# 题目

给你一个整数数组 nums 和一个整数 k ，按以下方法修改该数组：

选择某个下标 i 并将 nums[i] 替换为 -nums[i] 。

重复这个过程恰好 k 次。可以多次选择同一个下标 i 。

以这种方式修改数组后，返回数组 可能的最大和 。

示例 1：

输入：nums = [4,2,3], k = 1

输出：5

解释：选择下标 1 ，nums 变为 [4,-2,3] 。

示例 2：

输入：nums = [3,-1,0,2], k = 3

输出：6

解释：选择下标 (1, 2, 2) ，nums 变为 [3,1,0,2] 。

示例 3：

输入：nums = [2,-3,-1,5,-4], k = 2

输出：13

解释：选择下标 (1, 4) ，nums 变为 [2,3,-1,5,4] 。

提示：

1 <= nums.length <= 10^4

-100 <= nums[i] <= 100

1 <= k <= 10^4

# 分析

## 方法一：模拟

思路：

要使得数组的和最大，需要尽可能地选择绝对值较小的数进行取反操作。因此，首先将数组排序，然后从小到大遍历数组，对于负数，尽可能选择取反操作，直到操作次数用尽或者当前数字为非负数为止。

具体步骤如下：

1、将数组 nums 排序。

2、从小到大遍历数组 nums，对于每个元素：

如果该元素为负数且还有剩余操作次数 k，则将其取反，并将 k 减去 1。

如果该元素为非负数，或者 k 已经用完，则跳出循环。

3、计算数组 nums 的和，并根据剩余的操作次数 k 决定是否继续取反操作。如果 k 为奇数，则将数组中绝对值最小的元素取反。

代码：

class Solution {

public:

int largestSumAfterKNegations(vector<int>& nums, int k) {

std::sort(nums.begin(), nums.end());

int sum = 0, min\_abs = INT\_MAX;

for (int i = 0; i < nums.size(); ++i) {

if (nums[i] < 0 && k > 0) {

nums[i] = -nums[i];

--k;

}

sum += nums[i];

min\_abs = std::min(min\_abs, abs(nums[i]));

}

if (k % 2 == 1) {

sum -= 2 \* min\_abs;

}

return sum;

}

};