# 题目

给你一个整数数组 nums 和整数 k ，返回最大和 sum ，满足存在 i < j 使得 nums[i] + nums[j] = sum 且 sum < k 。如果没有满足此等式的 i,j 存在，则返回 -1 。

示例 1：

输入：nums = [34,23,1,24,75,33,54,8], k = 60

输出：58

解释：

34 和 24 相加得到 58，58 小于 60，满足题意。

示例 2：

输入：nums = [10,20,30], k = 15

输出：-1

解释：

我们无法找到和小于 15 的两个元素。

提示：

1 <= nums.length <= 100

1 <= nums[i] <= 1000

1 <= k <= 2000

# 分析

解题思路

首先把队列由大到小进行排序，然后通过左右指针的方式，遍历数组获取最大值。

代码

class Solution {

public:

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

int n = nums.size();

if(n<2) return -1;

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

int i=0, j=n-1, res = -1;

while(i<j)

{

int sum = nums[i]+nums[j];

if(sum<k)

{

res = max(res, sum);

i++;

}

else

{

j--;

}

}

return res;

}

};

class Solution {

public:

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

int n =nums.size();

int left =0;

int right = n-1;

int m = INT\_MIN;

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

while(left<right){

if(nums[left]+nums[right]>=k){

right--;

}

else{

m=max(m,(nums[left]+nums[right]));

left++;

}

}

return m==INT\_MIN?-1:m;

}

};

解题思路

1、排序 + 二分搜索找小于且最接近的数

2、排序 + 双指针

代码

class Solution {

public:

//找值小于target的最大索引

int binary\_search\_less(vector<int>& A, int target)

{

if (target <= A.front())

{

return -1;

}

if (target > A.back())

{

return A.size() - 1;

}

int left = 0;

int right = int(A.size()) - 1;

int index = -1;

while (left <= right)

{

int mid = left + (right - left) / 2;

if (target <= A.at(mid))

{

right = mid - 1;

}

else // target > A.at(mid)

{

index = max(index, mid);

left = mid + 1;

}

}

return index;

}

//1、

int twoSumLessThanK(vector<int>& A, int K) {

sort(A.begin(), A.end());

int res = -1;

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

{

int index = binary\_search\_less(A, K - A.at(i));

if (index != -1 && index != i) // 注意：index 不能等于 i 两个元素

{

int sum = A.at(i) + A.at(index);

if (sum > res && sum < K)

{

res = sum;

}

}

}

return res;

}

//2、

int twoSumLessThanK(vector<int>& A, int K) {

sort(A.begin(), A.end());

int res = -1;

int left = 0;

int right = int(A.size()) - 1;

while (left < right)

{

int sum = A.at(left) + A.at(right);

if (sum < K)

{

res = max(res, sum);

++left;

}

else

{

--right;

}

}

return res;

}

};