberY)

{

stepDown();

}

if(numberX == requiredX && numberY == requiredY)

{

isSet = true;

break;

}

}

if(isSet)

{

break;

}

if(zeroY < numberY)

{

moveRight();

moveDown();

}

if(zeroY > numberY)

{

moveRight();

moveUp();

}

accessBoard[(number-5) / 4][(number-5) % 4] = false;

moveLeft();

moveDown();

moveRight();

moveUp();

moveLeft();

moveDown();

moveLeft();

moveUp();

moveRight();

moveDown();

moveRight();

moveUp();

moveLeft();

moveLeft();

moveDown();

moveRight();

accessBoard[(number-5) / 4][(number-5) % 4] = true;

break;

}

if(requiredX > numberX)

{

System.out.print("Num x - ");

System.out.println(numberX);

stepRight();

System.out.print("Right - ");

System.out.println(Arrays.deepToString(board));

}

if(requiredX < numberX)

{

System.out.print("Num x - ");

System.out.println(numberX);

stepLeft();

System.out.print("Left - ");

System.out.println(Arrays.deepToString(board));

}

if(requiredY > numberY)

{

System.out.print("Num y - ");

System.out.println(numberY);

stepDown();

System.out.print("Down - ");

System.out.println(Arrays.deepToString(board));

}

if(requiredY < numberY)

{

System.out.print("Num y - ");

System.out.println(numberY);

stepUp();

System.out.print("Up - ");

System.out.println(Arrays.deepToString(board));

}

System.out.print("Moves - ");

System.out.println(moves);

}

accessBoard[requiredY][requiredX] = true;

//выбираем следующее число

if(number == 4 || number == 8 || number == 12)

{

isLineFinnished = true;

i0++;

}

if(number == 13 || number == 14 || number == 15)

{

isLineFinnished = false;

j0++;

}

if(!isLineFinnished)

{

number = i0\*4 + requiredX + 2;

}

else

{

if(number == 4 || number == 8 || number == 12)

{

number += i0;

}

else

{

number += 4;

}

}

System.out.print("Req number = ");

System.out.println(number);

if(number > 15)

{

isSolved = true;

}

}

System.out.println(Arrays.deepToString(board));

System.out.println(moves);

}

}

Вывод: были изучены методы поиска подстроки в строке и написана их реализация на языке программирования Java.