**西 南 石 油 大 学 实 验 报 告**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **课程** | **算法分析与设计** | **实验项目** | | **搜索算法设计与实现** | | | |
| **专业班级** | **计算机科学与技术2022级** | | **姓名** | | **王磊** | **学号** | **202231060435** |

## 分支定界搜索算法

### 1、实验内容

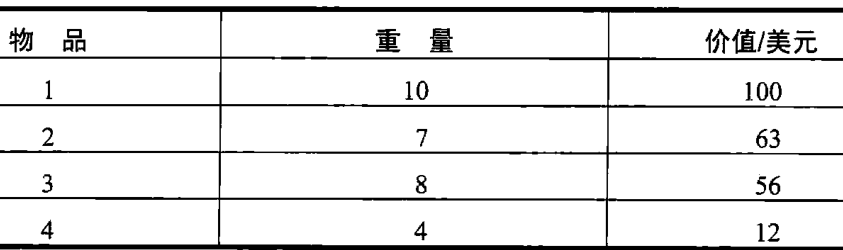
#### （1）实验步骤

1. 理解0-1背包问题及其约束。
2. 设计分支定界算法用于求解0-1背包问题。
3. 编写程序实现该算法。
4. 测试算法的性能。
5. 分析算法的时间复杂度和运行结果。

#### 算法原理

* 分支定界法是一种求解组合优化问题的算法。其基本思想是构造一个解空间树，然后系统地搜索这个解空间树以找到最优解。分支定界法利用了剪枝策略来减少需要搜索的节点数量，从而提高算法的效率。
* 具体到0-1背包问题，算法会从一个空的背包开始，将每个物品按一定顺序尝试加入背包中，计算当前背包的价值和重量，并判断是否继续搜索或进行剪枝。

1. 测试数据



#### 程序主要流程

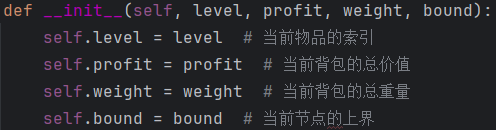
* 初始化问题参数（物品重量、价值和背包容量）。
* 定义一个用于表示节点的结构体或类，包含当前重量、当前价值和剩余容量等信息。
* 使用优先队列或其他合适的数据结构来管理待探索的节点。
* 实现分支定界算法的主循环，包括节点的扩展和剪枝操作。
* 输出最优解及其对应的价值。

#### 主要模块功能

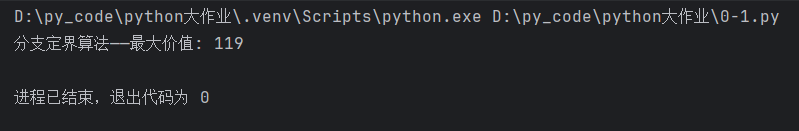
* 数据输入模块：读取并存储物品重量、价值和背包容量等数据。
* 分支定界算法模块：实现主要的搜索逻辑，包括节点扩展和剪枝。
* 结果输出模块：输出最优解及其价值。

#### 数据结构设计

* 使用优先队列存储待探索的节点。
* 节点结构体或类包含当前背包的总重量、总价值和剩余容量等信息。



#### 运行结果（截图）



#### （8）算法时间复杂度

* 分支定界算法的时间复杂度为 O(2n)

## 回溯搜索算法

### 1、实验内容

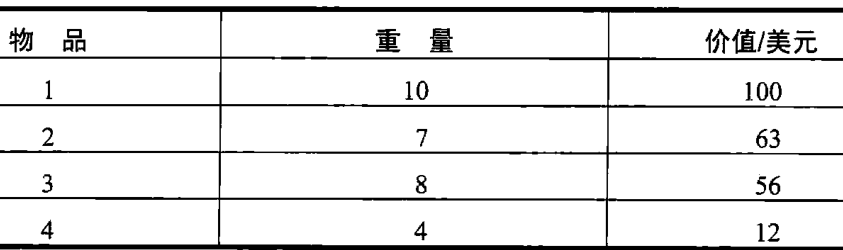
#### 实验步骤

1. 初始化当前最优解为0。
2. 从第一个物品开始，尝试选择或不选择该物品。
3. 递归地处理剩余的物品。
4. 每次选择或不选择物品后，更新当前重量和价值，并检查是否超过背包重量限制。
5. 如果所有物品都处理完毕，比较当前总价值与最优解，更新最优解。
6. 返回最终的最优解。

#### 算法原理

0-1背包问题是指给定一组物品，每个物品有一个重量和价值，在限定总重量的情况下，选择一些物品使得其总价值最大。回溯算法通过递归尝试每种可能的选择，找到最优解。

#### 测试数据



#### 程序主要流程

1. 初始化参数：

* 定义物品的重量和价值数组 weights 和 values。
* 定义背包的最大承重 W。

1. 定义回溯函数 backtrack：

* 参数包括当前物品索引 i，当前总重量 current\_weight，当前总价值 current\_value。
* 递归处理选择和不选择当前物品的两种情况。

1. 递归终止条件：

* 当所有物品都处理完毕时，比较当前总价值与最优解，更新最优解 best\_value。

1. 递归调用：

* 对于每一个物品，递归地选择或不选择，并更新当前重量和价值。

1. 求解并输出结果：

* 调用回溯函数，从第0个物品开始。
* 输出最终的最优解，即背包中物品的最大总价值。

#### 主要模块功能

1. 初始化模块：

* 设置物品的重量和价值，以及背包的最大承重。

1. 回溯函数模块：

* 递归地处理每一个物品的选择和不选择两种情况。
* 更新当前重量和价值。
* 终止条件判断，并更新最优解。

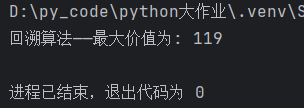
1. 求解与输出模块：

* 调用回溯函数，开始回溯搜索。
* 输出最终结果

#### 数据结构设计

1. 用数组表示存储物品的重量和价值
2. 用变量表示剩余最大承重和记录当前的最优解

#### 运行结果（截图）



#### 算法时间复杂度

使用回溯法解决01背包问题时，若可选物品为n个，则其解空间由长度为n的0-1向量组成此时时间复杂度为O(n2n)

## 1、小结

（自己对于实验的结论、认识、心得/体会等）

经过本实验，我理解了二者的联系与区别，并成功使用回溯法和分支限界法编写程序解决实际问题。以下是我的认识和体会。

从适应性来说回溯法和分支限界法都是通用求解方法，适用范围更广，可以用来求解NP问题。但是一般分支限界法会倾向于去求解最优解问题，当然也可以用来求解所有可行解，但是此时我们会更倾向于使用回溯法。

从使用难度上来说回溯法和分支限界法的编写难度较低，只需对解空间树进行搜索即可。但是约束函数和限界函数的编写的好坏对算法的效率影响非常大，如果不进行优化，这两个算法的时间和空间花费是不可接受的。

从算法效率上来说，回溯法和分支限界法的时间复杂度上界虽然很大，但是经过剪枝可以达到非常高效。但是到底效率能达到多高与剪枝的方法息息相关，一个好的剪枝方案可以带来优越的算法，而一个差的剪枝方案会带来一个没有实用性的算法。也就是说回溯法和分支限界法的效率的上限很高，但是下限也很低。总体上来看，回溯法要比分支界限法时间复杂度高，但是在合适的剪枝方案下，两个算法的效率都能非常高效。

本次实验让我收获良多，不仅深入学习了两种算法以及两者之间的区别，还通过练习之后熟练掌握了两种算法的使用。

## 附录：源代码

（完整代码）

