**算法分析与设计实验报告**

**第 2 次实验**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 姓名 | 杨杰 | 学号 | 201908010705 | | 班级 | 计科1907 |
| 时间 | 4.22 | 地点 | 软件大楼 | | | |
| 实验名称 | 动态规划法解0-1背包问题 | | | | | |
| 实验目的 | 通过上机实验，要求掌握动态规划解0-1背包算法的问题描述、算法设计思想、程序设计。 | | | | | |
| 实验原理 | 通过设定一个全局数组，用来存储子问题的求解值，然后每次通过调用子问题的解来最终构成原问题的解。利用动态规划法解0-1背包问题，根据不同数据规模的测试用例，能准确输出测试用例的最优解，并计算出程序运行所需要的时间。  递推方程为    构造最优解方法为 | | | | | |
| 实验步骤 | 1、把问题分解成若干个子问题，如背包仅可以容纳1个物品且可以容纳的质量为一等。 2、依次求出各个子问题的最优解。 3、每个子问题的最优解又可以从它的子问题的最优解中得到。 4、通过各个子问题的最优解得到原问题的最优解。 | | | | | |
| 关键代码 | void beibao(vector<int> w, vector<int> v, int n, int c)*//计算最优值*  {      int j = 0;      for (j = 0; j < w[n]; j++)          m[n][j] = 0;      for (j = w[n]; j <= c; j++)          m[n][j] = v[n];      for (int i = n - 1; i >= 1; i--)      {          for (j = 0; j < w[i]; j++)              m[i][j] = m[i + 1][j];          for (j = w[i]; j <= c; j++)          {              if (m[i + 1][j] > m[i + 1][j - w[i]] + v[i])                  m[i][j] = m[i + 1][j];              else                  m[i][j] = m[i + 1][j - w[i]] + v[i];          }      }      m[0][c] = m[1][c];      if (c >= w[0])          if (m[1][c - w[0]] + v[0] > m[1][c])              m[0][c] = m[1][c - w[0]] + v[0];  }  void beibao\_back(vector<int> w, int x[], int n, int c)*//构造最优解*  {      for (int i = 0; i < n; i++)      {          if (m[i][c] == m[i + 1][c])              x[i] = 0;          else          {              x[i] = 1;              c -= w[i];          }      }      if (c >= w[n])          x[n] = 1;      else          x[n] = 0;  } | | | | | |
| 算法复杂度分析 | beibao函数中两层for循环嵌套，外层for循环执行n-1次，内层for循环执行c+1次，总时间复杂度为O(nc).  beibao\_back函数中for循环执行n次，时间复杂度为O(n).  main函数调用了beibao函数和beibao\_back函数，最终的时间复杂度为O(nc).  空间复杂度为O(nc).  注意，这里的c不是一个常数，而是一个参数，代表背包的容量。 | | | | | |
| 测试结果  （含运行时间） | 小规模数据      中规模数据    大规模数据 | | | | | |
| 实验心得 | 最初调用rand()函数生成随机数时，并没有指定随机数的种子，导致每次生成的随机数都是相同的，后来调用srand((int)time(NULL))函数，将随机数种子设置为系统时间，解决了这个问题。  clock()函数、GetTickCount()函数、timeGetTime()函数、Boost库中的timer、高精度时控函数QueryPerformanceFrequency()和QueryPerformanceCounter()，以上函数都可以记录程序运行时间，但由于数据量较小时，程序运行时间很短，前三个函数的精度不足以计时，这时应该选用高精度时控函数QueryPerformanceFrequency()和QueryPerformanceCounter()。  动态规划求解具有以下的性质：  1.最优子结构性质：最优解包含了其子问题的最优解，不是合并所有子问题的解，而是找最优的一条解线路，选择部分子最优解来达到最终的最优解。  2.重叠子问题性质：先计算子问题的解，再由子问题的解去构造问题的解（由于子问题存在重叠，把子问题解记录下来为下一步使用，这样就直接可以从备忘录中读取）。其中备忘录中先记录初始状态。  经过这次实验，我对于动态规划法解0-1背包问题的相关代码已基本熟悉，算法知识得到了复习与巩固。在写代码与调试的过程中，在解决问题过程中，丰富了个人编程的经历和经验，提高了个人解决问题的能力。 | | | | | |
| 实验得分 |  | 助教签名 | |  | | |

**附录：完整代码**

*//动态规划法实现01背包*

#include <iostream>

#include <fstream>

#include <vector>

#include <windows.h>

using namespace std;

int m[2000][50000];

void beibao(vector<int> w, vector<int> v, int n, int c)*//计算最优值*

{

    int j = 0;

    for (j = 0; j < w[n]; j++)

        m[n][j] = 0;

    for (j = w[n]; j <= c; j++)

        m[n][j] = v[n];

    for (int i = n - 1; i >= 1; i--)

    {

        for (j = 0; j < w[i]; j++)

            m[i][j] = m[i + 1][j];

        for (j = w[i]; j <= c; j++)

        {

            if (m[i + 1][j] > m[i + 1][j - w[i]] + v[i])

                m[i][j] = m[i + 1][j];

            else

                m[i][j] = m[i + 1][j - w[i]] + v[i];

        }

    }

    m[0][c] = m[1][c];

    if (c >= w[0])

        if (m[1][c - w[0]] + v[0] > m[1][c])

            m[0][c] = m[1][c - w[0]] + v[0];

}

void beibao\_back(vector<int> w, int x[], int n, int c)*//构造最优解*

{

    for (int i = 0; i < n; i++)

    {

        if (m[i][c] == m[i + 1][c])

            x[i] = 0;

        else

        {

            x[i] = 1;

            c -= w[i];

        }

    }

    if (c >= w[n])

        x[n] = 1;

    else

        x[n] = 0;

}

int main()

{

    int c = 3000;

    int a, b;

    vector<int> w, v;

    ifstream infile("01beibao\_in\_l.txt", ios::in);

    while (infile >> a >> b)

    {

        w.push\_back(a);

        v.push\_back(b);

    }

    int n = w.size() - 1;

    ofstream outfile("01beibao\_out\_l.txt", ios::out);

    double time = 0;

    LARGE\_INTEGER nFreq, nBeginTime, nEndTime;

    QueryPerformanceFrequency(&nFreq);

    QueryPerformanceCounter(&nBeginTime);*//开始计时*

    beibao(w, v, n, c);

    for (int i = n; i >= 0; i--)

    {

        for (int j = 0; j <= c; j++)

            outfile << m[i][j] << ' ';

        outfile << endl;

    }

    outfile << m[0][c] << endl;

    int x[n + 1];

    beibao\_back(w, x, n, c);

    for (int i = 0; i <= n; i++)

        outfile << x[i] << ' ';

    outfile << endl;

    QueryPerformanceCounter(&nEndTime);*//停止计时*

    time = (double)(nEndTime.QuadPart - nBeginTime.QuadPart) / (double)nFreq.QuadPart;*//计算程序执行时间单位为s*

    outfile << "程序耗时" << time \* 1000 << "ms" << endl;

    infile.close();

    outfile.close();

    return 0;

}

*//01背包数据生成器*

#include <iostream>

#include <fstream>

#include <time.h>

#define random(a, b) (rand() % (b - a + 1) + a)

using namespace std;

int main()

{

    int t;

    cin >> t;

    ofstream outfile("01beibao\_in.txt", ios::out);

    srand((int)time(NULL));

    for (int i = 1; i <= 2 \* t; i++)

    {

        outfile << random(1, 100) << " ";

        if (i % 20 == 0)

            outfile << endl;

    }

    outfile.close();

    return 0;

}