Task name: Majority element

Task number: 2

ID	Full Name
20210002	ابانوب ابراهيم صلاح
20210011	ابوبكر خالد ابوبكر
20210016	احمد ابراهيم عبدالله
20210027	احمد السيد خليل احمد
20210032	احمد ايمن محمد عبد الغفار سكر

Non-recursive algorithm

pseudocode

```
function find_majority_element(arr, n):
   count[n]
   for i = 0 to n-1 do
      count[i] = 0
   majority = -1
   max_count = 0
   for i = 0 to n-1 do
      count[arr[i]] = count[arr[i]] + 1
      if count[arr[i]] > max_count then
         majority = arr[i]
         max_count = count[arr[i]]
   if max_count > n/2 then
      return majority
   else:
      return -1


function main():
   arr = [3, 3, 4, 2, 4, 4, 2, 4, 4]
   n = size of arr
   majority_element = find_majority_element(arr, n)
   if majority_element != -1 then
      print "The majority element is ", majority_element
   else:
      print "There is no majority element in the array"
   return 0
```

Analysis

1. `int count[n];` - Time complexity is O(1)

2. `for (int i = 0; i < n; i++) { count[i] = 0; }` - Time complexity is O(n)

3. `int majority = -1;` - Time complexity is O(1)

4. `int max_count = 0;` - Time complexity is O(1)

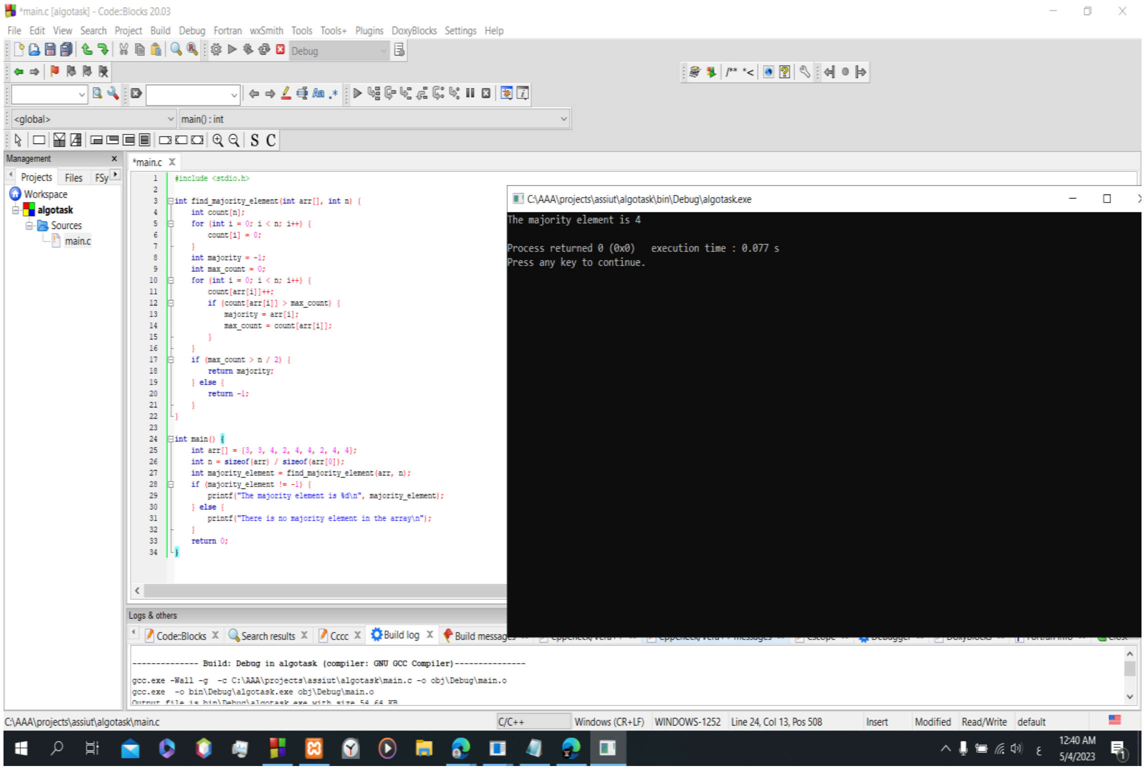
5. `for (int i = 0; i < n; i++) { count[arr[i]]++; ... }` - Time complexity is O(n)

6. `if (count[arr[i]] > max_count) { majority = arr[i]; max_count = count[arr[i]]; }` - Time complexity is O(1)

7. `if (max_count > n / 2) { return majority; } else { return -1; }` - Time complexity is O(1)

The time complexity of the `find_majority_element` function is O(n)

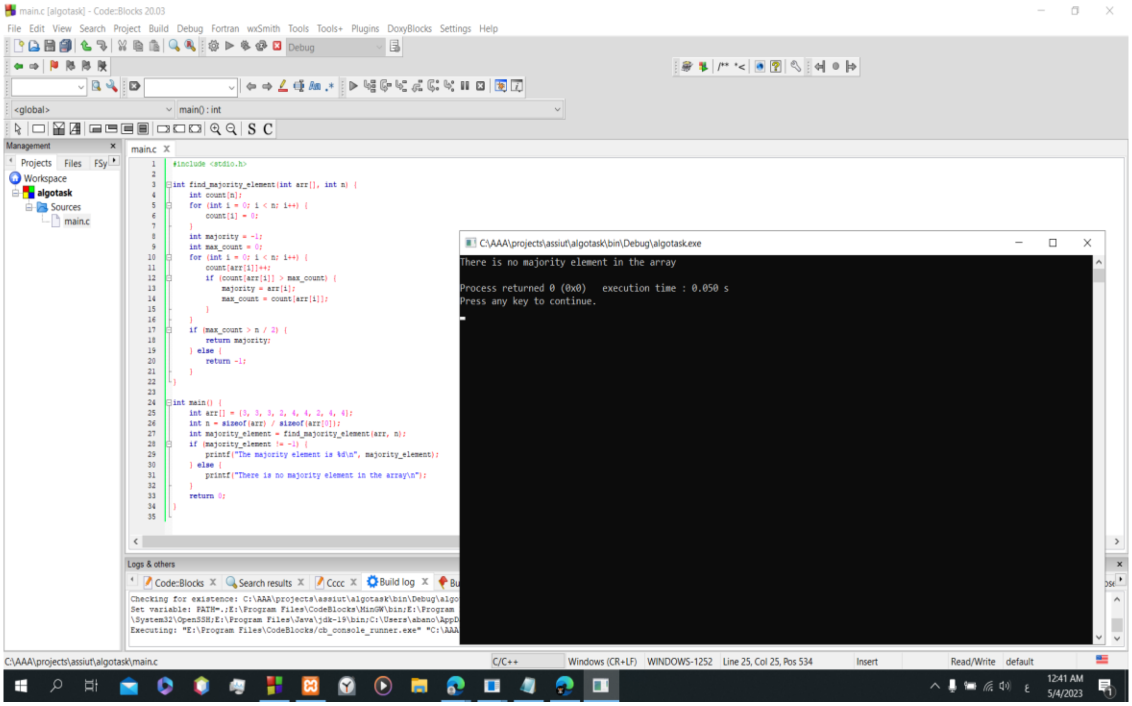
Output


*main.c [algotask] - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Debug
<global>
main() : int
Management
Projects Files FSy
Workspace
algotask
Sources
main.c
*main.c
C:\AAA\projects\assiut\algotask\bin\Debug\algotask.exe
Logs & others
Code::Blocks Search results Cccc Build log Build messages
C:\AAA\projects\assiut\algotask\main.c
C/C++ Windows (CR+LF) WINDOWS-1252 Line 24, Col 13, Pos 508 Insert Modified Read/Write default
12:40 AM
5/4/2023

```c
#include <stdio.h>

int find_majority_element(int arr[], int n) {
    int count[n];
    for (int i = 0; i < n; i++) {
        count[i] = 0;
    }
    int majority = -1;
    int max_count = 0;
    for (int i = 0; i < n; i++) {
        count[arr[i]]++;
        if (count[arr[i]] > max_count) {
            majority = arr[i];
            max_count = count[arr[i]];
        }
    }
    if (max_count > n / 2) {
        return majority;
    } else {
        return -1;
    }
}

int main() {
    int arr[] = {3, 3, 4, 2, 4, 4, 2, 4, 4};
    int n = sizeof(arr) / sizeof(arr[0]);
    int majority_element = find_majority_element(arr, n);
    if (majority_element != -1) {
        printf("The majority element is %d\n", majority_element);
    } else {
        printf("There is no majority element in the array\n");
    }
    return 0;
}
```

```
The majority element is 4

Process returned 0 (0x0)   execution time : 0.077 s
Press any key to continue.
```

```
-------------- Build: Debug in algotask (compiler: GNU GCC Compiler)---------------

gcc.exe -Wall -g  -c C:\AAA\projects\assiut\algotask\main.c -o obj\Debug\main.o
gcc.exe  -o bin\Debug\algotask.exe obj\Debug\main.o
Output file is bin\Debug\algotask.exe with size 54.64 KB
```


main.c [algotask] - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Debug
<global>
main() : int
Management
Projects Files FSy
Workspace
algotask
Sources
main.c
main.c
C:\AAA\projects\assiut\algotask\bin\Debug\algotask.exe
Logs & others
Code::Blocks Search results Cccc Build log
C:\AAA\projects\assiut\algotask\main.c
C/C++ Windows (CR+LF) WINDOWS-1252 Line 25, Col 25, Pos 534 Insert Read/Write default
12:41 AM
5/4/2023

```c
#include <stdio.h>

int find_majority_element(int arr[], int n) {
    int count[n];
    for (int i = 0; i < n; i++) {
        count[i] = 0;
    }
    int majority = -1;
    int max_count = 0;
    for (int i = 0; i < n; i++) {
        count[arr[i]]++;
        if (count[arr[i]] > max_count) {
            majority = arr[i];
            max_count = count[arr[i]];
        }
    }
    if (max_count > n / 2) {
        return majority;
    } else {
        return -1;
    }
}

int main() {
    int arr[] = {3, 3, 3, 2, 4, 4, 2, 4, 4};
    int n = sizeof(arr) / sizeof(arr[0]);
    int majority_element = find_majority_element(arr, n);
    if (majority_element != -1) {
        printf("The majority element is %d\n", majority_element);
    } else {
        printf("There is no majority element in the array\n");
    }
    return 0;
}

```

```
There is no majority element in the array

Process returned 0 (0x0)   execution time : 0.050 s
Press any key to continue.
```

```
Checking for existence: C:\AAA\projects\assiut\algotask\bin\Debug\algo
Set variable: PATH=.;E:\Program Files\CodeBlocks\MinGW\bin;E:\Program
\System32\OpenSSH;E:\Program Files\Java\jdk-19\bin;C:\Users\abano\AppD
Executing: "E:\Program Files\CodeBlocks/cb_console_runner.exe" "C:\AAA
```

Recursive algorithm

Pseudocode

```
FUNCTION findMajorityElement(array[], size)
    IF size equals 1 THEN(
        RETURN array[0]
      )
    SET mid to size divided by 2
    SET leftMajority to findMajorityElement(array, mid)
    SET rightMajority to findMajorityElement(array + mid, size - mid)
    IF leftMajority equals rightMajority THEN(
        RETURN leftMajority
)
    SET leftCount to 0
    SET rightCount to 0
    FOR i from 0 to size-1 DO(
        IF array[i] equals leftMajority THEN
            INCREMENT leftCount by 1
        ELSEIF array[i] equals rightMajority THEN
            INCREMENT rightCount by 1

   )
    IF leftCount is greater than size divided by 2 THEN
        RETURN leftMajority
    ELSEIF rightCount is greater than size divided by 2 THEN
        RETURN rightMajority
    ELSE
        RETURN -1
END FUNCTION
```

```
main()
  DECLARE arr as an array of integers with values {3, 3, 4, 2, 4, 4, 2, 4, 4}
  DECLARE size as the length of arr divided by the length of the first element of arr

  SET majorityElement to the result of calling findMajorityElement with arguments arr and size

  PRINT "The majority element of the array is: " concatenated with majorityElement
```

Analysis

1. `if (size == 1) { return array[0]; }` - Time complexity is O(1).

2. `int mid = size / 2;` - Time complexity is O(1)

3. `int leftMajority = findMajorityElement(array, mid);` - Time complexity is T(n/2), where T is the time complexity of the function and n is the size of the array.

4. `int rightMajority = findMajorityElement(array + mid, size - mid);` - Time complexity is T(n/2), where T is the time complexity of the function and n is the size of the array

5. `if (leftMajority == rightMajority) { return leftMajority; }` - Time complexity is O(1).

6. `int leftCount = 0, rightCount = 0;` - Time complexity is O(1).

7. `for (int i = 0; i < size; i++) { ... }` - Time complexity is O(n).

8. `if (leftCount > size/2) { return leftMajority; }` - Time complexity is O(1).

9. `else if (rightCount > size/2) { return rightMajority; }` - Time complexity is O(1).

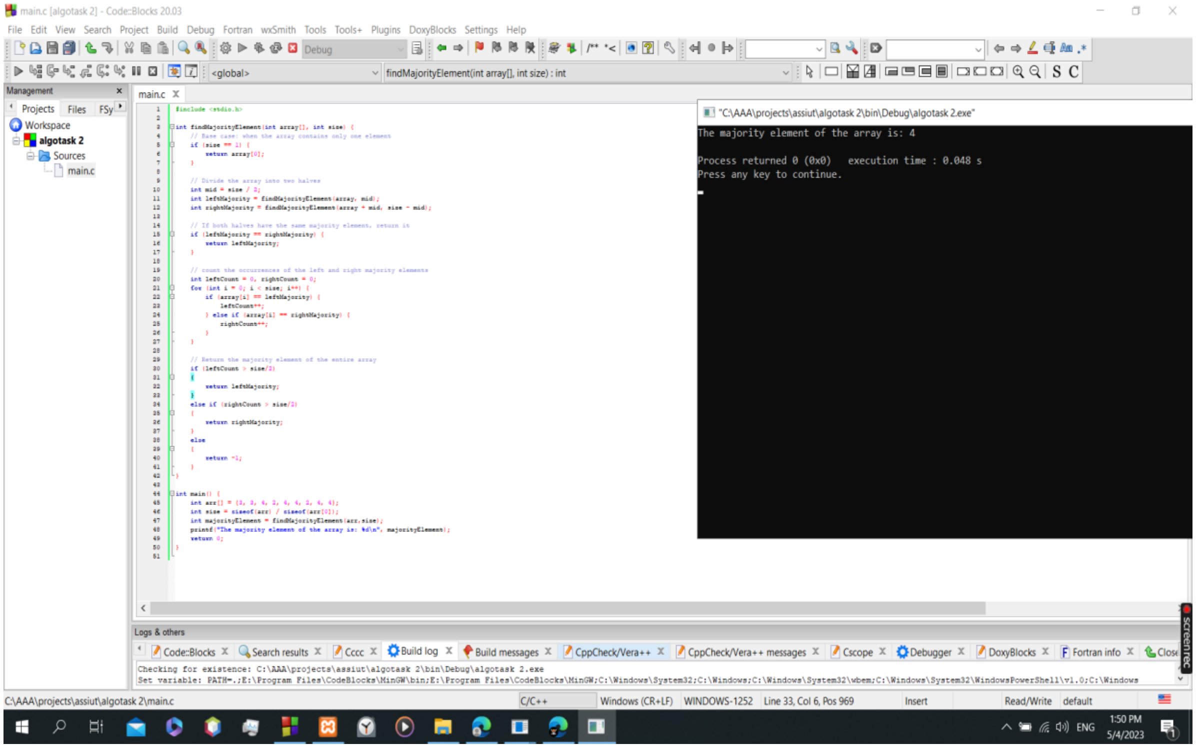
10. `else { return -1; }` - Time complexity is O(1).

the time complexity of the `findMajorityElement` function:

is O(n log n) in the worst case, where n is the size of the array.

is O(n) in the best case.

Output



main.c [algotask 2] - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Debug
<global>
findMajorityElement(int array[], int size) : int
Management
Projects Files FSy
Workspace
algotask 2
Sources
main.c

```c
#include <stdio.h>

int findMajorityElement(int array[], int size) {
    // Base case: when the array contains only one element
    if (size == 1) {
        return array[0];
    }

    // Divide the array into two halves
    int mid = size / 2;
    int leftMajority = findMajorityElement(array, mid);
    int rightMajority = findMajorityElement(array + mid, size - mid);

    // If both halves have the same majority element, return it
    if (leftMajority == rightMajority) {
        return leftMajority;
    }

    // count the occurrences of the left and right majority elements
    int leftCount = 0, rightCount = 0;
    for (int i = 0; i < size; i++) {
        if (array[i] == leftMajority) {
            leftCount++;
        } else if (array[i] == rightMajority) {
            rightCount++;
        }
    }

    // Return the majority element of the entire array
    if (leftCount > size/2)
    {
        return leftMajority;
    }
    else if (rightCount > size/2)
    {
        return rightMajority;
    }
    else
    {
        return -1;
    }
}

int main() {
    int arr[] = {3, 3, 4, 2, 4, 4, 2, 4, 4};
    int size = sizeof(arr) / sizeof(arr[0]);
    int majorityElement = findMajorityElement(arr,size);
    printf("The majority element of the array is: %d\n", majorityElement);
    return 0;
}
```

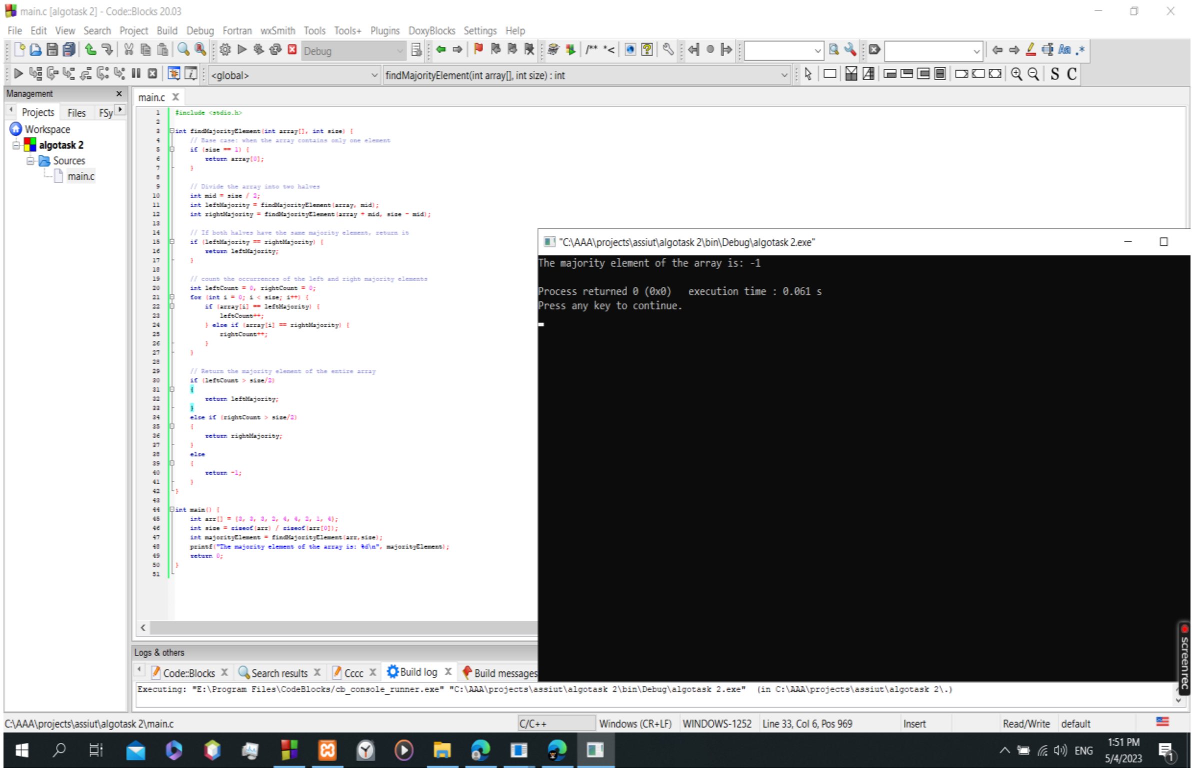
"C:\AAA\projects\assiut\algotask 2\bin\Debug\algotask 2.exe"

```
The majority element of the array is: 4

Process returned 0 (0x0)   execution time : 0.048 s
Press any key to continue.
```

Logs & others
Code::Blocks Search results Cccc Build log Build messages CppCheck/Vera++ CppCheck/Vera++ messages Cscope Debugger DoxyBlocks Fortran info Close

```
Checking for existence: C:\AAA\projects\assiut\algotask 2\bin\Debug\algotask 2.exe
Set variable: PATH=.;E:\Program Files\CodeBlocks\MinGW\bin;E:\Program Files\CodeBlocks\MinGW;C:\Windows\System32;C:\Windows;C:\Windows\System32\wbem;C:\Windows\System32\WindowsPowerShell\v1.0;C:\Windows
```

C:\AAA\projects\assiut\algotask 2\main.c C/C++ Windows (CR+LF) WINDOWS-1252 Line 33, Col 6, Pos 969 Insert Read/Write default
ENG 1:50 PM 5/4/2023
screenrec



main.c [algotask 2] - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Debug
<global>
findMajorityElement(int array[], int size) : int
Management
Projects Files FSy
Workspace
algotask 2
Sources
main.c

```c
#include <stdio.h>

int findMajorityElement(int array[], int size) {
    // Base case: when the array contains only one element
    if (size == 1) {
        return array[0];
    }

    // Divide the array into two halves
    int mid = size / 2;
    int leftMajority = findMajorityElement(array, mid);
    int rightMajority = findMajorityElement(array + mid, size - mid);

    // If both halves have the same majority element, return it
    if (leftMajority == rightMajority) {
        return leftMajority;
    }

    // count the occurrences of the left and right majority elements
    int leftCount = 0, rightCount = 0;
    for (int i = 0; i < size; i++) {
        if (array[i] == leftMajority) {
            leftCount++;
        } else if (array[i] == rightMajority) {
            rightCount++;
        }
    }

    // Return the majority element of the entire array
    if (leftCount > size/2)
    {
        return leftMajority;
    }
    else if (rightCount > size/2)
    {
        return rightMajority;
    }
    else
    {
        return -1;
    }
}

int main() {
    int arr[] = {3, 3, 3, 2, 4, 4, 2, 1, 4};
    int size = sizeof(arr) / sizeof(arr[0]);
    int majorityElement = findMajorityElement(arr,size);
    printf("The majority element of the array is: %d\n", majorityElement);
    return 0;
}
```

"C:\AAA\projects\assiut\algotask 2\bin\Debug\algotask 2.exe"

```
The majority element of the array is: -1

Process returned 0 (0x0)   execution time : 0.061 s
Press any key to continue.
```

```
Executing: "E:\Program Files\CodeBlocks/cb_console_runner.exe" "C:\AAA\projects\assiut\algotask 2\bin\Debug\algotask 2.exe"  (in C:\AAA\projects\assiut\algotask 2\.)
```

C:\AAA\projects\assiut\algotask 2\main.c C/C++ Windows (CR+LF) WINDOWS-1252 Line 33, Col 6, Pos 969 Insert Read/Write default
ENG 1:51 PM 5/4/2023
screenrec

Comparison

Here's a comparison table between the two codes:

	Non recursive algorithm	**Recursive algorithm**
Time Complexity	O(n)	O(n log n)
Best Case	O(n)	O(1)
Worst Case	O(n^2)	O(n log n)
Average Case	O(n^2)	O(n log n)
Space Complexity	O(n)	O(log n)
Strengths	Works well for uniformly distributed data	Works well for arbitrary data
Weaknesses	can be a problem for very large input sizes.	May take longer on average

Overall, recursive algorithm is better in terms of performance but requires more space due to the recursive calls. Non recursive algorithm is simpler and easier to read but has poor performance.