# 题目

给你一个数组 nums ，数组中有 2n 个元素，按 [x1,x2,...,xn,y1,y2,...,yn] 的格式排列。

请你将数组按 [x1,y1,x2,y2,...,xn,yn] 格式重新排列，返回重排后的数组。

示例 1：

输入：nums = [2,5,1,3,4,7], n = 3

输出：[2,3,5,4,1,7]

解释：由于 x1=2, x2=5, x3=1, y1=3, y2=4, y3=7 ，所以答案为 [2,3,5,4,1,7]

示例 2：

输入：nums = [1,2,3,4,4,3,2,1], n = 4

输出：[1,4,2,3,3,2,4,1]

示例 3：

输入：nums = [1,1,2,2], n = 2

输出：[1,2,1,2]

提示：

1 <= n <= 500

nums.length == 2n

1 <= nums[i] <= 10^3

# 分析

## 方法一：一次遍历

思路：

代码：

class Solution {

public:

std::vector<int> shuffle(std::vector<int>& nums, int n) {

std::vector<int> result;

result.reserve(2 \* n); // 预先分配足够的空间，提高效率

int x = 0, y = n; // x 指向 x 序列的起始位置，y 指向 y 序列的起始位置

while (x < n && y < 2 \* n) {

result.push\_back(nums[x]); // 将 x 序列的当前元素加入结果数组

result.push\_back(nums[y]); // 将 y 序列的当前元素加入结果数组

x++; // 移动 x 指针到下一个 x 元素

y++; // 移动 y 指针到下一个 y 元素

}

return result;

}

};

或：

class Solution {

public:

vector<int> shuffle(vector<int>& nums, int n) {

vector<int> ans(2 \* n);

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

ans[2 \* i] = nums[i];

ans[2 \* i + 1] = nums[i + n];

}

return ans;

}

};

复杂度分析

时间复杂度：O(n)，其中n是给定的参数。需要遍历长度为2n的数组nums一次将数组重新排列，每个元素重新排列的时间是O(1)。

空间复杂度：O(1)。注意返回值不计入空间复杂度。