# 题目

有一个数组，其中的值符合等差数列的数值规律，也就是说：

在 0 <= i < arr.length - 1 的前提下，arr[i+1] - arr[i] 的值都相等。

我们会从该数组中删除一个 既不是第一个 也 不是最后一个的值，得到一个新的数组  arr。

给你这个缺值的数组 arr，请你帮忙找出被删除的那个数。

示例 1：

输入：arr = [5,7,11,13]

输出：9

解释：原来的数组是 [5,7,9,11,13]。

示例 2：

输入：arr = [15,13,12]

输出：14

解释：原来的数组是 [15,14,13,12]。

提示：

3 <= arr.length <= 1000

0 <= arr[i] <= 10^5

# 分析

## 方法一：公式

等差数列和=（首项+尾项）\* 项数 ÷ 2

项数为当前arr中元素个数+1

“数列和”与“当前数组的和” 的差值就是要求的元素

class Solution:

def missingNumber(self, arr: List[int]) -> int:

return (arr[0] + arr[-1]) \* (len(arr) + 1) // 2 - sum(arr)

解题思路：

等差数列前 N 项和公式：S = (a1 + an) × n ÷ 2

本题关键信息：缺了一个，这一个不是头也不是尾。

因此，只要把不缺它的和 S1，减去缺了它的和 S2，不就是它本身吗？

**代码：**

class Solution {

public int missingNumber(int[] arr) {

int arraySum = 0;

for (int i : arr) {

arraySum += i;

}

int length = arr.length;

int realSum = (arr[0] + arr[length - 1]) \* (length + 1) / 2;

// 等差数列求和：S = (首项+末项)\*项数/2

return realSum - arraySum;

}

}

## 方法二：二分

因为没有删除首尾元素，可以直接根据元素个数和首位元素差算出初始等差数列的差值，然后用二分搜索寻找这个差值出现的位置。

class Solution {

public int missingNumber(int[] arr) {

int N = arr.length;

int dif = (arr[N - 1] - arr[0]) / N;

int l = 0;

int r = N - 1;

while(l < r - 1) {

int mid = l + (r - l + 1) / 2;

if(arr[mid] - arr[l] == dif \* (mid - l))

l = mid;

else

r = mid;

}

return (arr[l] + arr[r]) / 2;

}

}