# 两数之和853769620

## A. 暴力求解算法——$O(N^2)$

### 1. 题目信息

* A. 数组下标的计数方式：0，1，2，......，nums.length-1
* B. 如果有多重解出现，只需要返回一个即可。【在循环中，如果遇到解，可以break，提前结束】
* C. 每个数组中的整数，最多被使用一次。【返回的下标，必须是不同的】
* D. 下标返回可以无序。

### 2. 算法求解思路

#### 2.1 随机算法：

如果一次求解的概率为$1/n$, 那么只需要重复10n次，就“可以”得到问题的答案。

原理： $(1-1/x)^x \approx 1/e > 1/3$

#### 2.2 顺序求解思路：

a) 设定两个指针i， j 。永远可以假设 i < j。

b) 初始化时，则需要让j = i+1.

### 3. 总结

1. 随机算法不稳定，但是一旦可以确定一个概率，是可以暴力求解的。
2. 如果求解的问题是输入的一个子集，那么可以尝试暴力枚举。

## B. 二分搜索求解--$O(n \log n)$

### 1. 算法思路

* 二分查找算法：折半（要么我们可以找到target，要么我们可以确定一半的数不是target）。
* 返回条件：对每一个新数组（1）如果mid+1 > nuns.length-1 或者 （2）0 > mid-1；
* 在计算mid下标的时候，使用（右确界-左确界）/2，不用+1；否则会有超出数组界限的风险。NOTE：如果是非递归实现，需要后面加上左确界的偏移量。

### 2. 总结

* 如果排序的时间复杂度不高于算法的预期，一般可以尝试使用排序。
* 出现双重循环的时候，可以考虑使用其他办法替代内部循环。因为内部循环的不确定变量更少。

## C. Hash表求解-- $O(n)$

### 1. 算法思路

A. 整体的优化思路，实际上是在优化最内层的循环：本质是让算法可以快速定位到target-nums[i]。（指针i是最外层的指针）。

B. Hash表刚好可以满足我们$O(1)$时间内，快速判断数值是否在key中。

C. 数组中存在相同的数值，hash表value的覆盖对结果无影响：

* 如果target是由两个相同的数值构成。那么只有一个数值会存在HashMap中。另一个数值，可以通过指针i(外部循环)找到。只需额外添加一次index判断，避免获取同一个index。
* 如果解不是两个相同的数之和。那么hashmap中一定存储了可以构成解的两个数。

## 2. 总结

A. Hash表是一个非常强大的工具。在解决数值快速查询，查找的时候，可以优先考虑使用。当输入中存在大量重复的数值时，需要根据具体情况来判断Hash表是否可以使用。

B. 如果算法存在多重循环，一般改进的突破口都会在最内层的循环上。因为外部循环固定住了参数，内层的循环一般会被简化，通常会有一些更为高效的求解方式。