Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

**Лабораторна робота №** 7

по дисципліні «Теорія алгоритмів»

на тему: Динамічне програмування. Жадібні алгоритми.

|  |  |
| --- | --- |
| Виконали:  студент групи ІА-з91  Москаленко В.В.  Литвиненко О.Б.  Рибачук Б.В.  Душар А.Р.  Дата здачі \_\_\_\_\_\_\_\_\_\_\_\_  Захищено з балом \_\_\_\_\_\_\_\_\_\_\_ | Перевірив:  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Київ 2020

1. **Задача**

|  |
| --- |
| **Трикутник**.  Дано трикутник, який складається з чисел, наприклад:  7 3 8 8 1 0 2 7 4 4 4 5 2 6 5  Необхідно побудувати маршрут від вершини трикутника до основи рухаючись лише по діагоналі та знайти при цьому:   1. Найбільшу суму чисел на маршруті 2. Найменшу суму чисел 3. Обчислити (1) і (2) для інших вершин |

1. **Постановка проблеми.**

Нам відомо про трикутник який складається з чисел. Необхідно порахувати найбільшу і найменшу суму чисел на маршруті від вершини трикутника до основи рухаючись лише по діагоналі. Розв’язок буде однозначним.

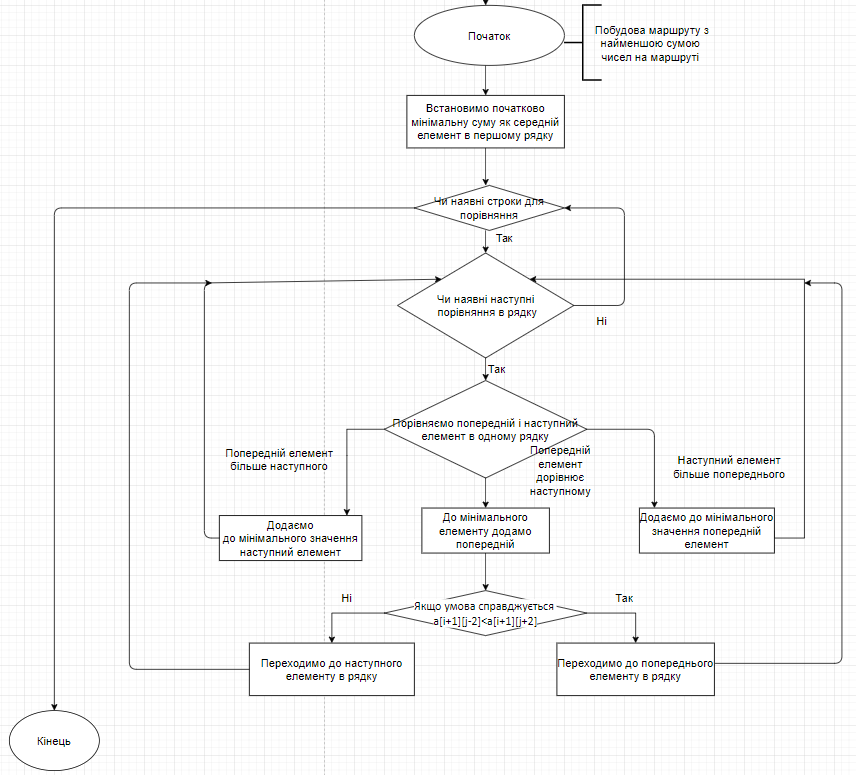
1. **Побудова моделі.**

Даний трикутник представимо у вигляді матриці. Проміжки між числами заповнимо числом -1, або на противагу цій матриці використаємо ран домно заповнену матрицю. За допомогою функції мінімальної та максимальної суми знайдемо відповідну кількість чисел.

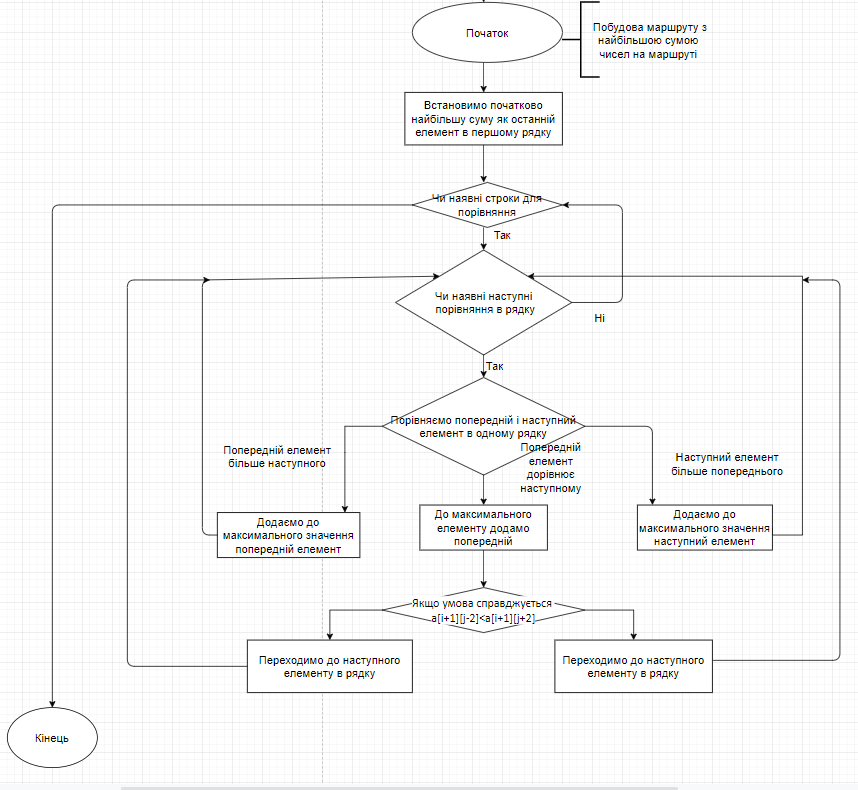
1. **Розроблення алгоритму.**

Алгоритм пошуку найменшої суми чисел включає в себе: циклічний перебір усіх рядків матриці, порівняння чи не вийшли ми за межі рядку і підрахунок мінімальної суми елементів. Порівнюючи попередні значення з наступним значенням за допомогою двух вкладених операторів ми розуміємо чи рухаємось ми в сторону найменшої суми або ні. Даний алгоритм нагадує і має пряму схожість з алгоритмом Дейкстри є стандартним у вирішенні проблем даного типу. За схожою схемою розроблений алгоритм пошуку найбільшої суми чисел на маршруті.

Побудова маршруту з найменшою сумою чисел на маршруті:



Побудова маршруту з найбільшою сумою чисел на маршруті:



1. **Перевірка правильності алгоритму.**

Даний алгоритм працює для всіх коректних вхідних даних. Алгоритм пошуку максимального елементу включає в себе: зовнішній цикл, який перебирає кількість рядків, вкладенні в нього умовний оператор з перевіркою не виходу за межі матриці і перевірка між собою попереднього елементу і наступного елементу. У випадку, якщо елементи рівні, то ми додаємо до максимальної суми попередній елемент. Якщо кількість строк не закінчилась ми перевіряємо між собою 2 елементи, що стоять на відстані 3ох елементів одне від одного, і в залежності від конкретного порівняння ми або інкрементуємо змінну, що вказує на порядковий номер елемента в рядку, або декрементуємо. У випадку, якщо попередній і наступні елементи не співпадають, то ми інкрементуємо найбільшу суму чисел на маршруті за допомогою найбільшого з цих двох елементів.

1. **Реалізація алгоритму**.

import java.io.\*;

import java.util.Random;

public class Lab7

{

public static void main(String[] args) {

Random random = new Random();

int a[][] = new int[][]

{{-1, -1, -1, -1, 7, -1, -1, -1, -1},

{-1, -1, -1, 3, -1, 8, -1, -1, -1},

{-1, -1, 8, -1, 1, -1, 0, -1, -1},

{-1, 2, -1, 7, -1, 4, -1, 4,-1},

{4, -1, 5,-1, 2, -1, 6, -1, 5}};

int min=sumMin(a);

int max=sumMax(a);

System.out.println("Max: "+max+"\nMin: "+min);

for (int i=0;i<5;i++){

for (int j=0;j<9;j++){

if (a[i][j]!=-1){

a[i][j]=random.nextInt((10)+1);

}

}

}

for (int i=0;i<5;i++){

for (int j=0;j<9;j++){

System.out.print(a[i][j]+" ");

}

System.out.println();

}

min=sumMin(a);

max=sumMax(a);

System.out.println("Max: "+max+"\nMin: "+min);

}

public static int sumMin(int a[][]){

int j=4;

int min=a[0][j];

for (int i = 1; i < 5; i++){

if (j!=8 || j!=0){

if (a[i][j-1]==a[i][j+1]){

min+=a[i][j-1];

if (i!=4){

if (a[i+1][j-2]<a[i+1][j+2]){

j--;

}else

if (a[i+1][j-2]>a[i+1][j+2]){

j++;

}

}

}else

if (a[i][j-1]<a[i][j+1]){

min+=a[i][j-1];

j--;

}else

if (a[i][j-1]>a[i][j+1]){

min+=a[i][j+1];

j++;

}

}

}

return min;

}

public static int sumMax(int a[][]){

int j=4;

int max=a[0][j];

for (int i = 1; i < 5; i++){

if (j!=0 || j!=8){

if (a[i][j-1]==a[i][j+1]){

max+=a[i][j-1];

if (i!=4){

if (a[i+1][j-2]<a[i+1][j+2]){

j++;

}else

if (a[i+1][j-2]>a[i+1][j+2]){

j--;

}

}

}else

if (a[i][j-1]<a[i][j+1]){

max+=a[i][j+1];

j++;

}else

if (a[i][j-1]>a[i][j+1]){

max+=a[i][j-1];

j--;

}

}

}

return max;

}

}

**7. Перевірка програми.**

Дана програма перевірена за допомогою захардкодженої матриці а також за допомогою ран домно створеної матриці з натуральних чисел.

**8.Висновок**

Даний алгоритм працює для всіх коректних вхідних даних. Алгоритм пошуку максимального елементу включає в себе: зовнішній цикл, який перебирає кількість рядків, вкладенні в нього умовний оператор з перевіркою не виходу за межі матриці і перевірка між собою попереднього елементу і наступного елементу. У випадку, якщо елементи рівні, то ми додаємо до максимальної суми попередній елемент.

**9. Вивід програми**

