10.2 - Knapsack

背包问题（Knapsack Problem）

给定一个包含 n 个物品的集合 $\{1, 2, \ldots, n\}$，每个物品 i 具有非负的重量 w_i 和非负的价值 v_i
给定一个界限 W

我们的目标是选择一个子集 S 使得 $\sum_{i \in S} w_i \leq W$ 并且 $\sum_{i \in S} v_i$ 最大化

很显然，子集和问题是背包问题的一个特殊实例

分数背包问题（Fractional Knapsack Problem）

分数背包问题是一类允许将物品分割成任意份数并选择其中一部分的优化问题

给定一个包含 n 个物品的集合 $\{1, 2, \ldots, n\}$，每个物品 i 具有非负的重量 w_i 和非负的价值 v_i
给定一个界限 W

我们的目标是从每个物品 i 中选择一个分数 x_i 使得 $\sum_{i \in S} x_i \cdot v_i$ 最大化，同时满足 $\sum_{i \in S} x_i \cdot w_i \leq W$
这里，分数表示每个物品选择的比例，$x \in \{x_1, \ldots, x_n\}$，$0 \leq x_i \leq 1$

贪心算法

- 按照物品的“单位价值”（即价值与重量的比值 $\frac{v_i}{w_i}$）对从高到低物品进行排序
- 从单位价值最高的物品开始，尽可能多地选择该物品，直到达到背包的重量限制

这个算法最优地解决了分数背包问题

0/1 背包问题

在 0/1 背包问题中，每个物品要么被完全选择（1），要么完全不选（0），不能像分数背包问题那样选择部分物品

使用贪心算法选择单位价值最高的物品在这个问题中不再奏效
为了解决 0/1 背包问题，我们通常需要使用动态规划等其他算法，以确保找到全局最优解

$$f(k, w) = \begin{cases} f(k-1, w) & , w_k > w \\ \max\left(f(k-1, w), f(k-1, w-w_k) + v_k\right) & , w_k \leq w \end{cases}$$

其中，w 是背包容量限制，v 是物品的价值，$f(k, w)$：当背包容量为 w 时有 k 件物品可以装，其最大价值总和
我们可以使用像解决 subset sum 问题一样的表格来解决这个问题（懒得写了）