**U. V. Patel College of Engineering**

**B. Tech CSE (CBA/MA/BDA) Sem-V**

**Subject: Algorithm Analysis and Design (2CSE502)**

**Practical 4**

Implement the following algorithms and count the no. of comparisons as well as no. of steps executed by function on various inputs for best case and worst case. Also write complexity in each case and draw a comparative chart.

1. Straight MaxMin

Code:

#include<iostream>

using namespace std;

int main()

{

int n,arr[50],brr[50],i,j;

int count=0;

int min,max;

cout<<"Array Size : ";

cin>>n;

cout<<"Array Elements : ";

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

{

cin>>arr[i];

}

min=max=arr[0];

for(j=1;j<n;j++)

{

count++;

if(max<arr[j])

{

max=arr[j];

}

else

{

count++;

if(min>arr[j])

{

min=arr[j];

}

}

}

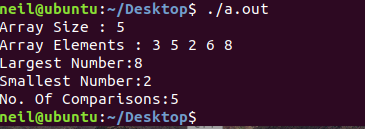
cout<<"Largest Number:"<<max<<"\n";

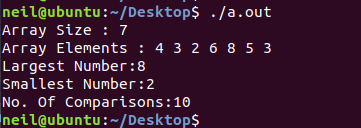
cout<<"Smallest Number:"<<min<<"\n";

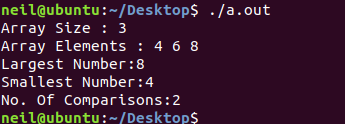
cout<<"No. Of Comparisons:"<<count<<"\n";

}

Output:







Divide and Conquer MinMax:

#include<stdio.h>

int a[50],max,min;

int counter;

void find(int i,int n){

int mid,max1,min1;

if(i==n)

max=min=a[i];

else if(i==n-1)

if(a[i]>=a[n-1]){

max=a[n];

min=a[i];

}

else{

max=a[i];

min=a[n];

}

else{

mid=(i+n)/2;

find(i,mid);

max1=max;

min1=min;

find(mid+1,n);

if(max<max1)

max=max1;

if(min>min1)

min=min1;

}

}

int main(){

int i,n;

printf("Enter size of array : ");

scanf("%d",&n);

printf("Enter elements in array-->\n");

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

scanf("%d",&a[i]);

max=min=a[0];

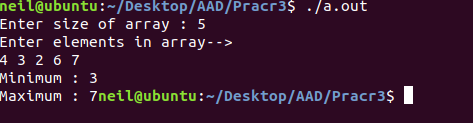
find(1,n-1);

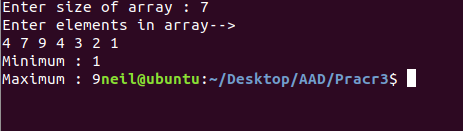
printf("Minimum : %d",min);

printf("\nMaximum : %d",max);

return 0;

}

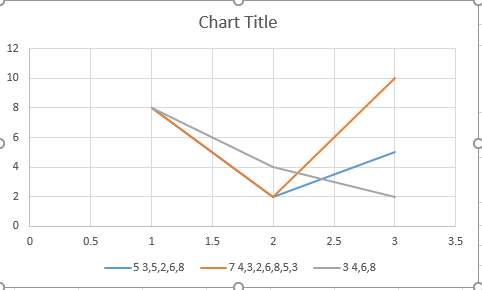




Analysis :

Straight maxmin :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Array Size | Element | Maximum | Minimum | No.ofComparison |
| 5 | 3,5,2,6,8 | 8 | 2 | 5 |
| 7 | 4,3,2,6,8,5,3 | 8 | 2 | 10 |
| 3 | 4,6,8 | 8 | 4 | 2 |



Straight Method: The number of comparison in method is **2n – 2.**

Divide and Conquer : Compared to Straight method, in divide and conquer approach, the number of comparisons is less. However, using the asymptotic notation both of the approaches are represented by **O(n)**.