# 题目

有两只老鼠和 n 块不同类型的奶酪，每块奶酪都只能被其中一只老鼠吃掉。

下标为 i 处的奶酪被吃掉的得分为：

如果第一只老鼠吃掉，则得分为 reward1[i] 。

如果第二只老鼠吃掉，则得分为 reward2[i] 。

给你一个正整数数组 reward1 ，一个正整数数组 reward2 ，和一个非负整数 k 。

请你返回第一只老鼠恰好吃掉 k 块奶酪的情况下，最大 得分为多少。

示例 1：

输入：reward1 = [1,1,3,4], reward2 = [4,4,1,1], k = 2

输出：15

解释：这个例子中，第一只老鼠吃掉第 2 和 3 块奶酪（下标从 0 开始），第二只老鼠吃掉第 0 和 1 块奶酪。

总得分为 4 + 4 + 3 + 4 = 15 。

15 是最高得分。

示例 2：

输入：reward1 = [1,1], reward2 = [1,1], k = 2

输出：2

解释：这个例子中，第一只老鼠吃掉第 0 和 1 块奶酪（下标从 0 开始），第二只老鼠不吃任何奶酪。

总得分为 1 + 1 = 2 。

2 是最高得分。

提示：

1 <= n == reward1.length == reward2.length <= 105

1 <= reward1[i], reward2[i] <= 1000

0 <= k <= n

# 分析

要解决“第一只老鼠恰好吃掉k块奶酪时的最大得分”问题，核心思路是贪心策略：先计算所有奶酪由第二只老鼠吃的基础得分，再通过“差值排序”选择让第一只老鼠吃k块“收益最大”的奶酪，最终得到总得分的最大值。

解题思路

1、基础得分计算：

首先假设所有奶酪都由第二只老鼠吃掉，计算此时的总得分（基础得分）。这是因为后续只需考虑“将部分奶酪从第二只老鼠转移给第一只老鼠”的收益，无需从零开始计算。

2、收益差值分析：

对于每块奶酪i，若从“第二只老鼠吃”改为“第一只老鼠吃”，得分变化为：reward1[i] - reward2[i]（即收益差值）。

- 若差值为正：转移后总得分增加，是“划算”的；

- 若差值为负：转移后总得分减少，是“不划算”的。

3、贪心选择最优k块奶酪：

要让总得分最大，需选择收益差值最大的k块奶酪让第一只老鼠吃。具体步骤：

- 计算所有奶酪的收益差值，存入数组；

- 对差值数组按“从大到小”排序；

- 取前k个最大的差值，累加到基础得分中（相当于将这k块奶酪转移给第一只老鼠，获得最大额外收益）。

4、边界处理：

- 若k=0：第一只老鼠不吃任何奶酪，总得分即基础得分；

- 若k=n：所有奶酪都由第一只老鼠吃，总得分即基础得分加上所有差值（此时差值总和 = sum(reward1) - sum(reward2)，结果等价于sum(reward1)）。

代码：

class Solution {

public:

int miceAndCheese(vector<int>& reward1, vector<int>& reward2, int k) {

int n = reward1.size();

int total = 0; // 基础得分：所有奶酪由第二只老鼠吃的总得分

vector<int> diffs; // 存储每块奶酪的收益差值（reward1[i] - reward2[i]）

// 1. 计算基础得分和收益差值

for (int i = 0; i < n; ++i) {

total += reward2[i]; // 基础得分：第二只老鼠吃所有奶酪

diffs.push\_back(reward1[i] - reward2[i]); // 计算转移收益

}

// 2. 对收益差值按从大到小排序（优先选收益最大的k块）

sort(diffs.rbegin(), diffs.rend()); // 逆序排序（大→小）

// 3. 累加前k个最大的差值（将k块奶酪转移给第一只老鼠）

for (int i = 0; i < k; ++i) {

total += diffs[i];

}

return total;

}

};

代码解释

1、基础得分与差值计算：

遍历所有奶酪，累加reward2[i]得到基础得分，同时计算每块奶酪的“转移收益差值”（第一只吃 vs 第二只吃的得分差），存入diffs数组。

2、差值降序排序：

使用sort(diffs.rbegin(), diffs.rend())对差值进行逆序排序（从大到小），确保前k个元素是“转移收益最大”的奶酪。

3、累加最大收益：

将前k个最大差值累加到基础得分中，此时的总得分即为“第一只老鼠恰好吃k块奶酪”的最大得分。

复杂度分析

- 时间复杂度：O(n log n)，其中n是奶酪的数量。

遍历计算基础得分和差值：O(n)；

对差值数组排序：O(n log n)（排序是时间主导项）；

累加前k个差值：O(k)（k ≤ n，可忽略）。

- 空间复杂度：O(n)，额外使用diffs数组存储n个收益差值。

（若允许修改原数组，可优化为O(1)空间，但O(n)空间更直观，且n ≤ 1e5时开销可接受。）