Find the odd appearing element in O(Log n) time

> 在O(log n)时间复杂度内寻找数组中出现为奇数次数的数，数组中所有重复出现的数都是成对出现的，除了多出的数，比如{2,2,1,2,2,1,1}

可以采用XOR的方法，此处介绍二分法

首先需要明确的是：在重复的那个数出现之前，所有成对的数，都是第一次出现在偶数索引上，第二次出现在奇数索引上，当重复数出现后后，之后成对的数，第一次都出现在了奇数索引上，第二次出现在了偶数索引上，以{2,2,1,2,2,1,1}为例，索引为{0,1,2,3,4,5,6}，所以：

1. 如果mid为偶数,则比较arr[mid]与arr[mid+1]是否相等， 如果相等，重复的数出现在mid的右边，否则在mid的左边
2. 如果mid为奇数，则比较arr[mid]与arr[mid - 1]是否相等，如果相等，重复的数出现在mid的右边，否则出现在mid的左边

```
#include <bits/stdc++.h>

using namespace std;

void odd(vector<int> &arr, int low, int high) {
    if (low > high) return;
    if (low == high) {
        printf("%d\n", arr[low]);
        return;
    }
    int mid = low + (high - low) / 2;
    if (mid % 2 == 0) {
        if (arr[mid] == arr[mid + 1])
            odd(arr, mid + 2, high);
        else
            odd(arr, low, mid);
    } else {
        if (arr[mid] == arr[mid - 1])
            odd(arr, mid + 1, high);
        else
            odd(arr, low, mid - 1);
    }
}


int main() {
    int T;
    scanf("%d", &T);
    while (T--) {
        int N;
        scanf("%d", &N);
        vector<int> arr(N);
        for (int i = 0; i < N; ++i) scanf("%d", &arr[i]);
        odd(arr, 0, N - 1);
    }
}
```