MIN MAX

Aim- Write a C program to implement Minmax Algorithm on array of integers

Problem Statement – Given a array of integers implement Minmax algorithm to find the minimum and maximum element in the array

INPUT - Number of elements in the array – 11

Array Elements – 24,76,-4,58,23,86,-14,25,87,23,43

OUTPUT – Print i, j , minimum element ,maximum element during each iteration and final return maximum and minimum element in the array

ALGORITHM –

i] Algorithm Minmax (I,j,max,min)

// A global array a[i:j] is present . I and j are integer indexes in the array such that

//min<=i<=j<=max and we have to find max and min element within the array a[i:j]

```
{  if (i=j ) then {
min:= max:=a[i] ; }
else if (j=i+1 ) then {
if (a[i]>a[j] ) then {
max:=a[i]; min:= a[j] ; }
else then {  min:= a[i] ; max:= a[j] ;} }
else  then {  mid:= floor((i+j)/2) ;
minmax (i,mid,max,min) ;
minmax (mid+1,j,max1,min1) ;
if (max1>max ){  max:= max1;}
if (min1<min) { min :=min1 ; }
} }
```

Time and Space Complexity :

I] Algorithm Minmax

Time Complexity:

i) **Best Case:**

- **O(n)**
- The array is divided into halves, and each element is compared exactly once as the algorithm processes all elements.

ii) **Worst Case:**

- **O(n)**
- Regardless of the arrangement of the elements, the algorithm always processes all elements through recursive division and comparisons.

iii) **Average Case:**

- **O(n)**
- On average, the algorithm divides the array into smaller segments and performs the same number of comparisons as in the best and worst cases.

Space Complexity:

i) **Best Case:**

- **O(log n)**
- This accounts for the recursive stack depth during the division of the array.

ii) **Worst Case:**

- **O(log n)**
- The recursion depth is logarithmic in the size of the array, which remains the same in all cases.

iii) **Average Case:**

- **O(log n)**
- On average, the recursive stack space grows logarithmically with the size of the array.

RECURSION EQUATION :

I] Algorithm Minmax

$$T(n) = 2T\left(\frac{n}{2}\right) + O(1)$$

PROGRAM –

```
#include <stdio.h>

#include <time.h>

#define MAX 20

int a[MAX];


void Min_max(int low, int high, int *min, int *max) {


   int mid, min1, max1;

   if (low == high) {

      *min = *max = a[low];

      printf("i = %d, j = %d, min = %d, max = %d.\n", low,
high, *min, *max);

   } else if (low == high - 1) {

      if (a[low] < a[high]) {

         *min = a[low];

         *max = a[high];

         printf("i = %d, j = %d, min = %d, max = %d.\n",
low, high, *min, *max);

      } else {

         *min = a[high];

         *max = a[low];

         printf("i = %d, j = %d, min = %d, max = %d.\n",
low, high, *min, *max);

      }

   } else {

      mid = (low + high) / 2;

      Min_max(low, mid, min, max);

      Min_max(mid + 1, high, &min1, &max1);

      if (min1 < *min) {

         *min = min1;

      }

      if (max1 > *max) {

         *max = max1;

      }

      printf("i = %d, j = %d, min = %d, max = %d.\n", low,
high, *min, *max);

   }


}

void displayArray(int n) {

   if (n == 0) {

      printf("Array is empty.\n");

   } else {

      printf("Array elements: ");

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

         printf("|%d ", a[i]);

      }

      printf("|\n");

   }

}


int main() {

   printf("**************************************
**********");

   printf("\n Roll number: 23B-CO-010\n");

   printf(" PR Number - 202311390\n");

   printf("**************************************
**********\n\n\n");


   int choice, i, n = 0, min, max;

     clock_t start, end;

   double cpu_time_used;



   while (1) {

      printf("\nMenu:\n");

      printf("1. Enter array elements\n");

      printf("2. Find min and max\n");

      printf("3. Display array\n");
```

```
    printf("4. Exit\n");
    printf("Enter your choice: ");
    scanf("%d", &choice);

    switch (choice) {
      case 1:
        printf("Enter the number of elements in the array: ");
        scanf("%d", &n);

        if (n > MAX) {
          printf("Number of elements exceeds the maximum allowed (%d). Please try again.\n", MAX);
          n = 0;
        } else {
          printf("Enter the elements of the array: ");
          for (i = 0; i < n; i++) {
            scanf("%d", &a[i]);
          }
        }
        break;

      case 2:  start = clock();
        if (n == 0) {
          printf("Array is empty. Please enter array elements first.\n");
        } else {
          Min_max(0, n - 1, &min, &max);
          printf("Minimum element: %d\n", min);
          printf("Maximum element: %d\n", max);
           end = clock();
         cpu_time_used = ((double) (end - start)) / CLOCKS_PER_SEC;
          printf("Time taken by Min_max: %f seconds\n", cpu_time_used);
        }
        break;

      case 3:
        displayArray(n);
        break;

      case 4:
        break ;

      default:
        printf("Invalid choice. Please try again.\n");
    }
  }
  return 0;
}
```

INPUT –

```
**************************************************
 Roll number: 23B-CO-010
 PR Number - 202311390
**************************************************



Menu:
1. Enter array elements
2. Find min and max
3. Display array
4. Exit
Enter your choice: 1
Enter the number of elements in the array: 11
Enter the elements of the array: 24
76
-4
58
23
86
-14
25
87
23
43

Menu:
1. Enter array elements
2. Find min and max
3. Display array
4. Exit
Enter your choice: 3
Array elements: |24 |76 |-4 |58 |23 |86 |-14 |25 |87 |23 |43 |

Menu:
1. Enter array elements
2. Find min and max
3. Display array
4. Exit
Enter your choice: 2
```

OUTPUT –

```
i = 0, j = 1, min = 24, max = 76.
i = 2, j = 2, min = -4, max = -4.
i = 0, j = 2, min = -4, max = 76.
i = 3, j = 4, min = 23, max = 58.
i = 5, j = 5, min = 86, max = 86.
i = 3, j = 5, min = 23, max = 86.
i = 0, j = 5, min = -4, max = 86.
i = 6, j = 7, min = -14, max = 25.
i = 8, j = 8, min = 87, max = 87.
i = 6, j = 8, min = -14, max = 87.
i = 9, j = 10, min = 23, max = 43.
i = 6, j = 10, min = -14, max = 87.
i = 6, j = 8, min = -14, max = 87.
i = 6, j = 8, min = -14, max = 87.
i = 9, j = 10, min = 23, max = 43.
i = 6, j = 10, min = -14, max = 87.
i = 0, j = 10, min = -14, max = 87.
Minimum element: -14
Maximum element: 87
```

TIME TAKEN -

```
Time taken by Min_max: 0.002000 seconds
```

CONCLUSION – The maximum element and minimum element in the array was successfully calculated using Minmax Algorithm without any errors