Maximum and minimum of an array using minimum number of comparisons

Write a C function to return minimum and maximum in an array. You program should make minimum number of comparisons.

First of all, how do we return multiple values from a C function? We can do it either using structures or pointers.

We have created a structure named pair (which contains min and max) to return multiple values.

```
struct pair
{
  int min;
  int max;
};
```

And the function declaration becomes: struct pair getMinMax(int arr[], int n) where arr[] is the array of size n whose minimum and maximum are needed.

METHOD 1 (Simple Linear Search)
Initialize values of min and max as minimum and maximum of the first two elements respectively. Starting from 3rd, compare each element with max and min, and change max and min accordingly (i.e., if the element is smaller than min then change min, else if the element is greater than max then change max, else ignore the element)

```
/* structure is used to return two values from minMax() */
#include<stdio.h>
struct pair
{
  int min;
```

Google™ Custom Search

GeeksforGeeks

53,522 people like GeeksforGeeks.

Interview Experiences
Advanced Data Structures
Dynamic Programming
Greedy Algorithms
Backtracking
Pattern Searching
Divide & Conquer
Mathematical Algorithms
Recursion

```
  int max;
};

struct pair getMinMax(int arr[], int n)
{
  struct pair minmax;
  int i;

  /*If there is only one element then return it as min and max both*/
  if (n == 1)
  {
     minmax.max = arr[0];
     minmax.min = arr[0];
     return minmax;
  }

  /* If there are more than one elements, then initialize min
      and max*/
  if (arr[0] > arr[1])
  {
      minmax.max = arr[0];
      minmax.min = arr[1];
  }
  else
  {
      minmax.max = arr[1];
      minmax.min = arr[0];
  }

  for (i = 2; i<n; i++)
  {
    if (arr[i] >  minmax.max)
      minmax.max = arr[i];

    else if (arr[i] <  minmax.min)
      minmax.min = arr[i];
  }

  return minmax;
}

/* Driver program to test above function */
int main()
{
  int arr[] = {1000, 11, 445, 1, 330, 3000};
  int arr_size = 6;
  struct pair minmax = getMinMax (arr, arr_size);
```

Geometric Algorithms

Popular Posts

All permutations of a given string
Memory Layout of C Programs
Understanding “extern” keyword in C
Median of two sorted arrays
Tree traversal without recursion and without stack!
Structure Member Alignment, Padding and Data Packing
Intersection point of two Linked Lists
Lowest Common Ancestor in a BST.
Check if a binary tree is BST or not
Sorted Linked List to Balanced BST

```
  printf("\nMinimum element is %d", minmax.min);
  printf("\nMaximum element is %d", minmax.max);
  getchar();
}
```

Time Complexity: O(n)

In this method, total number of comparisons is 1 + 2(n-2) in worst case and 1 + n – 2 in best case.
In the above implementation, worst case occurs when elements are sorted in descending order and best case occurs when elements are sorted in ascending order.

METHOD 2 (Tournament Method)
Divide the array into two parts and compare the maximums and minimums of the the two parts to get the maximum and the minimum of the the whole array.

```
Pair MaxMin(array, array_size)
   if array_size = 1
      return element as both max and min
   else if arry_size = 2
      one comparison to determine max and min
      return that pair
   else    /* array_size  > 2 */
      recur for max and min of left half
      recur for max and min of right half
      one comparison determines true max of the two candidates
      one comparison determines true min of the two candidates
      return the pair of max and min
```

Implementation

```
/* structure is used to return two values from minMax() */
#include<stdio.h>
struct pair
{
  int min;
  int max;
};
```

```
struct pair getMinMax(int arr[], int low, int high)
{
  struct pair minmax, mml, mmr;
  int mid;

  /* If there is only on element */
  if (low == high)
  {
     minmax.max = arr[low];
     minmax.min = arr[low];
     return minmax;
  }

  /* If there are two elements */
  if (high == low + 1)
  {
     if (arr[low] > arr[high])
     {
        minmax.max = arr[low];
        minmax.min = arr[high];
     }
     else
     {
        minmax.max = arr[high];
        minmax.min = arr[low];
     }
     return minmax;
  }

  /* If there are more than 2 elements */
  mid = (low + high)/2;
  mml = getMinMax(arr, low, mid);
  mmr = getMinMax(arr, mid+1, high);

  /* compare minimums of two parts*/
  if (mml.min < mmr.min)
    minmax.min = mml.min;
  else
    minmax.min = mmr.min;

  /* compare maximums of two parts*/
  if (mml.max > mmr.max)
    minmax.max = mml.max;
  else
    minmax.max = mmr.max;

  return minmax;
```

Recent Comments

Aman Hi, Why arent we checking for conditions...

Write a C program to Delete a Tree. · 24 minutes ago

kzs please provide solution for the problem...

Backtracking | Set 2 (Rat in a Maze) · 28 minutes ago

Sanjay Agarwal bool tree::Root_to_leaf_path_given_sum(tree...

Root to leaf path sum equal to a given number · 53 minutes ago

GOPI GOPINATH @admin Highlight this sentence "We can easily...

Count trailing zeroes in factorial of a number · 54 minutes ago

newCoder3006 If the array contains negative numbers also. We...

Find subarray with given sum · 1 hour ago

newCoder3006 Code without using while loop. We can do it...

Find subarray with given sum · 1 hour ago

```
}

/* Driver program to test above function */
int main()
{
  int arr[] = {1000, 11, 445, 1, 330, 3000};
  int arr_size = 6;
  struct pair minmax = getMinMax(arr, 0, arr_size-1);
  printf("\nMinimum element is %d", minmax.min);
  printf("\nMaximum element is %d", minmax.max);
  getchar();
}
```

Time Complexity: O(n)
Total number of comparisons: let number of comparisons be T(n). T(n) can be written as follows:
Algorithmic Paradigm: Divide and Conquer

```
T(n) = T(floor(n/2)) + T(ceil(n/2)) + 2
T(2) = 1
T(1) = 0
```

If n is a power of 2, then we can write T(n) as:

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

After solving above recursion, we get

```
T(n)  = 3/2n -2
```

Thus, the approach does 3/2n -2 comparisons if n is a power of 2. And it does more than 3/2n -2 comparisons if n is not a power of 2.

METHOD 3 (Compare in Pairs)
If n is odd then initialize min and max as first element.
If n is even then initialize min and max as minimum and maximum of the first two elements respectively.
For rest of the elements, pick them in pairs and compare their
maximum and minimum with max and min respectively.

```
#include<stdio.h>

/* structure is used to return two values from minMax() */
struct pair
{
  int min;
  int max;
};

struct pair getMinMax(int arr[], int n)
{
  struct pair minmax;
  int i;

  /* If array has even number of elements then
    initialize the first two elements as minimum and
    maximum */
  if (n%2 == 0)
  {
    if (arr[0] > arr[1])
    {
      minmax.max = arr[0];
      minmax.min = arr[1];
    }
    else
    {
      minmax.min = arr[0];
      minmax.max = arr[1];
    }
    i = 2;  /* set the startung index for loop */
  }

   /* If array has odd number of elements then
    initialize the first element as minimum and
    maximum */
  else
  {
    minmax.min = arr[0];
    minmax.max = arr[0];
    i = 1;  /* set the startung index for loop */
  }

  /* In the while loop, pick elements in pair and
     compare the pair with max and min so far */
  while (i < n-1)
  {
    if (arr[i] > arr[i+1])
    {
      if(arr[i] > minmax.max)
        minmax.max = arr[i];
      if(arr[i+1] < minmax.min)
        minmax.min = arr[i+1];
    }
    else
    {
      if (arr[i+1] > minmax.max)
        minmax.max = arr[i+1];
      if (arr[i] < minmax.min)
        minmax.min = arr[i];
    }
    i += 2; /* Increment the index by 2 as two
               elements are processed in loop */
  }

  return minmax;
}

/* Driver program to test above function */
int main()
{
  int arr[] = {1000, 11, 445, 1, 330, 3000};
  int arr_size = 6;
  struct pair minmax = getMinMax (arr, arr_size);
  printf("\nMinimum element is %d", minmax.min);
  printf("\nMaximum element is %d", minmax.max);
  getchar();
}
```

Time Complexity: O(n)

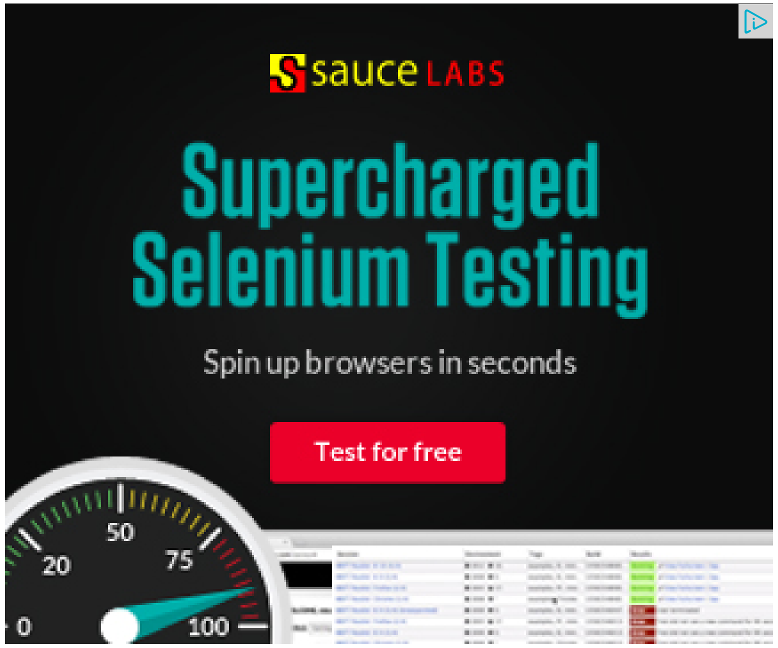
Total number of comparisons: Different for even and odd n, see below:

```
       If n is odd:    3*(n-1)/2
       If n is even:   1 Initial comparison for initializing min and max,
                           and 3(n-2)/2 comparisons for rest of the elements
                     =  1 + 3*(n-2)/2 = 3n/2 -2
```

Second and third approaches make equal number of comparisons when n is a power of 2.

In general, method 3 seems to be the best.

Please write comments if you find any bug in the above programs/algorithms or a better way to solve the same problem.



Related Tpoics:

- Remove minimum elements from either side such that 2*min becomes more than max
- Divide and Conquer | Set 6 (Search in a Row-wise and Column-wise Sorted 2D Array)
- Bucket Sort
- Kth smallest element in a row-wise and column-wise sorted 2D array | Set 1
- Find the number of zeroes
- Find if there is a subarray with 0 sum
- Divide and Conquer | Set 5 (Strassen's Matrix Multiplication)
- Count all possible groups of size 2 or 3 that have sum as multiple of 3

18 Tweet 0 0

Writing code in comment? Please use **ideone.com** and share the link here.

34 Comments **GeeksforGeeks**

Sort by Newest

Join the discussion…

OM · 20 days ago

In this method 1, total number of comparisons is 1 + 2(n-2) in worst case and
In the above implementation, worst case occurs when elements are sorted in
initially only we get the max element. so complexity will be same for both asce

^ | ˅ · Reply · Share ›

xxmajia · 3 months ago

how about that, use the below max(a,b) and min(a,b) function that can give yo
comparision at all, then do a linear search, with time complexisty O(N), and 0

max(a,b) {
c = a-b;
k = c>>31;
return a-k*c;
}

min(a, b) {
return a+b-max(a,b);
}

^ | ˅ · Reply · Share ›

zealfire · 4 months ago

how is t(n)=2(n/2)+2,=>3n/2-2

^ | ˅ · Reply · Share ›

Aniket · 8 months ago

It can be done by using merge sort by nlogn time complexity.While merging tw
element of both the array that will give smallest and by comparing the last iterr

Marsha Donna · 8 months ago

can sum1 pls explain the logic behind method3

Reply · Share ›

Gautam · 11 months ago

In 1st method if else block should be like this...
else
{
minmax.max = arr[1];
minmax.min = arr[0];
}

2 · Reply · Share ›

Marsha Donna → Gautam · 8 months ago

yes ur right

Reply · Share ›

GeeksforGeeks Mod → Marsha Donna · 8 months ago

Thanks for pointing this out. We have updated the code of 1st r

Reply · Share ›

Gautam · 11 months ago

In 1st method the if else block should not be like this??
if(arr[0] > arr[1])
{
minmax.max = arr[0];
minmax.min = arr[1];
}
else
{

minmax.max = arr[1];
minmax.min = arr[0];
}

^ | ⌄ • Reply • Share ›

radhakrishna • 3 years ago

DPM solution. let me know if there is anything wrong about it.

for i : 1 to n array

MinMax[i] : min max pair of array window ending at i

base case : (a[1], a[1]) if i =1

general case : Max(MinMax[i-1].max, a[i]) , Min(MinMax[i-1].min,a[i])

^ | ⌄ • Reply • Share ›

bhavneet ➜ radhakrishna • a year ago

it has got 2*(n-1) comparisons

```
/* Paste your code here (You may delete these lines if not wr:
```

^ | ⌄ • Reply • Share ›

jiabul(ju) • 3 years ago

/*maxmin*/

int maximum(int a[],int low,int right)
{
int m1 ;
int m2,mid;
if(low&rt;right)
{

```
m1=maximum(a,0,mid);
m2=maximum(a,mid+1,right);
return ((m1&rt;m2)?m1:m2);

}

int minimum(int a[],int low,int right)
{
int m1 ;
int m2,mid;
```

see more

Reply • Share ›

jiabul(ju) • 3 years ago

it can be done by o(log n).

Reply • Share ›

Priyanka Gupta jiabul(ju) • 7 months ago

exactly that wt i thought using divide and conquer

Reply • Share ›

Rajneesh • 3 years ago

Can't we do a radix sort and get the result without even comparing at all?

Reply • Share ›

shrivats Rajneesh • 2 years ago

Sorting is o(nlgn)

```
/* Paste your code here (You may delete these lines if not wr
```

Reply • Share ›

abc ➜ shrivats • a year ago

Radix Sort is O(N)

```
/* Paste your code here (You may delete these lines if
```

• Reply • Share ›

WgpShashank • 3 years ago

Proof of TC of Method 2

T(2)=1
T(n)= 2·T(n/2)+2 for n>2 (recurrence)
----
n=2^m //every where its n=2^m

T(2m)=2·T(2m-1)+2
T(2m)=2·(2T(2m-2)+2)+2…
.
T(2m)=2m-1·T(21)+2m-1+2m- 2+…+21 //2^m-1

//also we know : T(n)=2n-1+2n-2+…+2+1=2n-1

T(2m)=2m-1·T(21)+2m-2=3·2m/2-2=3n/2-2.

1 • Reply • Share ›

pramod • 3 years ago

we can do it by making max heap and min heap .O(n)

• Reply • Share ›

bala ➜ pramod • 3 years ago

Building a Max and Min heap are both O(nlogn) time complexities , not convey something else

Sandeep ➦ bala • 3 years ago
@bala: Building a Heap is O(n), not O(nLogn). Please refer this
^ | ⌄ • Reply • Share ›

Algoseekar ➦ Sandeep • 3 years ago
@sandeep can u please provide the poof of time compl
asap..it doesn't comes out to be what u says..???
^ | ⌄ • Reply • Share ›

apps ➦ Algoseekar • 3 years ago
height...no. of comparision
------...---------------
.1.........0
.2..........N/2___-- 2 elm comp
.3........2*(N/4) |
.4........2*(N/8) |..2 elm comp
.... |--for min &
.... |..max
.N.........2*1___|
----
^ | ⌄ • Reply • Share ›

apps ➦ Algoseekar • 3 years ago
just think in general way...

leaf nodes will be 'N'

internal nodes = 'N-1'

height no. of comparision
----- ------------------

1 0
2 N/2 __-- 2 elm comp
3 2*(N/4) |
4 2*(N/8) | 2 elm comp
. . |--for min &
. . | max
N 2*1 __|
----

noc = N/2 + 2 * [N/4 + N/8 + ... + 1]
noc = N/2 + N/4 + N/8 + ... + 1 +

see more

1 ^ | v • Reply • Share ›

bala ➦ Sandeep • 3 years ago
@Sandeep : yes, I am wrong. The max heap build takes
asymptotically tighter bound is O(n). Thanks for remindi

^ | v • Reply • Share ›

Venki • 4 years ago
Refer Sara Baase (Computer Algorithms) and Dromey (How to solve it by con

^ | v • Reply • Share ›

Anunay • 4 years ago
Instead of creating struct and additional checks before starting the loop, Metho

[sourcecode language="java"]
public void MinMaxInArray()
{
int[] arr = { 11, 14, 5, 140 };
int min = int.MaxValue;

```
for (int i = 0; i < arr.Length; i++)
{
if (arr[i] < min)
min = arr[i];

if (arr[i] > max)
max = arr[i];
}

Console.WriteLine("Min: {0}\nMax: {1}", min, max );
}
```

Reply • Share ›

Gauri • 4 years ago

In case of method 2 , how solving the recurrence relation

T(n) = 2*T(n/2) + 2 and T(2) = 1 and T(1) = 0

comes out to be

T(n) = 3/2n -2

Since solving this recurrence as following results in T(n) = 2*(n -1)

T(n/2) = 2*T(n/4) + 2 we have
T(n) = 2*(2*T(n/4) + 2)) + 2 = 4*T(n/4) + 4 +2
substituting for T(n/4) we obtain
T(n) = 4*(2*T(n/8) + 2)) + 4 + 2 = 8*T(n/8) + 8 + 4 + 2

Continuing in this manner we find that
T(n) = pow(2,k) * T(n/pow(2,k)) + 2*(pow(2,k) -1)

After k=log2(n) substitutions we have

T(n) = pow(2,log2(n))*T(n/pow(2,log2(n))) + 2*(pow(2,log2(n)) -1)
Noting that pow(2,log2(n)) = n we have

T(n) = n * T(1) +2*(n -1)
= 2*n-2

Please point out the error in the above calculation

• Reply • Share ›

abc Gauri • 3 years ago
The complexity calculated in this comment is correct. Right?
I am confused as I don't see any replies to this and even the post has
please clarify

• Reply • Share ›

GeeksforGeeks • 4 years ago
@B: We have modified the code to make the suggested optimization.

In method 1, total number of comparisons (after the suggested optimization) is
in best case.
In the above implementation, worst case occurs when elements are sorted in
occurs when elements are sorted in ascending order.

• Reply • Share ›

init GeeksforGeeks • 4 years ago
still the complexity of method 2 is 2*(n-1)???
correct me if i am wrong!!

• Reply • Share ›

Vipul init • 6 months ago
worst case occur while sorting in both ascending or descendin
checking first i.e. maximum or minimm

Reply • Share ›

B • 4 years ago

for method1,

```
 for(i = 2; i<n; i++)
{
  if(arr[i] >  minmax.max)
    minmax.max = arr[i];

  if(arr[i] <  minmax.min)
    minmax.min = arr[i];
}



Should be optimized to
  for(i = 2; i<n; i++)
  {
    if(arr[i] >  minmax.max)
      minmax.max = arr[i];

    else if(arr[i] <  minmax.min)
      minmax.min = arr[i];
  }
```

So, the no. of comparisons would pretty much be 3n/2 for all of the methods.
I would not recommend Method 2, as it would have huge function-loads overhe

Reply • Share ›

Dishant → B • 25 days ago

Thanks so much for sharing this.

Reply • Share ›

Subscribe Add Disqus to your site