**Experiment No: 3 Date:- 02-04-2021**

**AIM:** Implementation of MaxMin Algorithm using Divide and Conquer and obtaining its step count

**THEORY:**

MaxMin Algorithm is, as the name suggests, an algorithm to find the maxima and minima of an array of items. If it is implemented using the naive way, i.e. without the divide and conquer approach, number of comparisons adds up to 2(n-1), where n is size of array. This is quite a large number and will slow down the program execution for considerable large value of n. In order to reduce the number of comparisons, the Divide and Conquer strategy is used, which is comparatively optimal. It first checks if the size of the array is 1 or 2. If it is 1, then that number itself the maxima as well as the minima. If it is 2, then the two elements are compared and the corresponding maxima and minima elements are computed. However, when the size of the array is greater than 2, the algorithm divides the input array into two halves and keeps calling itself in these two halves recursively, while it compares the maxima and minima in each call.

After each successive recursive call, the maximum and minimum are finally computed.

The MaxMin algorithm runs in O(n) time in all the cases.

**ALGORITHM:**

Algorithm MAXMIN(a,n,low,high,max,min)

{

// a is array of size n

// n no of elemnts

// let max be a global variable

// let min be a global variable

if(low==high) {

if(min>a[low]) min:=a[low];

if(max>a[low]) max:=a[low];

}

else if(low+1==high)

{

if(a[high]>a[low])

{

min:=a[low];

max:=a[high];

}

else{

min:=a[high];

max:=a[low];

}

}

else{

mid=(low+high)/2;

MAXMIN(a,n,low,mid,max,min);

MAXMIN(a,n,mid+1,high,max1,min1);

if(max1>max) max:=max1

if(min1<min) min:=min1

}

}

In this approach, the array is divided into two halves. Then using recursive approach maximum and minimum numbers in each halves are found. Later, return the maximum of two maxima of each half and the minimum of two minima of each half.

**TIME Complexity, Recurrence Relation Formulation and Solving:**

**:**

If T(n) represents this no., then the resulting recurrence relations is

T (n)=T([n/2]+T[n/2]+2 n>2

1 n=2

1 n=1

When ‘n’ is a power of 2, n=2k for some positive integer ‘k’, then

T (n) = 2T(n/2) +2

= 2(2T(n/4)+2)+2

= 4T(n/4)+4+2

= 2k-1 T (2) + Σ 1 ≤ I ≤ k-1 ≤ 2i

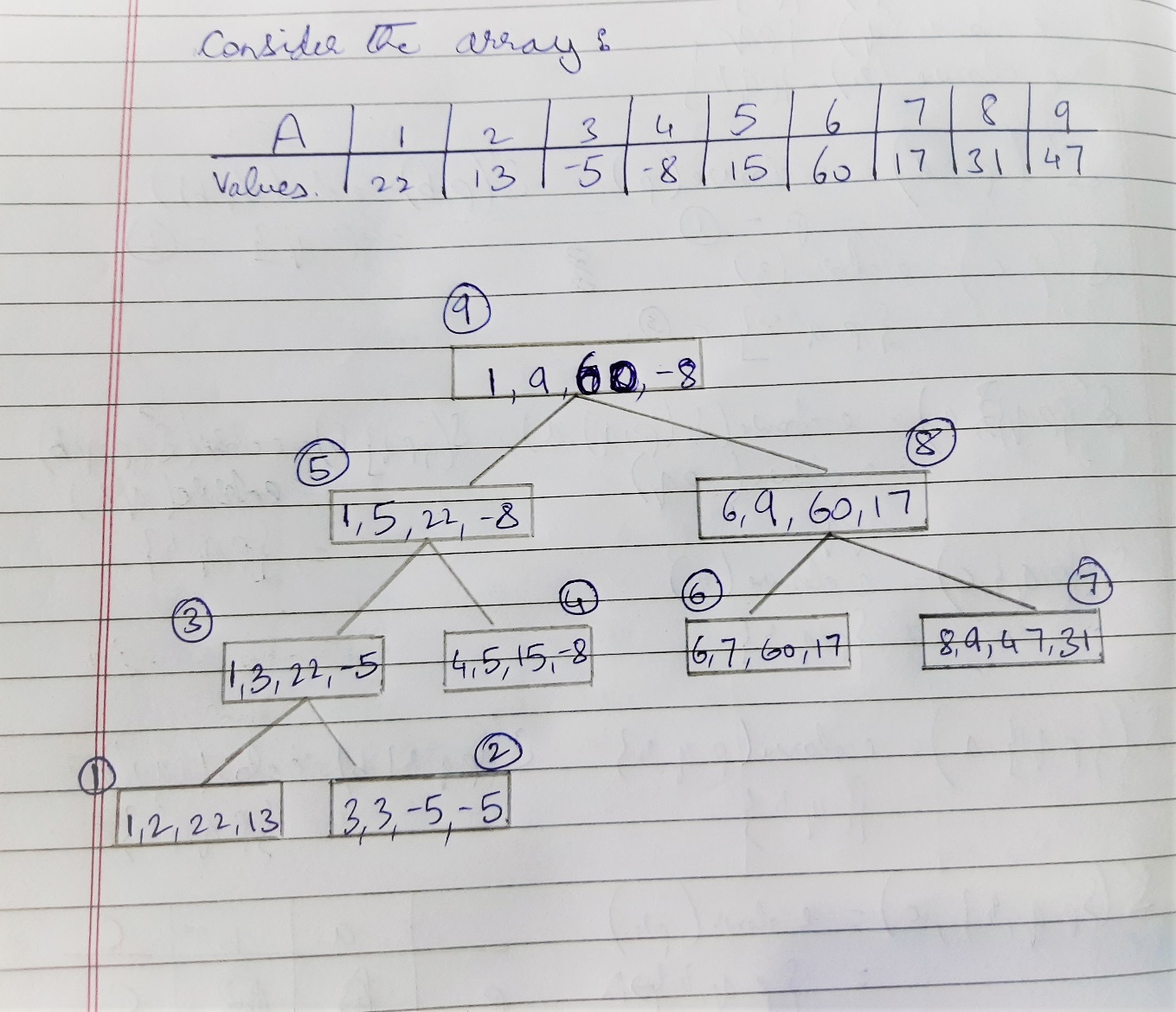
= 2k-1+ 2k - 2

T(n) = (3n/2) – 2

Note that (3n/2) - 2 is the best-average and worst-case no. of comparisons when ‘n’ is a power of 2.

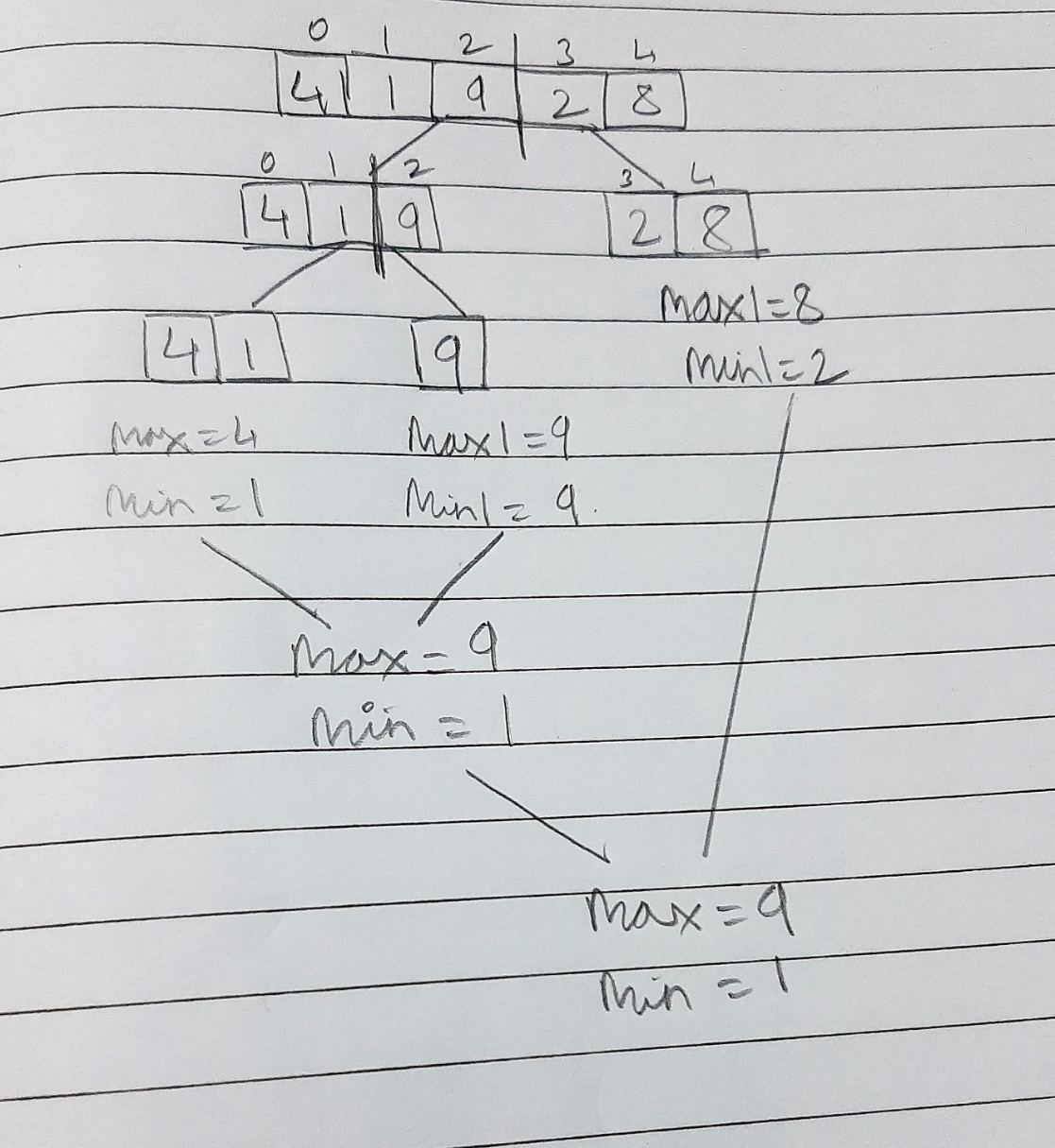
**Recursive Tree call Diagram:**

Shown below, is the recursive tree diagram:



**Problem Tracing:**

Consider array a={4,1,9,2,8 }. Here size of array n=5.



PROGRAM IMPLEMENTATION:

#include<iostream>

using std::cout;

using std::cin;

using std::endl;

int \*a,count,mid;

void maxmin(int i,int j, int &max, int &min)

{

count++; //if

if(i==j)

{

max=min=a[i];

count+=2; //assign

}

else if(i==j-1)

{

count++; //if

if(a[i]<a[j])

{

max=a[j];

min=a[i];

count+=2;

}

else

{

max=a[i];

min=a[j];

count+=2;

}

}

else

{

count++;

mid = (i+j)/2;

count++; //assign

maxmin(i,mid,max,min);

int max1,min1;

maxmin(mid+1,j,max1,min1);

count++; //if

if(max<max1)

{

count++;

max=max1;

}

count++; //if

if(min>min1)

{

count++;

min=min1;

}

}

count++;

return;

}

int main()

{

int n,max,min;

cout<<"Enter number of elements\n";

count++;

cin>>n;

count++;

a = new int[n];

count++;

cout<<"Enter "<<n<<" element(s) \n";

count++;

for(int i=0;i<n;i++)

{

count++; //for

cin>>a[i];

count++;

}

maxmin(0,n-1,max,min);

cout<<"The largest element:"<<max<<endl;

cout<<"The smallest element:"<<min<<endl;

cout<<"Count="<<count<<endl;

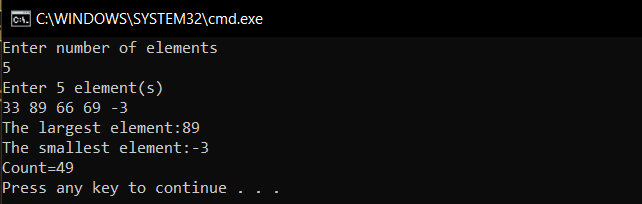
return 0;

}

OUTPUTS:

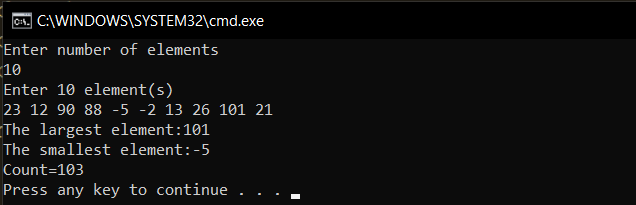
1. When n=5

**Count=49**

****

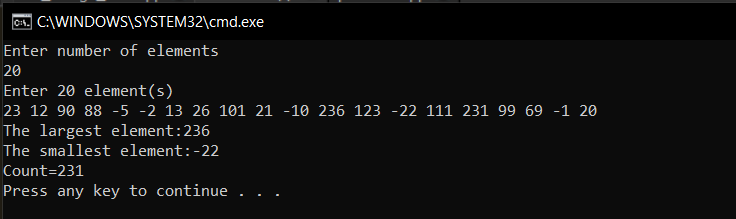
1. When n=10

**Count=103**

****

1. When n=20

**Count=231**

****

**Conclusion**: The program was successfully run and executed with main emphasis on the complexities of MaxMin Algorithm. The following was also observed.

1. The time complexity of MaxMin algorithm is O(n).
2. (3n/2) - 2 is the best-average and worst-case no. of comparisons when ‘n’ is a power of 2.