

Maximum and minimum of an array using minimum number of comparisons

Difficulty Level : Medium • Last Updated : 27 Feb, 2021

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

[Recommended: Please try your approach on **{IDE}** first, before moving on to the solution.](#)

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.

C

```
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)

C++

```
// C++ program of above implementation
#include<iostream>
using namespace std;

// Pair struct is used to return
// two values from getMinMax()
struct Pair
{
    int min;
    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 code
int main()
{
    int arr[] = { 1000, 11, 445,
                  1, 330, 3000 };
    int arr_size = 6;

    struct Pair minmax = getMinMax(arr, arr_size);

    cout << "Minimum element is "
         << minmax.min << endl;
    cout << "Maximum element is "
         << minmax.max;

    return 0;
}

```



Related Articles

```

/* structure is used to return two values from minMax() */
#include<stdio.h>
struct pair
{
    int min;
    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 element, 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);
    printf("nMinimum element is %d", minmax.min);
    printf("nMaximum element is %d", minmax.max);
    getchar();
}

```

Java

```

// Java program of above implementation
public class GFG {
    /* Class Pair is used to return two values from getMinMax() */
    static class Pair {

        int min;
        int max;
    }

    static Pair getMinMax(int arr[], int n) {
        Pair minmax = new Pair();
        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 */
public static void main(String args[]) {
    int arr[] = {1000, 11, 445, 1, 330, 3000};
    int arr_size = 6;
    Pair minmax = getMinMax(arr, arr_size);
    System.out.printf("\nMinimum element is %d", minmax.min);
    System.out.printf("\nMaximum element is %d", minmax.max);
}
}

```

Python3

```

# Python program of above implementation

# structure is used to return two values from minMax()

class pair:

```

```

def __init__(self):
    self.min = 0
    self.max = 0

def getMinMax(arr: list, n: int) -> pair:
    minmax = pair()

    # 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 in range(2, n):
        if arr[i] > minmax.max:
            minmax.max = arr[i]
        elif arr[i] < minmax.min:
            minmax.min = arr[i]

    return minmax

# Driver Code
if __name__ == "__main__":
    arr = [1000, 11, 445, 1, 330, 3000]
    arr_size = 6
    minmax = getMinMax(arr, arr_size)
    print("Minimum element is", minmax.min)
    print("Maximum element is", minmax.max)

# This code is contributed by
# sanjeev2552

```

C#

```

// C# program of above implementation
using System;

class GFG
{
    /* Class Pair is used to return

```

```
two values from getMinMax() */
class Pair
{
    public int min;
    public int max;
}

static Pair getMinMax(int []arr, int n)
{
    Pair minmax = new Pair();
    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 Code
public static void Main(String []args)
{
```



```

int []arr = {1000, 11, 445, 1, 330, 3000};
int arr_size = 6;
Pair minmax = getMinMax(arr, arr_size);
Console.WriteLine("Minimum element is {0}",
                  minmax.min);
Console.WriteLine("\nMaximum element is {0}",
                  minmax.max);
    }
}

// This code is contributed by PrinciRaj1992

```

Output:

```

Minimum element is 1
Maximum element is 3000

```

Time Complexity: $O(n)$

In this method, the total number of comparisons is $1 + 2(n-2)$ in the worst case and $1 + n - 2$ in the best case.

In the above implementation, the worst case occurs when elements are sorted in descending order and the 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 two parts to get the maximum and the minimum of the whole array.

```

Pair MaxMin(array, array_size)
    if array_size = 1
        return element as both max and min
    else if array_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

C++

```
// C++ program of above implementation
#include<iostream>
using namespace std;

// structure is used to return
// two values from minMax()
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 one 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;
}

// Driver code
int main()
{
    int arr[] = { 1000, 11, 445,
                  1, 330, 3000 };
    int arr_size = 6;

    struct Pair minmax = getMinMax(arr, 0,
                                     arr_size - 1);

    cout << "Minimum element is "
          << minmax.min << endl;
    cout << "Maximum element is "
          << minmax.max;

    return 0;
}

// This code is contributed by nik_3112

```

C

```

/* 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 one 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;
}
```



```

/* 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();
}

```

Java

```

// Java program of above implementation
public class GFG {
    /* Class Pair is used to return two values from getMinMax() */
    static class Pair {

        int min;
        int max;
    }

    static Pair getMinMax(int arr[], int low, int high) {
        Pair minmax = new Pair();
        Pair mml = new Pair();
        Pair mmr = new Pair();
        int mid;

        // If there is only one 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;
}

/* Driver program to test above function */
public static void main(String args[]) {
    int arr[] = {1000, 11, 445, 1, 330, 3000};
    int arr_size = 6;
    Pair minmax = getMinMax(arr, 0, arr_size - 1);
    System.out.printf("\nMinimum element is %d", minmax.min);
    System.out.printf("\nMaximum element is %d", minmax.max);
}
}

```

Python3

Python program of above implementation

```

def getMinMax(low, high, arr):
    arr_max = arr[low]
    arr_min = arr[low]

    # If there is only one element
    if low == high:
        arr_max = arr[low]
        arr_min = arr[low]
        return (arr_max, arr_min)

    # If there is only two element
    elif high == low + 1:
        if arr[low] > arr[high]:
            arr_max = arr[low]

```



```

        arr_min = arr[high]
    else:
        arr_max = arr[high]
        arr_min = arr[low]
    return (arr_max, arr_min)
else:

    # If there are more than 2 elements
    mid = int((low + high) / 2)
    arr_max1, arr_min1 = getMinMax(low, mid, arr)
    arr_max2, arr_min2 = getMinMax(mid + 1, high, arr)

    return (max(arr_max1, arr_max2), min(arr_min1, arr_min2))

# Driver code
arr = [1000, 11, 445, 1, 330, 3000]
high = len(arr) - 1
low = 0
arr_max, arr_min = getMinMax(low, high, arr)
print('Minimum element is ', arr_min)
print('Maximum element is ', arr_max)

# This code is contributed by DeepakChhitarka

```

C#



```

// C# implementation of the approach
using System;

public class GFG {
    /* Class Pair is used to return two values from getMinMax() */
    public class Pair {

        public int min;
        public int max;
    }

    static Pair getMinMax(int []arr, int low, int high) {
        Pair minmax = new Pair();
        Pair mml = new Pair();
        Pair mmr = new Pair();
        int mid;

        // If there is only one 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;
}

/* Driver program to test above function */
public static void Main(String []args) {
    int []arr = {1000, 11, 445, 1, 330, 3000};
    int arr_size = 6;
    Pair minmax = getMinMax(arr, 0, arr_size - 1);
    Console.WriteLine("\nMinimum element is {0}", minmax.min);
    Console.WriteLine("\nMaximum element is {0}", minmax.max);
}
}

// This code contributed by Rajput-Ji

```

Output:



Minimum element is 1

Maximum element is 3000

Time Complexity: $O(n)$

Total number of comparisons: let the number of comparisons be $T(n)$. $T(n)$ can be written as follows:

Algorithmic Paradigm: Divide and Conquer

$$T(n) = T(\text{floor}(n/2)) + T(\text{ceil}(n/2)) + 2$$

$$T(2) = 1$$

$$T(1) = 0$$

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



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

After solving the above recursion, we get

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

Thus, the approach does $3n/2 - 2$ comparisons if n is a power of 2. And it does more than $3n/2 - 2$ 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.

C++

```
// C++ program of above implementation
#include<iostream>
using namespace std;

// 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];
        }

        // Set the starting index for loop
        i = 2;
    }

    // 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];

    // Set the starting index for loop
    i = 1;
}

// 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];
    }

    // Increment the index by 2 as
    // two elements are processed in loop
    i += 2;
}
return minmax;
}

// Driver code
int main()
{
    int arr[] = { 1000, 11, 445,
                  1, 330, 3000 };
    int arr_size = 6;

    Pair minmax = getMinMax(arr, arr_size);

    cout << "nMinimum element is "
         << minmax.min << endl;
    cout << "nMaximum element is "
         << minmax.max;
```



```
    return 0;
}

// This code is contributed by nik_3112
```

C

```
#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 starting 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 starting 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();
}

```

Java

```

// Java program of above implementation
public class GFG {

    /* Class Pair is used to return two values from getMinMax() */
    static class Pair {

        int min;
        int max;
    }

    static Pair getMinMax(int arr[], int n) {

```

```

Pair minmax = new Pair();
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 starting 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 starting 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  tion */

```

public static void main(String args[]) {
    int arr[] = {1000, 11, 445, 1, 330, 3000};
    int arr_size = 6;
    Pair minmax = getMinMax(arr, arr_size);
    System.out.printf("\nMinimum element is %d", minmax.min);
    System.out.printf("\nMaximum element is %d", minmax.max);
}
}

```

Python3

Python3 program of above implementation

```

def getMinMax(arr):

    n = len(arr)

    # If array has even number of elements then
    # initialize the first two elements as minimum
    # and maximum
    if(n % 2 == 0):
        mx = max(arr[0], arr[1])
        mn = min(arr[0], arr[1])

        # set the starting index for loop
        i = 2

    # If array has odd number of elements then
    # initialize the first element as minimum
    # and maximum
    else:
        mx = mn = arr[0]

        # set the starting index for loop
        i = 1

    # 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]:
            mx = max(mx, arr[i + 1])
            mn = min(mn, arr[i])
        else:
            mx = max(mx, arr[i])
            mn = min(mn, arr[i + 1])

        # Increment the index by 2 as two
        # elements are processed in
        i += 2

```

```
return (mx, mn)
```

```
# Driver Code
```

```
if __name__ == '__main__':
```

```
    arr = [1000, 11, 445, 1, 330, 3000]
```

```
    mx, mn = getMinMax(arr)
```

```
    print("Minimum element is", mn)
```

```
    print("Maximum element is", mx)
```

```
# This code is contributed by Kaustav
```

C#

```
// C# program of above implementation
```

```
using System;
```

```
class GFG
```

```
{
```

```
    /* Class Pair is used to return
       two values from getMinMax() */
```

```
    public class Pair
```

```
    {
```

```
        public int min;
```

```
        public int max;
```

```
    }
```

```
    static Pair getMinMax(int []arr, int n)
```

```
    {
```

```
        Pair minmax = new Pair();
```

```
        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 starting 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 starting 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 Code
public static void Main(String args)

```



```

{
    int []arr = {1000, 11, 445, 1, 330, 3000};
    int arr_size = 6;
    Pair minmax = getMinMax(arr, arr_size);
    Console.WriteLine("Minimum element is {0}",
                      minmax.min);
    Console.WriteLine("\nMaximum element is {0}",
                      minmax.max);
}
}

// This code is contributed by 29AjayKumar

```

Output:

```

Minimum element is 1
Maximum element is 3000

```

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
                  and 3(n-2)/2 comparisons for rest of the e
                  = 1 + 3*(n-2)/2 = 3n/2 - 2

```

Second and third approaches make the 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.

Like 0



[Previous](#)[Next](#)

RECOMMENDED ARTICLES

Page : [1](#) [2](#) [3](#)

- 01

Search an element in an unsorted array using minimum number of comparisons
22, Jun 17
- 02

Second minimum element using minimum comparisons
21, Dec 16
- 03

Middle of three using minimum comparisons
07, Dec 17
- 04

Number of comparisons in each direction for m queries in linear search
05, Dec 18
- 05

Minimum number greater than the maximum of array which cannot be formed using the numbers in the array
23, Apr 19
- 06

Maximum XOR value of maximum and second maximum element among all possible subarrays
08, Jan 20
- 07

Minimum distance between the maximum and minimum element of a given Array
16, Jul 20
- 08

Partition array into two subsets with minimum Bitwise XOR between their maximum and minimum
20, Jan 21

Article Contributed By :

Vote for difficulty

Current difficulty : [Medium](#)

[Easy](#)[Normal](#)[Medium](#)[Hard](#)[Expert](#)

Improved By : [kamleshbhalui](#), [Rajput-Ji](#), [Kaustav kumar Chanda](#), [Akanksha_Rai](#), [princiraj1992](#), [29AjayKumar](#), [sanjeev2552](#), [DeepakChhitarka](#), [nik_3112](#), [maafkaroplz](#), [anjalitejasvi501](#), [anshulpurohit11](#), [dhairyabahl5](#)

Article Tags : [Numbers](#), [Arrays](#), [Divide and Conquer](#), [Searching](#)

Practice Tags : [Arrays](#), [Searching](#), [Divide and Conquer](#), [Numbers](#)

[Improve Article](#)[Report Issue](#)

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

[Load Comments](#)

5th Floor, A-118,
Sector-136, Noida, Uttar Pradesh - 201305

feedback@geeksforgeeks.org

[Company](#)[About Us](#)[Learn](#)[Algorithms](#)

Careers
Privacy Policy
Contact Us
Copyright Policy

Data Structures
Languages
CS Subjects
Video Tutorials

Practice

Courses
Company-wise
Topic-wise
How to begin?

Contribute

Write an Article
Write Interview Experience
Internships
Videos

@geeksforgeeks , Some rights reserved

