算法第四次作业

李雨轩
计算机2205
2204112913

一. 特殊的0-1背包问题

1. 题目描述

若在0-1背包问题中，各个物品重量递增排序时，其价值恰好依递减排序。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。

2. 问题分析与算法设计

由于各个物品重量递增排序时，其价值恰好依递减排序，且各个物品的选择只有不选 (0) 或者选 (1) ，因此可以制定贪心策略：**每次选择重量最小、且价值最大的物品放入背包中，直到背包无法再容纳下一个物品为止**。

以下是该贪心算法的伪代码：

```
function greedy_knapsack(values[], weights[], W, N):
    // 按照重量递增、价值递减的顺序排序物品
    sort items by weight in increasing order and by value in decreasing order

    let total_value = 0
    let current_weight = 0

    for each item in items:
        if current_weight + item.weight <= W:
            current_weight += item.weight
            total_value += item.value

    return total_value
```

3. 算法的正确性

在这个特殊的0-1背包问题中，由于物品的重量递增排序时，其价值恰好依递减排序，因此每次选择重量最小、价值最大的物品放入背包中，可以使得背包的空间得到最充分的利用，从而达到最大化背包中物品的总价值。这是因为，如果我们选择放入重量较大的物品，即使其价值较高，也可能导致后续无法放入更多的物品，从而使得总价值不是最大化的。因此，贪心策略是正确的。

一个严格的证明。

我们假设我们已经将物品按照重量递增、价值递减的顺序排序。令 w_i 表示第 i 个物品的重量，v_i 表示第 i 个物品的价值。现在我们来证明贪心算法的正确性。

证明：

假设存在一个最优解 S^*，其包含物品 $i_1, i_2, \ldots, i_k$。我们将 S^* 中的物品按照重量递增的顺序排列为 $i_{p_1}, i_{p_2}, \ldots, i_{p_k}$，即 $w_{p_1} \leq w_{p_2} \leq \ldots \leq w_{p_k}$。

我们还假设贪心算法的解为 S，其包含物品 $j_1, j_2, \ldots, j_m$，按照重量递增的顺序排列为 $j_{q_1}, j_{q_2}, \ldots, j_{q_m}$，即 $w_{q_1} \leq w_{q_2} \leq \ldots \leq w_{q_m}$。

我们现在来证明对于所有 $r \leq \min(k, m)$，有 $v_{p_r} \geq v_{q_r}$。

基本思路是贪心算法每次选择的物品 j_r 要么与 i_r 相同，要么比 i_r 的价值更高。

我们用归纳法来证明这个性质。

- **归纳基础 $r = 1$:** 显然，贪心算法选择的第一个物品 j_1 要么等于 i_1，要么价值更高。因此，$v_{p_1} \geq v_{q_1}$。
- **归纳假设：** 假设对于 $r = k$，有 $v_{p_k} \geq v_{q_k}$。
- **归纳步骤：** 考虑 $r = k + 1$ 的情况。根据贪心算法的选择性质，我们有两种情况：
  - 如果 $w_{p_{k+1}} \leq w_{q_{k+1}}$，那么根据已知条件，$v_{p_{k+1}} \geq v_{q_{k+1}}$。
  - 如果 $w_{p_{k+1}} > w_{q_{k+1}}$，那么根据贪心算法的选择性质，$v_{p_{k+1}} \geq v_{p_k} \geq v_{q_k} \geq v_{q_{k+1}}$。

由归纳法的假设和归纳步骤可知，对于所有 $r \leq \min(k, m)$，有 $v_{p_r} \geq v_{q_r}$。

因此，贪心算法得到的解 S 价值不小于最优解 S^*。但由于贪心算法是一个可行解，所以贪心算法得到的解是最优解。因此，贪心算法能够得到最优解，证毕。

二. 边的最大权值达到最小的生成树

1. 题目描述

试设计一个构造图G生成树的算法，使得构造出的生成树的边的最大权值达到最小。

2. 问题分析与算法设计

题目要求设计一个贪心策略，以使得生成树的边的最大权值达到最小。注意到对于一个连通图，**对于一个连通图，其任何一颗最小生成树都是边的最大权值达到最小的生成树**，于是有下面的算法。

```
KRUSKAL-FUNCTION(G, w)
  F := 空集合
  for each 图 G 中的顶点 v
    do 将 v 加入森林 F
  所有的边(u, v) ∈ E依权重 w 递增排序
 for each 边(u, v) ∈ E
    do if u 和 v 不在同一棵子树
       then F := F ∪ {(u, v)}
          将 u 和 v 所在的子树合并
```

3. 算法的正确性

由于Kruskal正确性已知，我们只需要证明：**对于一个连通图，其任何一颗最小生成树都是边的最大权值达到最小的生成树**。

使用反证法。

假设存在一个连通图，其有一棵最小生成树，但是这棵最小生成树的边的最大权值不是最小的生成树。

设 T 是连通图 G 的一个最小生成树，而 T' 是一个具有更小的最大边权值的生成树。设 e 是 T' 中的一条边，其权值最大。由于 T 是最小生成树，因此 T 中不存在权值大于 e 的边。但是我们可以将 e 替换为 G 中的另一条边 f（不在 T' 中，否则 T' 不是树），并保持图的连通性，同时使得替换后的生成树的总权值不增加。这样我们得到了一个新的生成树，其最大边权值仍然比 T' 的最大边权值小，这与 T' 有最小的最大边权值的假设矛盾。

因此，得证 T 是 G 的一棵最小生成树，且其边的最大权值达到最小的生成树。