Brute Force:

def median(a: int, b: int) -> float:

    n=len(a)

    m=len(b)

    num1=0

    num2=-1

    if (n+m)%2==0:

        num1=((n+m)//2)-1

        num2=(n+m)//2

    else:

        num1= (n+m)//2

    i=0

    j=0

    median=0

    count=0

    ele=0

    while(i<n and j<m):

        if a[i]<b[j]:

            ele=a[i]

            i+=1

        else:

            ele=b[j]

            j+=1

        count+=1

        if (count-1)==(num1):

            median=ele

            if num2==-1:

                break

        if num2!=-1 and count-1==num2:

            median+=ele

            median=median/2

            break

    return float(median)

OPTIMAL

def median(a: int, b: int) -> float:

    A,B=a,b

    if len(B)<len(A):

        A,B=B,A #As binary search only on smaller array

    half=(len(A)+len(B))//2

    tot=len(A)+len(B)

    l,r=0,len(A)-1

    while(l<=r):

        mid=(l+r)//2

        j=half-2-mid#as both i and j start with zero and we need index

        #remaining elemnt

        Aleft, BLeft, Aright, Bright = float('-inf'), float('-inf'), float('inf'), float('inf')

        #for out of bounds solution

        if mid>=0:

            Aleft=A[mid] WE CHECK 4 ELEMENTS

        if j>=0:

            BLeft=B[j]

        if mid+1<len(A):

            Aright=A[mid+1]

        if j+1<len(B):

            Bright=B[j+1]

        #Binary Seach part

        if Aleft<=Bright  and BLeft<=Aright:

            if tot%2==1:

                return float(min(Aright,Bright))

            else:

                return ((max(Aleft,BLeft)+min(Aright,Bright))/2.0)

        elif Aleft>Bright:#to many elemnts in A decreAse

            r=mid-1

        else:

            l=mid+1