**暨南大学本科实验报告专用纸**

课程名称 算法分析与设计实验 成绩评定

实验项目名称 指导教师 李展 实验地点 N116

实验项目编号 4-0 实验项目类型 综合型

学生姓名 张瑞鹏 学号 2020101124

学院 信息科学技术学院 系 计算机 专业 计算机科学与技术

实验时间 2022 年 月 日 午～ 月 日 下 午

**（一）问题描述**

**若在0-1背包问题中，各物品依重量递增排列时其价值恰好依递减排列。对这个特殊的0-1背包问题，设计一个有效算法找出最优解，并说明算法的正确性。**

**（二）算法思路**（用文字简单说明）

**假设背包容量为total，用wi表示每个物品的质量，vi表示价值。**

**贪心选择：在题目前提之下，先选排在前面的物品，直到背包无法放入为止。**

**最优子结构：假设已经找到最优解S，证明S-{x1}为total-w1的最优解。归谬法：假设存在最优解S1优于S-{X1}，则有S1+{x1}优于S，矛盾，故假设不成立，证毕！**

**算法实现部分：**

**假设已经按照题目所给的要求排序完成，只需要逐个判断是否能放入，若可以放入输出，否则终止程序。**

**（三）算法实施步骤和流程**（伪代码/流程图等方式描述）

int valuemax(int n,int weightlimit, int\* value, int\* weight) {

int left = weightlimit;

int result = 0;

for (int i = 0; i < n; i++) {

if (weight[i] <= left) {

left -= weight[i];

cout << "物品 " << value[i]<<" " << weight[i] << " 放入背包" << endl;

result += value[i];

}

else break;

}

return result;

}

**（四）源代码**（通过了编译运行的正确程序）

#include<iostream>

using namespace std;

int valuemax(int n,int weightlimit, int\* value, int\* weight) {

int left = weightlimit;

int result = 0;

for (int i = 0; i < n; i++) {

if (weight[i] <= left) {

left -= weight[i];

cout << "物品 " << value[i]<<" " << weight[i] << " 放入背包" << endl;

result += value[i];

}

else break;

}

return result;

}

int main(void) {

int n = 6;

int weightlimit = 18;

int value[6] = { 8,6,5,4,3,1 };

int weight[6] = { 1,3,5,6,7,8 };

cout << "test example" << endl;

for (int i = 0; i < n; i++) {

cout << value[i] << " " << weight[i] << endl;

}

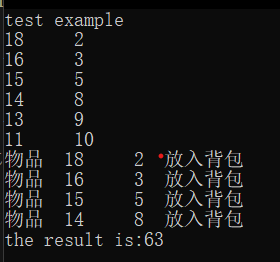
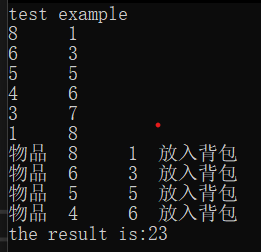
int result = valuemax(n,weightlimit, value, weight);

cout << "the result is:" << result << endl;

return 0;

}

**（五）测试结果**（至少有两个以上算例及程序运行结果，截图贴进实验报告）



**（六）实验总结**（至少三句话，可以写复杂度分析、遇到问题、可能的改进措施、心得体会等）

1.贪心算法比较代码十分简洁，需要严格的证明才能够说明算法的正确性，不能含糊，就像完整证明数学公式一样！

2．代码的时间复杂度为O(n)，完成最多一次遍历即可！

3.特殊的背包问题和普通的背包问题差别比较大，需要注意，遇到特殊的背包问题如果采用动态规划就小题大做。但是正常的背包问题使用贪心算法求解就会发生错误!

**暨南大学本科实验报告专用纸(附页)**