K-th element of two sorted Arrays

> 给定两个有序数组，查询两个数组中第k个数

暴力法

双指针分别遍历A，B数组

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

using namespace std;

int kthEle(vector<int> &A, vector<int> &B, int &N, int &M, int &K) {
    int leftA = 0, leftB = 0, k = 1;
    while (leftA < A.size() && leftB < B.size() && k < K) {
        A[leftA] < B[leftB] ? leftA++ : leftB++;
        k++;
    }
    if (leftA < A.size() && leftB < B.size() && k == K) return min(A[leftA], B[leftB]);
    if (leftA < A.size() && k == K) return A[leftA];
    if (leftB < B.size() && k == K) return B[leftB];
    while (leftA < A.size() && k < K) {
        k++;
        leftA++;
    }
    while (leftB < B.size() && k < K) {
        k++;
        leftB++;
    }
    if (leftA < A.size()) return A[leftA];
    return B[leftB];

}

int main() {
    int T;
    scanf("%d", &T);
    while (T--) {
        int N, M, K;
        scanf("%d %d %d", &N, &M, &K);
        vector<int> A(N), B(M);
        for (int i = 0; i < N; ++i) scanf("%d", &A[i]);
        for (int i = 0; i < M; ++i) scanf("%d", &B[i]);
        printf("%d\n", kthEle(A, B, N, M, K));

    }
}
```

二分法

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

using namespace std;

int kthEle(vector<int> &A, vector<int> &B, int startA, int endA, int startB, int endB, int K) {
    if (startA > endA) return B[startB + K - 1];
    if (startB > endB) return A[startA + K - 1];
    // 从A和B的开始位置到两个数组中间位置的元素个数
    int midA = startA + (endA - startA) / 2, midB = startB + (endB - startB) / 2;
    int halfLen = midA - startA + midB - startB + 2;
    if (A[midA] < B[midB]) {
        // 前一种情况
        // 此时在合并的数组中A[startA...midA]元素一定在B[midB]的左侧，
        // 即此时第k大的元素一定比B[midB]这个元素小（严格来说不大于）
        // 故以后没有必要搜索 B[midB...endB]这些元素

        // 后一种情况
        // 此时在合并的数组中A[startA...midA]元素一定在B[bMid]的左侧，
        // 所以前K个元素中一定包含A[startA...minA]（可以使用反证法来证明这点）。
        // 但是无法判断A[midA+1...endA]与B[startB...endB]之间的关系，所以需要进行判断
        // 此时K就剩下除去A[startA...midA]这些元素，个数为k - (midA - startA + 1)
        return halfLen > K ? kthEle(A, B, startA, endA, startB, midB - 1, K) : kthEle(A, B, midA + 1, endA, startB,
                                                                                      endB, K - (midA - startA + 1));
    } else {
        return halfLen > K ? kthEle(A, B, startA, midA - 1, startB, endB, K) : kthEle(A, B, startA, endA, midB + 1,
```

```
                                                                                endB, K - (midB - startB + 1));
    }
}

int main() {
    int T;
    scanf("%d", &T);
    while (T--) {
        int N, M, K;
        scanf("%d %d %d", &N, &M, &K);
        vector<int> A(N), B(M);
        for (int i = 0; i < N; ++i) scanf("%d", &A[i]);
        for (int i = 0; i < M; ++i) scanf("%d", &B[i]);
        printf("%d\n", kthEle(A, B, 0, N - 1, 0, M - 1, K));

    }
}
```