# 题目

给定一个数组，包含从 1 到 N 所有的整数，但其中缺了两个数字。你能在 O(N) 时间内只用 O(1) 的空间找到它们吗？

以任意顺序返回这两个数字均可。

示例 1:

输入: [1]

输出: [2,3]

示例 2:

输入: [2,3]

输出: [1,4]

提示：

nums.length <= 30000

# 分析

## 方法一：数学

这个问题可以通过数学方法解决，而不需要额外的空间。我们可以先计算出缺失的两个数字的和，然后再计算它们的乘积。接着，我们遍历数组，分别累加所有数字的和以及它们的平方和，最终就可以得到缺失的两个数字。

具体步骤如下：

1、计算出缺失的两个数字的和：缺失的两个数字为 1 到 N 中未出现的两个数字，假设它们为 x 和 y，则有 x + y = (1 + N) \* N / 2 - sum，其中 sum 为数组中所有数字的和。

2、计算出缺失的两个数字的乘积：缺失的两个数字的乘积为 1 到 N 中未出现的两个数字的平方和减去数组中所有数字的平方和。假设缺失的两个数字为 x 和 y，则有 x \* y = (1^2 + 2^2 + ... + N^2) - sumOfSquares，其中 sumOfSquares 为数组中所有数字的平方和。

3、遍历数组，分别累加所有数字的和以及它们的平方和，得到 sum 和 sumOfSquares。

4、解方程组，得到缺失的两个数字 x 和 y。

class Solution {

public:

vector<int> missingTwo(vector<int>& nums) {

int N = nums.size() + 2; // 缺失的数字个数为 2，数组长度加上这两个数字的和为 N

long long sum = (long long)N \* (N + 1) / 2; // 1 到 N 的和

long long sumOfSquares = (long long)N \* (N + 1) \* (2 \* N + 1) / 6; // 1 到 N 的平方和

for (int num : nums) {

sum -= num;

sumOfSquares -= (long long)num \* num;

}

long long diff = sum \* sum - sumOfSquares; // (x + y)^2 - (x^2 + y^2) = 2xy

long long prod = diff / 2; // 缺失的两个数字的乘积为 diff / 2

// 解二元一次方程组 x + y = sum，xy = prod，得到 x 和 y

long long xPlusY = sum;

long long x = (xPlusY + sqrt(xPlusY \* xPlusY - 4 \* prod)) / 2;

long long y = xPlusY - x;

return {int(x), int(y)};

}

};

在这个实现中，我们首先计算出缺失的两个数字的和sum和乘积prod，然后通过解二元一次方程组的方式得到缺失的两个数字。需要注意的是，为了避免整型溢出，我们将sum和prod都定义为long long类型。