JAVA指导:动态规划方法 PDF转换可能丢失图片或格式，建议阅读原文
https://www.100test.com/kao_ti2020/144/2021_2022_JAVA_E6_8C_87_E5_AF_BC_c104_144977.htm 动态规划 是 最优化原理 中的一种重要的方法。 动态规划在查找有很多 重叠子问题 的情况的最优解时有效。它将问题重新组合成子问题。为了避免多次解决这些子问题，它们的结果都逐渐被计算并被保存，从简单的问题直到整个问题都被解决。因此，动态规划保存递归时的结果，因而不会在解决同样的问题时花费时间。 动态规划只能应用于有 最优子结构 的问题。最优子结构的意思是局部最优解能决定全局最优解。简单地说，问题能够分解成子问题来解决。 总的来说，动态规划的优势在于： 重叠子问题 最优子结构 记忆化 问题描述: 动态规划举例 图10-3（a）示出了一个数字三角形，请编一个程序， 计算从顶至底的某处的一条路劲， 使该路劲所经过的数字的总和最大。 （1） 每一步可沿左斜线向下或右斜线向下； （2） 1＜三角形行数≤100； （3） 三角形中的数字为0，1，……99。 输入数据： 由input.txt文件中首先读到的是三角形的行数， 在例子中input.txt表示如图13-3（b）. 7 3 8 8 1 0 2 7 4 4 4 5 2 6 5 5 6 9 5 9 8思路: 1.算出每个节点的规划值(也就是待比较的最大值),并记录搜索路径 2.取三角形底边所有规划值中的最大值 3.输出路径 主程序 import java.util. * . /** * 用动态规划法求出最优路径 * @author zqc * */ public class dynamicsolvetrianglepath { private string[][] str_triangle = null . private node[][] triangle_nodes = null . private list nodes. private list paths. // 节点 static class node { private int value. private list path = new vector(). public list getpath() {

return path. } public void setpath(list p) { path = p. } // n=(0,0) or (0,1) or (2,2) public void addpath(string n) { path.add(n). } public void addpath(list pa) { path.addall(pa). } public int getvalue() { return value. } public void setvalue(int value) { this .value = value. } } public dynamicsolvetrianglepath() { initnodes(). findpath(). } // 从根节点开始,逆向推解出到达所有节点的最佳路径 private void initnodes(){ this.str_triangle = readtriangle.gettriangle(). triangle_nodes = new node[str_triangle.length][]. nodes = new vector(). for(int i = 0 . i }else{//除边界节点外其他节点 node node1 = triangle_nodes[i-1][j-1]. node node2 = triangle_nodes[i-1][j]. node bigger = max(node1,node2). list ph = bigger.getpath(). n.addpath(ph). n.addpath(currentpath). int val = bigger.getvalue() c(str_triangle[i][j]). n.setvalue(val). } triangle_nodes[i][j] = n. n = null. }//end of for j }//end of for i } private node max(node node1,node node2){ int i1 = node1.getvalue(). int i2 = node2.getvalue(). return i1>i2?node1:node2. } private int c(string s){ return integer.parseint(s.trim()). } //找出最佳路径 private void findpath(){ if(this.triangle_nodes==null)return. int max = 0. int mi = 0. int mj = 0. for(int i = 0 . i max){ max = t. mi = i. mj = j. } } } system.out.println(max path: triangle_nodes[mi][mj].getpath()). system.out.println(max value: max). } public static void main(string[] args)throws exception{ dynamicsolvetrianglepath dsp = new dynamicsolvetrianglepath(). } } 用于读取文件的辅助类 import java.io. * . import java.util. * ./** * 输入文本格式为 * 类似这样一个数字三角形 * * x * x x * x x x * x x x x * x x x x x * *

@author zqc * */ public class readtriangle { public static string triangle_file =c:/java/triangle.txt . public static string[][] gettriangle() { string[][] rtn. try{ file f =new file(readtriangle.triangle_file). bufferedreader br=new bufferedreader(new filereader(f)). list l =new vector(). string line =br.readline(). while (line != null) { l.add(line.trim()). line=br.readline(). } int heigth=l.size(). rtn =new string[heigth][]. for (int i =0 . i结果输出如下: max path:[(0,0), (1,0), (2,0), (3,1), (4,1), (5,2)] max value:39 同样的方法可以解决很多类似的问题,比如,游戏中的最短路径.最优化投资问题.背包问题等等. 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com