(a)

Let, A is the given array

We split it into 2 part A1 & A2 of half size

Now, if x is the majority element of A1 & A2, then x must be the majority element of A.

So, We can use divide and conquer .

Let, n = size of A;

Major(int lo , int hi){

if(lo==hi)return A[lo];

int mid = (lo+hi)/2;

left = Major(lo,mid); //majority element of 1st half

right = Major(mid+1,hi); // majority element of 2nd half // T(n/2)

if(left==right)return left;

int left\_count=0 , right\_count=0;

for(int i=1 ; i<=n ; i++)left\_count += a[i]==left;

for(int i=1 ; i<=n ; i++)right\_count += a[i]==right; // total =~ O(n)

if(left\_count > mid - (hi-lo+1)%2)return left;

else if(right\_count > mid - (hi-lo+1)%2)return right; // here , if A = {1,2,1} I considered 1

else return No Majority Element ;

}

So,Time complexity = T(n) = 2T(n/2) + O(n) = O(nlogn) [by master theorem]

(b)

As procedure says, there will be n/2 pairs & from each pair we can take at most 1 so, if we take from every pair((it will be possible if there is a majority element)) total element will be n/2 ….and it’s the highest………..(1)

On\_major()

{

temp

cnt=0;

for (int i = 1; i <=n; i++)

{

if (!cnt)

{

temp = A[i];

cnt = 1;

}

else

{

if (temp == A[i])cnt++;

else cnt--;

}

}

if (cnt == 0)return -1;

cnt = 0;

for (int i = 1; i <= n; i++)

{

if (temp == A[i])cnt++;

}

return cnt > n/2 - (n%2) ? temp : “Nope”;

}

NB: I have attached a cpp file name Task\_2 with int array to test the algo