**华中科技大学计算机科学与技术学院**

**算法分析与设计报告**



专 业： 计算机科学与技术

班 级： ACM1701

学 号： U201714850

姓 名： 丁文隆

成 绩：

指导教师： 何琨

**完成日期： 2019年11月17日**

# 实验二

## 实验题目：动态规划算法——编程实现矩阵链乘法

## 二、实验目的与内容

1、实验目的：

了解动态规划算法的一般设计思路，学习如何构造状态转移方程和保存状态。

了解在进行矩阵链乘法时，不同的运算方案（运算先后）所带来的性能差距。

掌握矩阵链乘法问题的具体含义，理解其在实现最优解表时的运算方案和转移方程以及其推导过程。

2、实验内容：

设计并编程实现拥有计算最小乘法次数的矩阵链乘法匹配方案：

输入：第一行输入一个数n，代表要处理的矩阵个数；下面跟着n行，每一行两个数，为矩阵的行数和列数。（p[i].first，p[i].second）

输出：最小的乘法次数，匹配方案。

自行设计测试用例测试数据，测试程序的正确性。

分析算法的正确性与复杂度。

三、算法设计

1、算法描述

实验中我们维护两个动态规划中用于存储状态的表，一个表记为opt[i][j]，用于存储从第i号矩阵到第j号矩阵所用的最小的乘法次数，另一个表记为record[i][j]，存储使得从第i号元素到第j号元素产生最小乘法次数的断点的编号k。设计状态方程和边界条件完成两个表的存储。递归查找record表就能够输出匹配方案。表项opt[0][N-1]存入的是最大解。

STEP 1:

STEP 2:

STEP 3:

STEP 4:

STEP 5:

2、算法流程图

算法描述的流程图如图X.1：

图X.1

## 四、实验环境

操作系统：Windows 10

编译环境：Dev C++ 5.11

编译器 ：TDM-GCC 4.9.2 64-bit Release

## 五、实验过程

通过构造最优解表opt和断点记录表record实现的动态规划算法的源代码如下所示，其中每一块代码的释义、变量名的解释在注释中均有体现。

|  |
| --- |
| 动态规划实现矩阵链乘法 |
| #include <iostream>  #include <utility>  #define INF 100000  #define N 6  using namespace std;  int opt[N][N], record[N][N]; //opt：最优解矩阵，record：用于记录括号位置  pair<int, int> p0[N]; //记录每个矩阵的大小  int p[N+1]; //存储矩阵大小序列  void opt\_generate(int p[N+1], int opt[N][N], int record[N][N]){  for(int i=0;i<N;i++){  opt[i][i]=0; //对角线开销置0  }  for(int i=0;i<N;i++){  record[i][i]=i; //每个元素开始时括号为自己（边界情况，链长为1的情况）  }  /\*构造最优解矩阵\*/  for(int i=2;i<=N;i++){ //i代表当前所处理的矩阵链的长度  for(int j=0;j<=N-i;j++){ //j代表当前链开始的横坐标  int z=j+i-1; //z代表当前链结束的横坐标  opt[j][z]=INF; //初始值为无穷大  for(int k=j;k<z;k++){ //选择断点k，寻找最优解  int opt\_t=opt[j][k]+opt[k+1][z]+p[j]\*p[k+1]\*p[z+1];  if(opt\_t<opt[j][z]){  opt[j][z]=opt\_t;  record[j][z]=k; //保存断点k  }  }  }  }  }  //打印出括号序列  void print(int record[N][N], int i,int j){  if(i==j){  cout<<"A"<<i;  }  else{  cout<<"(";  print(record,i,record[i][j]); //左边寻找  print(record,record[i][j]+1,j); //右边寻找  cout<<")";  }  }  int main(){  for(int i=0;i<N;i++){  cin>>p0[i].first;  cin>>p0[i].second;  }    //如果输入不合法，汇报错误  for(int i=1;i<N;i++){  if(p0[i].first!=p0[i-1].second){  cout<<"This matrix can not multiplication."<<endl;  return 0;  }  }    //生成矩阵大小序列  for(int i=0;i<N;i++){  p[i]=p0[i].first;  }  p[N]=p0[N-1].second;    opt\_generate(p,opt,record);  cout<<"最少要进行的乘法步骤为："<<opt[0][N-1]<<"次"<<endl;  print(record,0,N-1);  return 0;  } |

## 六、算法测试

（请详细描述样例设计理由以及详细样例设计）

测试样例1

|  |  |  |
| --- | --- | --- |
| 样例输入 | 理论输出 | 样例输出 |
| christina  christsina |  |  |

由于理论输出与样例输出相符，所以测试样例1验证成功。

测试样例2

|  |  |  |
| --- | --- | --- |
| 样例输入 | 理论输出 | 样例输出 |
| christina  christsina |  |  |

由于理论输出与样例输出相符，所以测试样例2验证成功。

综上，算法通过所有样例的测试。

## 七、结果分析

（分析算法的正确性、时间复杂度、空间复杂度、实验中遇到的问题等）

## 八、总结

(通过本次实验得到了什么结论，实验体会）

# 实验二（选做）

## 一、实验题目：动态规划算法——编程实现0-1背包问题求解算法

## 二、实验目的与内容

1、实验目的：

掌握动态规划算法的设计思路，学习构造状态转移方程和动态规划表的方法。

了解0-1背包问题的具体含义以及其状态转移方程和状态表的含义和推导过程。

2、实验内容：

设计并编程实现0-1背包问题的算法：

输入：第一行输入两个数M和n，M代表背包容量，n代待选表物品的件数；下面给出n行，每行两个数，给出每件物品的重量w[i]和价值p[i]。

输出：要求求出背包物品所能装入的最大价值。

自行设计测试用例测试数据，测试程序的正确性。

分析算法的正确性与复杂度。

## 三、算法设计

1、算法描述

在实验中，我们构造一个表KNAP[i][j]，表示前i件物品装入j容量背包的最大价值，通过状态转移方程和边界条件完成这张表。背包能装入的最大价值就存储在KNAP[n][M]中。

STEP 1:

STEP 2:

STEP 3:

STEP 4:

STEP 5:

2、算法流程图

算法描述的流程图如图X.1：

图X.1

## 四、实验环境

操作系统：Windows 10

编译环境：Dev C++ 5.11

编译器 ：TDM-GCC 4.9.2 64-bit Release

## 五、实验过程

利用动态规划实现0-1背包问题的源代码如下所示，其中每一部分的意义和变量名释义的已经写在了注释中：

|  |
| --- |
| 动态规划实现0-1背包问题 |
| #include <iostream>  #include <algorithm>  using namespace std;  int main(){  int M; //背包总容量  int n; //物品的件数  cin>>n>>M;    int w[n+5],p[n+5]; //每件物品的重量w[i],价值p[i]  int KNAP[n+5][M+5]; //前i件物品装入j容量的最大价值    for(int i=1;i<=n;i++){ //读入物品重量  cin>>w[i];  }  for(int i=1;i<=n;i++){ //读入物品价值  cin>>p[i];  }    for(int j=0;j<=M;j++){ //初始化边界条件，前0件商品最大价值为0  KNAP[0][j]=0;  }    for(int i=1;i<=n;i++){  for(int j=0;j<=M;j++){  if(w[i]>j){ //如果背包容量不够第i件商品，则一定不装入  KNAP[i][j]=KNAP[i-1][j];  }  else{ //否则选择是否装第i件物品  KNAP[i][j]=max(KNAP[i-1][j],KNAP[i-1][j-w[i]]+p[i]);  }  }  }  cout<<"背包能装入的最大价值为："<<KNAP[n][M];  return 0;  } |

## 六、算法测试

（请详细描述样例设计理由以及详细样例设计）

测试样例1

|  |  |  |
| --- | --- | --- |
| 样例输入 | 理论输出 | 样例输出 |
| christina  christsina |  |  |

由于理论输出与样例输出相符，所以测试样例1验证成功。

测试样例2

|  |  |  |
| --- | --- | --- |
| 样例输入 | 理论输出 | 样例输出 |
| christina  christsina |  |  |

由于理论输出与样例输出相符，所以测试样例2验证成功。

综上，算法通过所有样例的测试。

## 七、结果分析

（分析算法的正确性、时间复杂度、空间复杂度、实验中遇到的问题等）

## 八、总结

(通过本次实验得到了什么结论，实验体会）