**算法分析与设计实验报告**

**第 2 次实验**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 姓名 | 邹林壮 | 学号 | 202208040412 | | 班级 | 计算机科学与技术（拔尖班）2201 |
| 时间 | 2024.10.19 | 地点 | 院楼432 | | | |
| 实验名称 | 用动态规划算法求解 0-1 背包问题 | | | | | |
| 实验目的 | 通过上机实验，要求掌握动态规划算法的问题描述、算法设计思想、程序设计。 | | | | | |
| 实验原理 | 利用动态规划法求解 0-1 背包问题，并计算出程序运行所需要的时间。 | | | | | |
| 实验步骤 | 1. 首先确定输入输出，输入为n件物品，n件物品的质量w，价值v，背包的容量c,输出为装入物品的最大价值,约束是容量c，必须取出全部。 2. 需要构建该问题的最优子结构，假设所给0-1背包问题的一个最优解f(1)=（x1,x2,...,xn)，下面是相应子问题的一个最优解f(2)（x2,x3,...,xn)： 约束条件： IMG_256 目标函数： IMG_257   反证法证明：如果f(2)不是子问题的最优解，那么就存在f‘(2）>f(2),那整个问题的最优解为f’(2)+w1x1>f(2)+w1x1=f(1),与我们的前提f(1)是最优解相矛盾。  3.定义一个二维数组 dp，其中 dp[i][j] 表示在前 i个物品中，背包容量为 j时可以获得的最大价值。则状态转移方程为  dp[i][j]=max(dp[i−1][j],dp[i−1][j−w[i]]+v[i]) w[i]<=j  dp[i][j]=dp[i−1][j] 0<=j<=w[i]  即如果第i个物品的重量不超过j，我们可以选择放入背包或者不放入。  如果选择放入，那么背包内的总价值为dp[i-1][j-w[i]]+v[i]；  如果选择不放入，那么背包内的总价值就是dp[i-1][j]。  我们取这两种情况中的最大值作为dp[i][j]的值。  如果第i个物品的重量超过j，无法再放入，背包内的总价值就是dp[i-1][j]。  当i=0或者j=0时，dp[i][j] = 0，表示没有物品或背包容量为 0 时，最大价值为 0。  4.最后定义两层for循环，对物品件数和容量c按照状态转移方程定义，得到最终答案。  5.通过对dp表的搜索，我们可以构建出最优解，即dp[i][j]与其上方的dp[i-1][j]比较，如果相等，说明第i件物品没有取，对应为0；如果不相等，说明第i件物品取了，对应为1，并且将容量j减去wi，最终构造出最优解  6.由于空间上的限制，可以对01背包进行优化，这主要依赖于状态转移的依赖性：在01背包问题的DP解法中，状态转移方程表明，计算 dp[i][j] 时只依赖于 dp[i-1][j] 和 dp[i-1][j-w[i]]。这意味着，当我们计算当前行 i 时，我们只需要知道上一行i-1的值，因此可以省略第一个维度的内容。 | | | | | |
| 关键代码 | void \_01bag(int c,int \*v,int \*w,int n,int m[N][N]){  for(int j=0;j<=c;j++)m[0][j]=0;  for(int i=1;i<=n;i++)  for(int j=0;j<=c;j++){  if(j>=w[i])m[i][j]=max(m[i-1][j],m[i-1][j-w[i]]+v[i]);  else m[i][j]=m[i-1][j];  }  }  void \_01bagchoice(int c,int \*w,int n,int m[N][N],int\* x){  for(int i=n;i>0;i--){  if(m[i][c]==m[i-1][c])x[i]=0;  else{  c-=w[i];  x[i]=1;  }  }  }  //空间优化  void \_01bag(int c, int \*v, int \*w, int n) {  for (int j = 0; j <= c; j++) m[j] = 0;  for (int i = 1; i <= n; i++) {  for (int j = c; j >= w[i]; j--) {  m[j] = max(m[j], m[j - w[i]] + v[i]);  }  }}  void \_01bagchoice(int c, int \*w, int n) {  int current\_c = c;  for (int i = n; i > 0; i--) {  if (m[current\_c] == m[current\_c - w[i]]) x[i] = 0;  else {  x[i] = 1;  current\_c -= w[i];  }  }} | | | | | |
| 测试结果 | 确定正确性：保证了正确性  IMG_256  2）时间复杂度：O(nc)，其中n是物品的数量，c是背包的容量。因为两层循环遍历填充一个大小为n×c的二维数组。  空间复杂度：O(nc)，因为我们使用了一个二维数组来存储中间结果。  3）构造了三个规模的数据集：10，100，10000  IMG_256  由于我们建立的是n\*n的二维数组，所以最大开辟空间不能超过10000的量级，限制了问题规模 | | | | | |
| 实验心得 | 1. 这个题受到二维数组空间开辟的限制，因此数据集的规模并不是特别大，其优化方式，是对空间进行降维，从而实现问题的优化的 2. 这个题让我更加了解了动态规划构建最优子结构，以及如何证明最优子结构，并且可以顺利写出状态转移方程，并根据方程求得最终问题解决的程序 | | | | | |
| 实验得分 |  | 助教签名 | |  | | |

**附录：完整代码**

**问题代码：**

**#include<iostream>**

**#include<cmath>**

**using namespace std;**

**#define N 10001**

**int c,n;//c:capacity n:number**

**int v[N],w[N];//value weight**

**int m[N][N];//maxvalue**

**int x[N];//choice**

**void \_01bag(int c,int \*v,int \*w,int n,int m[N][N]){**

**for(int j=0;j<=c;j++)m[0][j]=0;**

**for(int i=1;i<=n;i++)**

**for(int j=0;j<=c;j++){**

**if(j>=w[i])m[i][j]=max(m[i-1][j],m[i-1][j-w[i]]+v[i]);**

**else m[i][j]=m[i-1][j];**

**}**

**}**

**void \_01bagchoice(int c,int \*w,int n,int m[N][N],int\* x){**

**for(int i=n;i>0;i--){**

**if(m[i][c]==m[i-1][c])x[i]=0;**

**else{**

**c-=w[i];**

**x[i]=1;**

**}**

**}**

**}**

**//void \_01bag(int c, int \*v, int \*w, int n) {**

**// for (int j = 0; j <= c; j++) m[j] = 0;**

**// for (int i = 1; i <= n; i++) {**

**// for (int j = c; j >= w[i]; j--) {**

**// m[j] = max(m[j], m[j - w[i]] + v[i]);**

**// }**

**// }**

**//}**

**//**

**//void \_01bagchoice(int c, int \*w, int n) {**

**// int current\_c = c;**

**// for (int i = n; i > 0; i--) {**

**// if (m[current\_c] == m[current\_c - w[i]]) x[i] = 0;**

**// else {**

**// x[i] = 1;**

**// current\_c -= w[i];**

**// }**

**// }**

**//}**

**int main(){**

**cin>>n;**

**cin>>c;**

**for(int i=1;i<=n;i++)cin>>w[i]>>v[i];**

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

**for(int j=0;j<N;j++)m[i][j]=0;**

**\_01bag(c,v,w,n,m);**

**cout<<m[n][c];**

**\_01bagchoice(c,w,n,m,x);**

**for(int i=1;i<=n;i++)cout<<"第"<<i<<"件物品选用："<<x[i]<<endl;**

**}**

**测量时间代码：**

**#include <iostream>**

**#include <fstream>**

**#include <vector>**

**#include <cstdlib>**

**#include <ctime>**

**#include <chrono>**

**#include <algorithm>**

**using namespace std;**

**#define N 10001**

**int c, n; // c: capacity, n: number**

**int v[N], w[N]; // value weight**

**int m[N][N]; // max value**

**int x[N]; // choice**

**void \_01bag(int c, int \*v, int \*w, int n, int m[N][N]) {**

**for (int j = 0; j <= c; j++) m[0][j] = 0;**

**for (int i = 1; i <= n; i++)**

**for (int j = 0; j <= c; j++) {**

**if (j >= w[i]) m[i][j] = max(m[i - 1][j], m[i - 1][j - w[i]] + v[i]);**

**else m[i][j] = m[i - 1][j];**

**}**

**}**

**void \_01bagchoice(int c, int \*w, int n, int m[N][N], int\* x) {**

**for (int i = n; i > 0; i--) {**

**if (m[i][c] == m[i - 1][c]) x[i] = 0;**

**else {**

**c -= w[i];**

**x[i] = 1;**

**}**

**}**

**}**

**// 生成随机数据并写入文件**

**void generateRandomData(int itemCount, const string& filename) {**

**ofstream fout(filename);**

**fout << itemCount << " " << 1000 << endl; // 假设最大容量为1000**

**for (int i = 0; i < itemCount; i++) {**

**int weight = rand() % 100 + 1; // 随机生成重量 (1 到 100)**

**int value = rand() % 100 + 1; // 随机生成价值 (1 到 100)**

**fout << weight << " " << value << endl;**

**}**

**fout.close();**

**}**

**// 读取数据从文件**

**void readDataFromFile(const string& filename) {**

**ifstream fin(filename);**

**fin >> n >> c; // 读取物品数量和最大容量**

**for (int i = 1; i <= n; i++) {**

**fin >> w[i] >> v[i]; // 读取每个物品的重量和价值**

**}**

**fin.close();**

**}**

**int main() {**

**srand(time(0)); // 初始化随机数种子**

**vector<int> sizes = {10, 100, 10000}; // 三种不同规模的数据**

**for (int itemCount : sizes) {**

**string filename = "knapsack\_data\_" + to\_string(itemCount) + ".txt";**

**// 生成随机数据**

**generateRandomData(itemCount, filename);**

**// 从文件中读取数据**

**readDataFromFile(filename);**

**// 测量运行时间**

**auto start = chrono::high\_resolution\_clock::now();**

**\_01bag(c, v, w, n, m);**

**auto end = chrono::high\_resolution\_clock::now();**

**auto duration = chrono::duration\_cast<chrono::milliseconds>(end - start);**

**// 输出结果**

**cout << "对于 " << itemCount << " 件物品，最大价值为: " << m[n][c] << endl;**

**cout << "运行时间: " << duration.count() << " 毫秒。" << endl;**

**// 选择的物品**

**\_01bagchoice(c, w, n, m, x);**

**// for (int i = 1; i <= n; i++) {**

**// cout << "第 " << i << " 件物品选用： " << x[i] << endl;**

**// }**

**cout << "-------------------------------------------" << endl;**

**}**

**return 0;**

**}**