哈尔滨工业大学

数据科学与大数据技术专业

实验报告

课程名称：计算机算法基础

实验项目：动态规划

实验题目：动态规划算法的设计与实现

实验日期：2019.11.22

班级：1534701

学号姓名：1153470114 薛博洋

1153470104 沈孙乐

|  |  |  |
| --- | --- | --- |
| 设计成绩 | 报告成绩 | 指导老师 |
|  |  | 张岩 |

**一、实验目的**

动态规划是解决优化问题的一种常见方法。动态规划法在求解问题中，对于每一步决策，列出各种可能的局部解，再依据某种判定条件，舍弃那些肯定不能得到最优解的局部解，在每一步都经过筛选，以每一步都是最优解来保证全局是最优解。

**二、实验要求及实验环境**

1. 设计多段图的最短路径问题的动态规划算法，并证明其具有最优子结构性质和重叠子问题性质，以及算法的正确性，写在实验报告中。

2. 采用邻接矩阵存储多段图，实现多段图最短路径和最短路径长度的动态规划算法。

3. 采用邻接表存储多段图，实现多段图最短路径和最短路径长度的动态规划算法。

4. 比较两种存储方法所实现的算法的时空性能。

**三、设计的基本思想和算法的原理**

**四、实验结果与分析**

**五、经验体会与不足**

1、

2、

**六、附录：源代码（带注释）**

利用邻接矩阵实现的算法：

#include <stdio.h>

#include <stdlib.h>

#define N 8

#define Z 5000

#define inf 100000

/\*define a function that give 0 randomly\*/

int givezero(){

return (rand() % 4);

}

/\*the function to form a graph in a matrix

\*/

void makeG(int graph[][N]){

/\*make the n\*n graph\*/

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

graph[i][i] = 0;

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

if(givezero()){

graph[i][j] = -1 \* rand() % 8;

graph[j][i] = -1 \* graph[i][j];

}else {

graph[i][j] = inf;

graph[j][i] = graph[i][j];

}

}

}

}

void printG(int graph[][N]){

/\*print the matrix in a matrix way\*/

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

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

printf("%d\t", graph[i][j]);

}

printf("\n");

}

printf("\n");

}

/\*the matrix to save the closest distance between every 2 points\*/

void initial(int d[][N]){

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

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

d[i][j] = Z;

}

d[i][i] = 0;

}

}

/\*give an array, record the node on the closest road

\*and give a function to print the shortest road

\*/

int node[N];

void printRoad(int n[]){

int before = 0;

printf("the road: 0\t");

while(before != N-1){

printf("%d\t", n[before]);

before = n[before];

}

}

/\*the function to find the shortest way

\*input:graph matrix, output:length of the road between a & b

\*/

int dijkstra(int g[][N], int d[][N], int a, int b){

/\*dijkstra algoritm

\*find out the shortest way between pointer a & b

\*int dis is the steps between them

\*/

int distance = 1000;

int shortest = -1;

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

/\*make sure the direction is right\*/

if(g[a][i] > 0 && g[a][i] < 200){

/\*get the distance of i & b\*/

int dis\_i;

if(d[i][b] == Z){

dis\_i = dijkstra(g, d, i, b);

}else{

dis\_i = d[i][b];

}

/\*test if any could be shorter in set of d(a-i) + d(i-b)\*/

if(distance > (g[a][i] + dis\_i)){

distance = g[a][i] + dis\_i;

shortest = i;

}

}

}

node[a] = shortest;

d[a][b] = distance;

return distance;

}

int main(void){

/\* initialize the graph matrix g\*/

int g[N][N];

makeG(g);

g[0][N-1] = 50;

/\*give a fixed example\*/

int g2[8][8] = {

{0, 1, 2, 5, inf, inf, inf, inf},

{inf, 0, inf, inf, 4, 11, inf, inf},

{inf, inf, 0, inf, 9, 5, 16, inf},

{inf, inf, inf, 0, inf, inf, 2, inf},

{inf, inf, inf, inf, 0, inf, inf, 18},

{inf, inf, inf, inf, inf, 0, inf, 1},

{inf, inf, inf, inf, inf, inf, 0, 2},

{inf, inf, inf, inf, inf, inf, inf, 0},

};

/\*initial distance matrix d\*/

int d[N][N];

initial(d);

/\*show the graph, do the dijkstra, show the distance matrix, print result\*/

printG(g);

int length = dijkstra(g2, d, 0, N-1);

printG(d);

/\*print the result\*/

printf("the length of road: %d\n", length);

printRoad(node);

}