

专业课程实验报告

课程名称：算法分析与设计

开课学期： 2023 至 2024 学年 第 1 学期

专业： 软件工程中外 年级班级：1

学生姓名： 雷艺湧 学号：222021321062002

实验教师：苟建平

计算机与信息科学学院 软件学院

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验项目名称 | | 贪心算法实验---背包问题 | | | |
| 实验时间 | | 年 月 日 | 实验类型 | | □验证性 □设计性 □综合性 |
| 一、实验目的   1. 掌握贪心法的基本思想方法； 2. 了解适用于用贪心法求解的问题类型，并能设计相应贪心法算法； 3. 掌握贪心算法复杂性分析方法分析问题复杂性。   二、实验要求   1. 严格按照实验内容进行实验，培养良好的算法设计和编程的习惯； 2. 认真听讲，服从安排，独立思考并完成实验。 | | | | | |
| 三、实验内容与设计（主要内容，操作步骤、算法描述或程序代码）  贪心算法基本原理：  贪心算法是一种短视的算法，他在每步选取局部最优的选择，从而希望达到全局最优。  实验思路：  背包问题与01背包问题的最大不同是，01背包要求物品不能分割，要么选取有没不选取，而背包问题允许将物品进行分割，只要总大小不超过背包容量。这样的话，我们可以用每个物品的价值量除物品的重量来获得物品的性价比，再根据性价比进行从大到小的排序，每次都尽可能把性价比大的全部装完再去装下一种物品。这样，每次都会选择局部最优，从而达到全局最优（装的价值量尽可能大）  时间复杂度和空间复杂度：  时间复杂度：O(nlogn)  空间复杂度：O(n)  伪代码：   |  | | --- | | 按照 cost[i] 从大到小排序  for 每件物品i  将i放入的数量 = min(w[i], 剩余容量)  总价值 += 数量 \* v[i]  剩余容量 -= 数量 |   详细代码：   |  | | --- | | # 物品类 class Item:  def \_\_init\_\_(self, name, weight, value):  self.name = name  self.weight = weight  self.value = value  self.cost = value / weight   # 贪心算法求解0-1背包 def fractional\_knapsack(items, capacity):  items.sort(key=lambda x: x.cost, reverse=True)  total\_value = 0   for item in items:  if capacity <= 0:  return total\_value   amount = min(item.weight, capacity)  total\_value += amount \* item.cost  capacity -= amount   return total\_value   # 测试 item1 = Item('A', 2, 3) item2 = Item('B', 3, 4) item3 = Item('C', 4, 5) items = [item1, item2, item3] capacity = 5  max\_value = fractional\_knapsack(items, capacity) print(max\_value) | | | | | | |
| （续前表） | | | | | |
| 四、测试数据和执行结果 （在给定数据下，执行操作、算法和程序的结果，可使用数据、图表、截图等给出）  测试截图如下：  2023-12-01 01-37-49 的屏幕截图 | | | | | |
| 五、实验结果分析及总结（对实验的结果是否达到预期进行分析，总结实验的收获和存在的问题等）  实验结果分析：实验成功，每次都选取局部最优（选择性价比最高的装）成功装载最大价值量，实现了全局最优。  总结：本次实验让我深刻的认识到贪心算法是如何进行选择的，他不会像动态规划那样考虑到以后的情况，而是短视的考虑当下是否选取最优选项，这两种方法都可以达到立项的最优解。 | | | | | |
| 教  师  评  阅 | 实验内容和设计（A-E）： | | |  | |
| 操作过程、算法或代码（A-E）： | | |  | |
| 实验结果（A-E）： | | |  | |
| 实验分析和总结（A-E）： | | |  | |
| 实验成绩（A-E）：  反馈评语： | | | | |