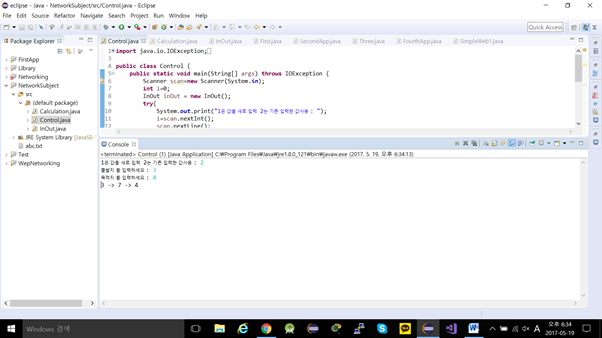
다익스트라 알고리즘

1. 실행 장면



2. 소스코드

1. control

import java.io.IOException;

import java.util.Scanner;

public class Control {

public static void main(String[] args) throws IOException {

Scanner scan=new Scanner(System.in);

int i=0;

InOut inOut = new InOut();

try{

System.out.print("1은 값을 새로 입력 2는 기존 입력한 값사용 : ");

i=scan.nextInt();

scan.nextLine();

}catch (Exception e) {

System.out.println("올바른 값을 입력하시오");

}

switch (i) {

case 1:

inOut.input();

case 2:

inOut.ouput();

break;

default:

break;

}

}

}

2. Calculation

import java.awt.List;

import java.util.ArrayList;

import java.util.Arrays;

public class Calculation {

private int number;

private int[][] inputs;

private int inf = 999999999;

// start node부터 i node까지의 최단거리

private int[] dist;

// 각 노드별로 방문한 적이 있는지 표시

private boolean[] visited;

// 각노드 번호

private int[] prev;

ArrayList<String> ps = new ArrayList<String>();

public Calculation(int number, int[][] inputs) {

this.number = number;

this.inputs = inputs;

dist = new int[number];

visited = new boolean[number];

prev = new int[number];

Arrays.fill(visited, false);

Arrays.fill(prev, 0);

Arrays.fill(dist, 999999999);

}

public void process() {

ps.add("비용값 : " + Arrays.toString(dist) + " 확정여부 " + Arrays.toString(visited));

}

public void ssp(int start) {

dist[start] = 0; // 최초 시작 값 distance 초기화

prev[start] = 0;

for (int j = 0; j < number; j++) { // dist 값을 한번 초기화 했으므로 n-1번만 진행

int check=-1;

int min = inf + 1; // dist 최소값 찾기 위한 임시 값 초기화

int index = -1;

for (int k = 0; k < number; k++) {

if ((visited[k] == false) && (min > dist[k])) {

min = dist[k];

index = k;

}

}

visited[index] = true;

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

if (inputs[index][i] != 0 && dist[i] > dist[index] + inputs[index][i]) {

// 인접 행렬을 검사하여 최적의 값을 찾아

dist[i] = dist[index] + inputs[index][i];

prev[i] = index + 1;

//System.out.println("dist[i] : " + dist[i] + " i : " + i + " prev[i] : " + prev[i]+" j :"+j);

process();

check=1;

}

}

if(check==-1){ // 변동사항이 없을때

process();

}

}

}

public String router(int start, int destnation) {

String rout = Integer.toString(destnation + 1);

while(prev[destnation]!=0){

rout = Integer.toString(prev[destnation]) + " -> " + rout;

destnation = prev[destnation] - 1;

}

return rout;

}

public int[][] getInputs() {

return inputs;

}

public int[] getprev() {

return prev;

}

public void setInputs(int[][] inputs) {

this.inputs = inputs;

}

public int getInf() {

return inf;

}

public void setInf(int inf) {

this.inf = inf;

}

public int[] getdist() {

return dist;

}

}

3. InOut

import java.io.BufferedReader;

import java.io.FileNotFoundException;

import java.io.FileReader;

import java.io.IOException;

import java.io.PrintWriter;

import java.util.Arrays;

import java.util.Scanner;

import java.util.StringTokenizer;

public class InOut {

Scanner scan = new Scanner(System.in);

Calculation cal;

public void ouput() throws IOException {

BufferedReader br = new BufferedReader(new FileReader("d:/data.txt"));

int number=0,start,destnation;

String line=br.readLine();

StringTokenizer tokens=new StringTokenizer(line, " ");

number= Integer.parseInt(tokens.nextToken());

int [][] inputs=new int[number][number];

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

line=br.readLine();

tokens=new StringTokenizer(line, " ");

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

inputs[i][j] = Integer.parseInt(tokens.nextToken());

}

}

this.cal= new Calculation(number,inputs);

try{

System.out.print("출발지 를 입력하세요 : ");

start=scan.nextInt();

scan.nextLine();

System.out.print("목적지 를 입력하세요 : ");

destnation=scan.nextInt();

scan.nextLine();

}catch (Exception e) {

System.out.println("올바른 값을 입력해 주세요");

return;

}

cal.ssp(start-1);

String router=cal.router(start-1,destnation-1);

System.out.println(router);

br.close();

resultWriter(cal.getdist(), router);

processWriter();

}

public void processWriter() throws FileNotFoundException{

PrintWriter pw = new PrintWriter("d:/process.txt");

for(int i=0;i<cal.ps.size();i++){

pw.println((i+1) + " 번째 진행현황입니다.");

pw.println(cal.ps.get(i).toString() );

}

pw.close();

}

public void input() throws IOException {

int nodeNumber, edgeNumber;

try {

System.out.print("노드 개수 입력 : ");

nodeNumber = scan.nextInt();

scan.nextLine();

System.out.print("에지 개수 입력 : ");

edgeNumber = scan.nextInt();

scan.nextLine();

} catch (Exception e) {

System.out.println("올바른 값을 입력해주세요.");

return;

}

tableWriter(nodeNumber, edgeNumber);

}

public int[][] initialization(int nodeNumber) {

int[][] inputs = new int[nodeNumber][nodeNumber];

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

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

inputs[i][j] = 999999999;

}

}

return inputs;

}

public void resultWriter(int[] dist,String router) throws FileNotFoundException{

PrintWriter pw = new PrintWriter("d:/result.txt");

pw.println("결과값 데이터 입니다 .");

pw.println("경로 값 입니다 : "+router);

pw.println("각 경로 비용 입니다 : "+Arrays.toString(dist));

pw.println("각 노드의 경로값 : " +Arrays.toString(cal.getprev()));

pw.close();

}

public void tableWriter(int nodeNumber, int edgeNumber) throws IOException {

PrintWriter pw = new PrintWriter("d:/data.txt");

int[][] inputs = initialization(nodeNumber);

int start, arrive;

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

System.out.print("출발 도착 가중치 값을 입력하세요 ");

try {

start = scan.nextInt();

arrive = scan.nextInt();

// 초기화

inputs[start - 1][arrive - 1] = scan.nextInt();

inputs[arrive - 1][start - 1] = inputs[start - 1][arrive - 1];

scan.nextLine();

} catch (Exception e) {

System.out.println("올바른 값을 입력하세요");

return;

}

}

String data;

pw.println(nodeNumber+" ");

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

data = "";

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

data += String.valueOf(inputs[i][j]+" ");

}

pw.println(data);

}

pw.close();

}

}

1. 출력된 txt파일 결과

1. data

8

0 2 999999999 999999999 999999999 3 999999999 999999999

2 0 4 1 999999999 999999999 999999999 999999999

999999999 4 0 999999999 999999999 999999999 3 999999999

999999999 1 999999999 0 3 999999999 2 999999999

999999999 999999999 3 3 0 999999999 999999999 4

3 999999999 999999999 999999999 999999999 0 6 999999999

999999999 999999999 999999999 2 999999999 6 0 4

999999999 999999999 999999999 999999999 4 999999999 4 0

2. result

결과값 데이터 입니다 .

경로 값 입니다 : 3 -> 7 -> 4

각 경로 비용 입니다 : [6, 4, 0, 5, 8, 9, 3, 7]

각 노드의 경로값 : [2, 3, 0, 7, 4, 7, 3, 7]

3. process

1 번째 진행현황입니다.

비용값 : [999999999, 4, 0, 999999999, 999999999, 999999999, 999999999, 999999999] 확정여부 [false, false, true, false, false, false, false, false]

2 번째 진행현황입니다.

비용값 : [999999999, 4, 0, 999999999, 999999999, 999999999, 3, 999999999] 확정여부 [false, false, true, false, false, false, false, false]

3 번째 진행현황입니다.

비용값 : [999999999, 4, 0, 5, 999999999, 999999999, 3, 999999999] 확정여부 [false, false, true, false, false, false, true, false]

4 번째 진행현황입니다.

비용값 : [999999999, 4, 0, 5, 999999999, 9, 3, 999999999] 확정여부 [false, false, true, false, false, false, true, false]

5 번째 진행현황입니다.

비용값 : [999999999, 4, 0, 5, 999999999, 9, 3, 7] 확정여부 [false, false, true, false, false, false, true, false]

6 번째 진행현황입니다.

비용값 : [6, 4, 0, 5, 999999999, 9, 3, 7] 확정여부 [false, true, true, false, false, false, true, false]

7 번째 진행현황입니다.

비용값 : [6, 4, 0, 5, 8, 9, 3, 7] 확정여부 [false, true, true, true, false, false, true, false]

8 번째 진행현황입니다.

비용값 : [6, 4, 0, 5, 8, 9, 3, 7] 확정여부 [true, true, true, true, false, false, true, false]

9 번째 진행현황입니다.

비용값 : [6, 4, 0, 5, 8, 9, 3, 7] 확정여부 [true, true, true, true, false, false, true, true]

10 번째 진행현황입니다.

비용값 : [6, 4, 0, 5, 8, 9, 3, 7] 확정여부 [true, true, true, true, true, false, true, true]

11 번째 진행현황입니다.

비용값 : [6, 4, 0, 5, 8, 9, 3, 7] 확정여부 [true, true, true, true, true, true, true, true]