广东金融学院实验报告

课程名称：算法分析与设计

装订线

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验编号  及实验名称 | 算法分析与设计实验4 | | | 系 别 | 计算机科学与技术 |
| 姓 名 | 林旋华 | 学 号 | 181543306 | 班 级 | 18计科（3） |
| 实验地点 | 电教503 | 实验日期 | 2020.10.15 | 实验时数 | 2 |
| 指导教师 | 郭艺辉 | 同组其他成员 | 无 | 成 绩 |  |
| 1. 实验目的及要求   1) 掌握动态规划算法的基本思想以及基本原理。  2) 掌握使用动态规划算法求解问题的一般特征以及步骤。  3) 掌握动态规划算法设计方法以及复杂性分析方法。  4) 掌握动态规划法解矩阵连乘问题算法设计思想、设计过程以及程序实现。 | | | | | |
| 1. 实验环境及相关情况（包含使用软件、实验设备、主要仪器及材料等）   1) 操作系统：Windows操作系统  2) 开发工具：Eclipse、JDK  3) 开发语言：Java | | | | | |
| 1. 实验内容及步骤（包含简要的实验步骤流程）   矩阵连乘问题的问题提出是，给定*n*个矩阵｛A1, A2, …, A*n*｝，A*i*的维数为*pi*-1×*pi*，A*i*与A*i*+1是可乘的，*i*=1, 2 , …, *n*-1。如何确定计算矩阵连乘积的计算次序，使得依此次序计算矩阵连乘积需要的数乘次数最少。设有5个矩阵A1A2A3A4A5连乘，找出最优计算次序以使得矩阵连乘所需要的计算次数最少，采用动态规划法解决此问题。将算法编程实现, 并将程序与运算结果填写入实验结果。分析算法的时间复杂性，并用大Ο来表示其时间复杂性。 | | | | | |
| 1. 四、实验结果（包括程序或图表、结论陈述、数据记录及分析等，可附页） 2. 代码如下：   package week6;  public class MatrixMultiply {  static int MN; //表示矩阵链中矩阵的数目  static int[]p; //存放各个矩阵的维数  static int [][][]A;//存放要进行连乘的多个矩阵  static int [][]m; //用来存放Ai到Aj的最少乘次数  static int [][]s; //用来存放Ai到Aj的最后断开位置  //构造函数  public MatrixMultiply() {  MN=0;  int[] p = {5, 200, 2, 100, 30, 200};  }  //构造函数,L为矩阵的数目  public MatrixMultiply(int L) {  MN=5;  int[] p = {5, 200, 2, 100, 30, 200};  A=new int [MN][][];  m=new int [MN+1][MN+1];  s=new int [MN+1][MN+1]; //随机生成连乘矩阵的维数[1-11]    //随机生成各个矩阵  for(int i=0;i<MN;i++)  {  A[i]=new int [p[i]][p[i+1]];  CreatMatrix(A[i],p[i],p[i+1]);  }  }  //创建矩阵a，维数为m\*n 随机数填充数组内容  public void CreatMatrix(int[][] a, int m, int n) {  // TODO Auto-generated method stub  for(int i=0;i<m;i++)  for(int j=0;j<n;j++)  a[i][j]=(int) Math.round(Math.random()\*50)-10;  }  //输出连乘的所有矩阵  public void printAllM()  {  for (int i=0;i<this.MN;i++)  {  System.out.println("A"+(i+1)+": "+A[i].length +"\*"+A[i][0].length );  printM(A[i]);  }  }  //输出单个矩阵的值  public void printM(int[][] a) {  for(int i=0;i<a.length;i++)  {  System.out.print(" ");  for(int j=0;j<a[i].length;j++)  System.out.print(" "+a[i][j]);  System.out.println();  }  }  public static void main(String[]args) {  //构造函数分配内存空间  MatrixMultiply M=new MatrixMultiply(7);  M.printAllM();  int[] p = {5, 200, 2, 100, 30, 200};  M.matrixChain(p,M.m,M.s);  System.out.print("矩阵链所需的最少乘次数为："+M.m[1][M.MN]);  System.out.println();  String []s=new String[M.MN+1];  for(int i=1;i<=M.MN;i++)  {  s[i]="A"+i;  }  M.traceback(M.s,1,M.MN,s);  System.out.print("矩阵连乘最优连乘顺序：");  for(int i=1;i<=M.MN;i++)  {  System.out.print(s[i]);  }  }  public void traceback(int[][] s, int i, int j, String[] c) {  if(i==j)return;  traceback(s,i,s[i][j],c);  traceback(s,s[i][j]+1,j,c);  c[i]="("+c[i];  c[j]=c[j]+")";  System.out.println("矩阵连乘 A"+i+","+s[i][j]+"和 A"+(s[i][j]+1)+","+j);  }  //作用：计算矩阵连乘时，矩阵链的最少乘次数  private void matrixChain(int[] p, int[][] m, int[][] s)  {  int n=p.length-1;//矩阵链长度为1,不需要进行乘运算，即m[i][i]值为0  for(int i=1;i<=n;i++)  m[i][i]=0;  for(int r=2;r<=n;r++)  {  for(int i=1;i<=n-r+1;i++)  {  int j=i+r-1;  m[i][j]=m[i+1][j]+p[i-1]\*p[i]\*p[j];  s[i][j]=i;  for(int k=i+1;k<j;k++)  {  int t=m[i][k]+m[k+1][j]+p[i-1]\*p[k]\*p[j];  if(t<m[i][j])  {  m[i][j]=t;  s[i][j]=k;  }  }  }  }  }  }   1. 运行结果：      1. 时间复杂度分析：   算法matrixChain的主要计算量取决于对r,i和k的3重循环。循环体内的计算量为O(1)，而三重循环的总次数为O(n^3)。因此，该算法的计算时间上界为O(n^3)。 | | | | | |
| 1. 实验总结（包括心得体会、问题回答及实验改进意见，可附页）   通过本次实验，我基本掌握了动态规划算法的基本思想以及基本原理，了解了用动态规划算法求解问题的一般特征以及步骤，学习了动态规划算法设计方法以及复杂性分析方法，同时对动态规划法解矩阵连乘问题算法设计思想、设计过程以及程序实现有了进一步的认识。  在算法优化的过程中，我认识到用动态规划迭代方式解决此问题，可依据其递归式自底向上的方式进行计算。在计算过程中，保存已解决的子问题的答案。每个子问题只计算一次，而在后面需要时只需简单检查一下，从而避免了大量的重复计算，最终得到多项式时间的算法。 | | | | | |
| 六、教师评语  1、完成所有规定的实验内容，实验步骤正确，结果正确；  2、完成绝大部分规定的实验内容，实验步骤正确，结果正确；  3、完成大部分规定的实验内容，实验步骤正确，结果正确；  4、基本完成规定的实验内容，实验步骤基本正确，所完成的结果基本正确；  5、未能很好地完成规定的实验内容或实验步骤不正确或结果不正确。  6、其它：  评定等级：优秀 良好 中等 及格 不及格  教师签名：郭艺辉 | | | | | |