

**课 程 实 验 报 告**

**课程名称： 数据结构课程实验**

**专业班级： 计算机2206**

**学 号： 2225015653**

**姓 名： 王子涵**

**指导教师： 赵鹏**

**报告日期： 20xx年x月xx日**

**计算机科学与技术学院**

**目 录**

[实验1 背包问题求解 1](#_Toc531373256)

[1.1选题简介 1](#_Toc531373257)

[1.2问题分析 1](#_Toc531373258)

[1.3数据结构与算法设计 1](#_Toc531373259)

[1.3.1数据结构设计 1](#_Toc531373260)

[1.3.2算法设计 1](#_Toc531373261)

[1.4实验过程 1](#_Toc531373262)

1.4.1:开发环境.................................................................................................................................1

1.4.2:核心代码.................................................................................................................................1

[1.5实验结果与分析 2](#_Toc531373265)

[1.5.1程序测试 2](#_Toc531373266)

[1.5.2结果分析 2](#_Toc531373267)

[1.6算法分析与心得 2](#_Toc531373268)

[实验2 我是一个标题1格式的文字 3](#_Toc531373269)

[实验3 我是一个标题1格式的文字 4](#_Toc531373270)

[实验4 我是一个标题1格式的文字 5](#_Toc531373271)

[附录 实验源代码 6](#_Toc531373272)

[附录.1实验目录结构 6](#_Toc531373273)

[附录.2程序源代码 6](#_Toc531373274)

[参考文献 9](#_Toc531373275)

# 实验1 背包问题

## 1.1选题简介

假设有一个能装入总体积为T的背包和n件体积分别为w1,w2,…wn的物品，能否从n件物品中挑选若干件恰好装满背包，即使w1+w2+…+wm=T，要求找出所有满足上述条件的解。若每件物品同时具有体积和价值，背包有大小限制，求物品总体积最大值-------最优或近似最优解

## 1.2问题分析

可利用回溯法的设计思想来解决背包问题。首先，将物品排成一列，然后，顺序选取物品装入背包，若已选取第i件物品后未满，则继续选取第i+1件，若该件物品“太大”不能装入，则弃之，继续选取下一件，直至背包装满为止。

如果在剩余的物品中找不到合适的物品以填满背包，则说明“刚刚”装入的物品“不合适”，应将它取出“弃之一边”，继续再从“它之后”的物品中选取，如此重复，直到求得满足条件的解，或者无解。

由于回溯求解的规则是“后进先出”，自然要用到“栈”。

## 1.3数据结构和算法设计

### 1.3.1数据结构

使用栈，以顺序结构实现。PS:成员函数的实现较为冗长，故未给出

|  |
| --- |
| template <class Elem>  class Stack  {  public:    Stack(int size);    ~Stack() { delete[] data; }    bool IsEmpty() const;    bool IsFull();    Elem Pop();    void Push(Elem x);    inline Elem Top() { return data[top - 1]; }    int Size() { return top; }  private:    int maxsize;    int top;    Elem \*data;  }; |

### 

### 1.3.2算法设计

①利用栈的特性:后进先出，达到一种回溯的效果，来计算出刚好装入物品的体积为背包体积的各种方案

但是如果考虑每件商品的价值，在计算最优价值的时候,若使用栈把每一次的方案的价值记录下来并比较时间复杂度可能达到O(2^n)，因此在计算价值时采取了dp动态规划的思想,将时间复杂度改进到O(n^2)。

②动态规划，对于任意一个物品有两种状态，放入和不放入。

放入：剩余重量=当前包的重量-放入的该物品重量，拿着剩余重量，从剩余物品中选取来尽可能使总价值最大化后，再加上本物品的价值。也就是，放入的最大总价值=剩余重量的最大价值+本物品的最大价值。

不放入：剩余重量=当前包的重量，拿着剩余重量，从剩余物品中选取来尽可能使总价值最大化。不放入的最大总价值=剩余重量的最大值。

这边：拿着剩余重量，从剩余物品中选取来尽可能使总价值最大化。其实就是这里的子问题，动规dp的精髓就是，记录下子问题的最优解。

因为每个物体，都有装与不装两种选择，所以我们得到状态转移方程：

f [ j ] =max(f [ j ], f [ j-w [ i ]] +w [ i ]);

f [ j ] 为：当总容量为 j 时，不放第 i 件物品，所能装的最大体积。

f [ j-w [ i ]] +w [ i ] 为：当总容量为 j 时，放了第 i 件物品后，所能装的最大体积。（即 j 减去第 i 件物品体积 的容量能装的最大体积+第 i 件物品的体积。 w [ i ] 为第 i 件物品体积）



图1.1 XXXX流程图

## 1.4实验过程

### 1.4.1开发环境

本次实验中使用的环境配置如下：

（1）编译器及其版本：MinGW-w64

（2）编辑器：Visual Studio Code

### 1.4.2核心代码

|  |
| --- |
| void KnapSack(int N, int V, int \*BagV, int \*BagValue)  {    cout << "无价值,仅考虑体积:\n";    int V\_sum = V; // 实时记录体积和    int count = 0; // 记录解的个数    Stack<int> st(N);    int i = 1, k = 0;    while (i < N)    {      if (V\_sum >= BagV[k]) // 先放入一个到栈中,要判断是否符合体积      {        st.Push(k);        V\_sum -= BagV[k];        break;      }      else      {        i++, k++;      }; // i记录下一个目标，k记录第一个放入的    }    if (i == N)    {      cout << "No answer";      return;    }  while (1)    {      while (V\_sum > 0 && i < N)      {        if (V\_sum >= BagV[i])        {          V\_sum -= BagV[i];          st.Push(i);        }        if (V\_sum == 0)        {          count++;          break;        }        i++;      }  if (V\_sum == 0)  // 找到合适的一组解并输出        Output(st, count, N, BagV, BagValue);      if (!st.IsEmpty())  {  // 排除i==N-1的情况,以及判断是否筛选完成        if (st.Size() == 1)        {          i = st.Pop() + 1;          V\_sum += BagV[i - 1];          if (i == N)            break;        }        else if (i == N - 1)        {          V\_sum += BagV[st.Pop()];          i = st.Top() + 1;          V\_sum += BagV[st.Pop()];        }        else        {          i = st.Pop() + 1;          V\_sum += BagV[i - 1];        }      }    } |

## 1.5实验结果与分析

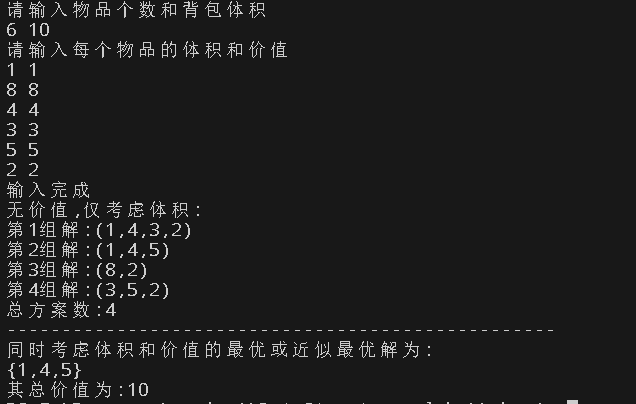
### 1.5.1程序测试

完成代码编写后，编译运行程序，并做如下的测试：

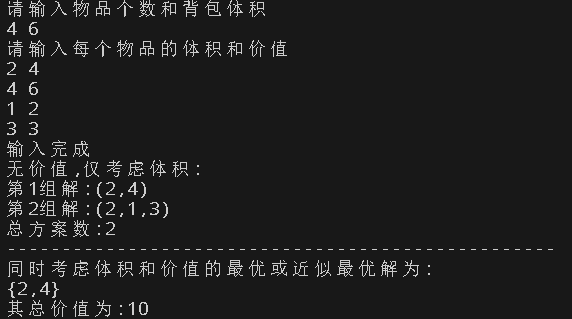
表1.1 XXXX测试内容

|  |  |  |  |
| --- | --- | --- | --- |
| **项目** | **测试** | **程序输出** | **测试结果** |
| 1 | (1,1)(8,8)(4,4)(3,3)(5,5)(2,2) | 有4种方案,最优价值为10,详细结果见下图 | 通过 |
| 2 | (2,4)(4,6)(1,2)(3,3) | 两种方案，最优价值为  10 | 通过 |
| 3 | (6,6) | 超过背包体积，无解 | 通过 |
| 4 | (3,8)(1,10) | 体积未装满，但有近似最优解 | 通过 |

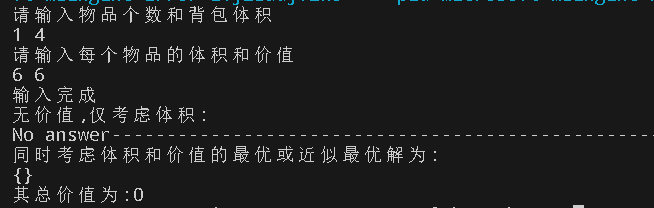
测试项目1：



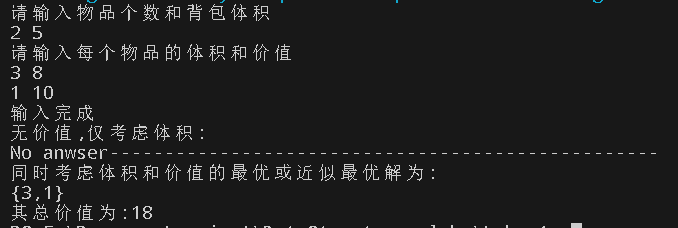
测试项目2：



测试项目3:



测试项目4:



### 1.5.2结果分析

我们前两组测试用例满足体积刚好为V有多种情况，并依次输出其方案和最大的价值及对应的物品体积

第三组数据没有一件物品能够装得下，因此程序输出No answer;

第四组数据无法满足体积刚好为V的情况，但价值上有近似最优解（总体积小于背包体积）

经过多组用例验证，程序符合实验要求，完全达到预期的实验目的。

## 1.6心得与体会

经过该次实验，我更加深刻的理解了数据结构：栈的原理及设计使用，并且学会计算算法的时间及空间复杂度来判断程序算法设计的优劣，以及为了进一步优化算法，采用了dp动态规划,将算法的时间复杂度从O（2^n）优化到了O(n^2)，大大减小了时间开销，优化进程。

# 实验2 我是一个标题1格式的文字

# 实验3 我是一个标题1格式的文字

# 实验4 我是一个标题1格式的文字

# 附录 实验源代码

## 附录.1实验目录结构

我是正文



图X.1 XXXX流程图

## 附录.2程序源代码

代码段X.1 XXX

**int** main(**void**) {

printf("Hello world!**\n**");

**return** 0;

}

注：文档的一些格式说明

标题1：宋体 小三 加粗 居中 单倍行距 段前12磅 段后12磅

标题2：宋体 四号 加粗 靠左 1.25倍行距 段前6磅 段后0磅

标题3：宋体 小四 加粗 靠左 单倍行距 段前3磅 段后8磅

正文：宋体 小四 1.25倍行距 段前0磅 段后0磅

西文字体统一为Times New Roman字体

内部小标题，中文括号，后面不加空格

阿拉伯数字标题，加点后加空格，例：1. 我是文字

图表格式说明：

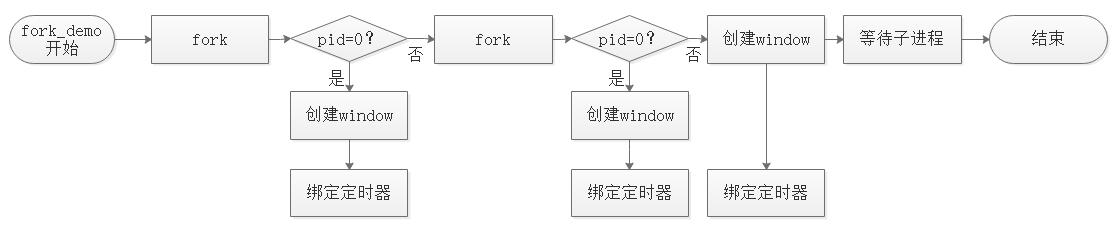


图1.1 fork\_demo流程图

图：环绕文字嵌入型，调整至适当大小，图前不空行，图后空一行，图前面的两个空格不能要。

图下方文字：宋体 小五 居中，标号从1.1开始按章节标号，标号与图标题之间空一格。

表：表居中，表前后各空一行，表头宋体 小五 加粗 居中，表正文宋体 小五。

表上方文字：表标题在表上方，宋体 小五 居中，标号方式和图相同。

表5.2 文件系统测试内容

|  |  |  |  |
| --- | --- | --- | --- |
| **项目** | **测试** | **命令** | **测试结果** |
| 1 | 测试创建目录 | mkdir test | 通过 |
| 2 | 测试创建文件 | touch data | 通过 |
| 3 | 测试列出内容 | ls -l | 通过 |
| 4 | 测试向文件写数据 | vim data | 通过 |
| 5 | 测试从文件读数据 | cat data | 通过 |
| 6 | 测试移动文件 | mv data testmv | 通过 |
| 7 | 测试复制文件 | cp testmv /test | 通过 |
| 8 | 测试只读权限 | chmod 444 testmv | 通过 |
| 9 | 测试只写权限 | chmod 244 testmv | 通过 |
| 10 | 测试添加用户 | useradd zxcpyp 888888 | 通过 |
| 11 | 测试删除用户 | userdel zxcpyp | 通过 |
| 12 | 测试格式化 | fmt | 通过 |

代码段格式说明：段前不空行，段后空一行，代码字体为Times New Roman。

方框效果和高亮效果制作：利用网站<http://pygments.org/>进行高亮化处理，复制到Word文档中，将字体改为Times New Roman 小四 黑色。

代码段前文字：宋体 小五 居中，标号和图一致。

代码段X.1 XXX

**int** main(**void**) {

printf("Hello world!**\n**");

**return** 0;

}

目录格式：

请按照本文档中的目录格式，我已经设置好，直接更新目录即可。

页码格式：

页码从正文开始计数，不计入目录，我已经设置好，无需修改。

**对于课程设计报告的说明：**

课程设计的报告，无非是在之前加上一个前言写出课题背景与意义，加上一个需求分析来详细说明，在之后加上一个总的感想与心得。测试模块则在倒数第二个统一进行测试。

在格式的控制和标题的控制上，完全适用本文档所提供的格式。无非是对于具体不同的课程设计任务，中间的部分会有一些不同罢了。

当然了，还是要注意封面的修改和页眉的修改。

# 参考文献

1 庞丽萍，阳富民著. 计算机操作系统（第2版）. 北京：人民邮电出版社，2014.

2 [美] Andrew S.Tanenbaum, Herbert Bos著，陈向群，马洪兵译. 现代操作系统（原书第4版）. 北京：机械工业出版社，2017.

3 [德] Michael Kerrisk著，孙剑，徐从年等译. Linux/UNIX 系统编程手册. 北京：人民邮电出版社，2014.

4 [美] Robert Love著，陈莉君，康华译. Linux内核设计与实现（原书第3版）.北京：机械工业出版社，2011.

5 [美] Randal E.Bryant，David O'Hallaron著，龚奕利，贺莲译. 深入理解计算机系统（原书第3版）. 北京：机械工业出版社，2016.

6 Arch Linux官方wiki文档. URL：<https://wiki.archlinux.org/>

参考文献格式说明：

A 图书：

[序号] 作者. 书名. 版本(第×版). 译者. 出版地：出版者, 出版年：起页-止页.

B 期刊：

[序号] 作者. 文章名称. 期刊名称，年号，卷号(期号)：起页-止页.

C 会议论文集：

[序号] 作者. 文章名称. In(见)：整本文集的编者姓名ed. (多编者用eds.). 文集名. 会址. 开会年. 出版地：出版者,出版年：起页-止页.

D 专利：

[序号] 专利申请者. 专利题名. 专利国别，专利文献种类，专利号，出版年：起页-止页.

E 学位论文：

[序号] 作者. 题名：[博士或硕士学位论文]. 保存地点：保存单位(如华中科技大学)，年份.

F 网页:

[序号] URL：网络地址

**本文档禁止一切个人、媒体、网站转载发布，引用请带上本仓库链接**

<https://github.com/zxc479773533/HUST-CS-Report-Template>