Project (10): Diagonal Difference

	Name	id
1	محمد شريف جلال محمد	20210787
2	محمد هشام حامد مصطفى	20210844
3	محمد مصطفى محمود عبدالمجيد	20210837
4	محمد وليد محمد مختار	20210847
5	محمد علي احمد علي	20210818
6	مينا ريمون نوفير فهيم	20210985

The first algorithm (non-recursive)

Pseudocode:

Read n from user.

Create arr with size n*n.

For i= 0 to n-1

For j=0 to n-1

Read arr[i][j] from user.

LeftSum=0

Rightsum=0

For i=0 to n-1

LeftSum = leftSum + arr[i][i]

Rightsum=RightSum+ arr[i][n-i-1]

Diff= absolute value of LeftSum-RightSum

Prints Diff

Time complexity t(n):

- Read the size of the square matrix from the user. This takes constant time and has a time complexity of O(1).

- Create a 2D array (arr) of size n x n. This takes O(n^2) time, since it needs to allocate memory for n x n elements.

- Read the elements of the matrix from the user and store them in the arr array. This takes O(n^2) time since it needs to iterate over each element of the array.

- checks if each element of the matrix is within the range [-100, 100]. This check takes constant time per element, so the overall time complexity of this takes O(n^2).

- Initialize the variables leftSumDiagonal and rightSumDiagonal to 0. This takes constant time and has a time complexity of O(1).

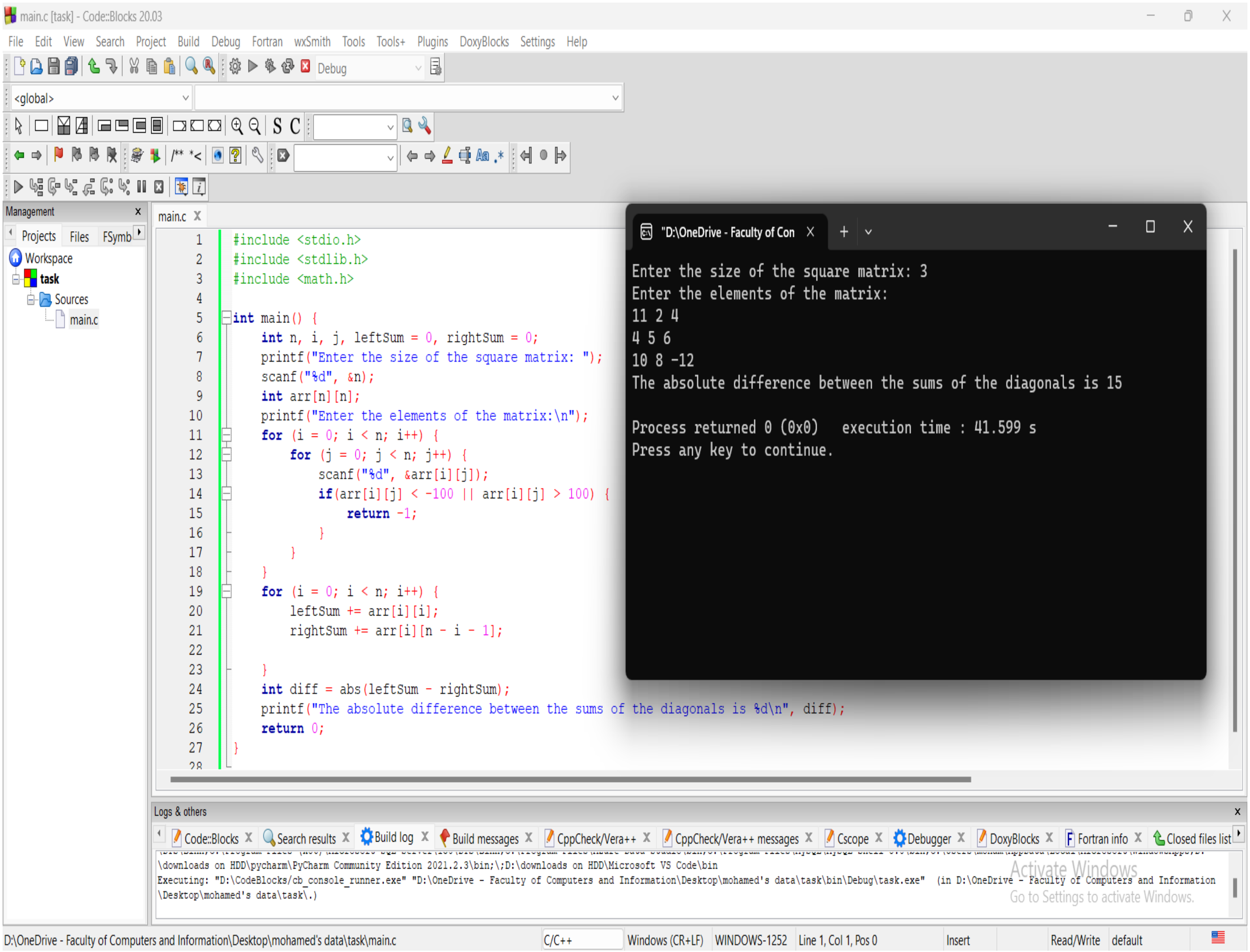
- Iterate over the rows and columns of the matrix to calculate the sum of the left diagonal and the right diagonal. This takes O(n) time since it needs to iterate over each row and column of the matrix once.

- Calculate the absolute difference between the two sums using the absolute function. This takes constant time and has a time complexity of O(1).

- Print the result. This takes constant time and has a time complexity of O(1).

- Therefore, the overall time complexity of the code can be expressed as:
  $T(n) = O(1) + O(n^2) + O(n^2) + O(1) + O(n) + O(1) + O(1)$
  $= O(n^2)$

- **The code**


main.c [task] - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Management
Projects Files FSymb
Workspace
task
Sources
main.c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int main() {
int n, i, j, leftSum = 0, rightSum = 0;
printf("Enter the size of the square matrix: ");
scanf("%d", &n);
int arr[n][n];
printf("Enter the elements of the matrix:\n");
for (i = 0; i < n; i++) {
for (j = 0; j < n; j++) {
scanf("%d", &arr[i][j]);
if(arr[i][j] < -100 || arr[i][j] > 100) {
return -1;
}
}
}
for (i = 0; i < n; i++) {
leftSum += arr[i][i];
rightSum += arr[i][n - i - 1];
}
int diff = abs(leftSum - rightSum);
printf("The absolute difference between the sums of the diagonals is %d\n", diff);
return 0;
}
"D:\OneDrive - Faculty of Con
Enter the size of the square matrix: 3
Enter the elements of the matrix:
11 2 4
4 5 6
10 8 -12
The absolute difference between the sums of the diagonals is 15
Process returned 0 (0x0) execution time : 41.599 s
Press any key to continue.
Logs & others
Code::Blocks Search results Build log Build messages CppCheck/Vera++ CppCheck/Vera++ messages Cscope Debugger DoxyBlocks Fortran info Closed files list
Executing: "D:\CodeBlocks/cb_console_runner.exe" "D:\OneDrive - Faculty of Computers and Information\Desktop\mohamed's data\task\bin\Debug\task.exe" (in D:\OneDrive - Faculty of Computers and Information\Desktop\mohamed's data\task\.)
D:\OneDrive - Faculty of Computers and Information\Desktop\mohamed's data\task\main.c
C/C++ Windows (CR+LF) WINDOWS-1252 Line 1, Col 1, Pos 0 Insert Read/Write default

The second algorithm (recursive)

Pseudocode:

```
Define function diagonalSumDiff(matrix, n, i, leftsumdiagonal, rightsumdiagonal)
   if n = i
   returns abs(leftsumdiagonal - rightsumdiagonal)
   end if
   leftsumdiagonal += matrix[i][i]
   rightsumdiagonal += matrix[i][n - i - 1]
   i++
   return diagonalSumDiff(matrix, n, i, leftsumdiagonal, rightsumdiagonal)
end function
 main function
user enters size of matrix n*n.
For i=0 to n
For j=0 to n
Read arr[i][j] from user.
If matrix[i][j] >=100 or matrix[i][j]<=-100
Prints Invalid matrix element.
Returns -1
intialize diff = diagonalSumDiff(matrix, n, 0, 0, 0);
prints diff
```

Time complexity t(n):

The time complexity of the diagonalSumDiff function in this algorithm is O(n), where n is the size of the matrix, because each recursive call of the function processes one element of each diagonal. The function makes a recursive call n times, once for each element in the main diagonal and the opposite diagonal, and performs a constant amount of work during each call.

The time complexity of the main function is O(n^2), where n is the size of the matrix, because it reads in n^2 elements from the input and checks their validity.

Therefore, the overall time complexity of the algorithm is O(n^2).

The steps:

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

T(n) = T(n-3) + n-2 + n-1 + n

.

.

T(n) = T(n-k) + kn - k(k-1)/2 ...(1)

For base case:

n - k = 1 so we can get T(1)

=> k = n - 1

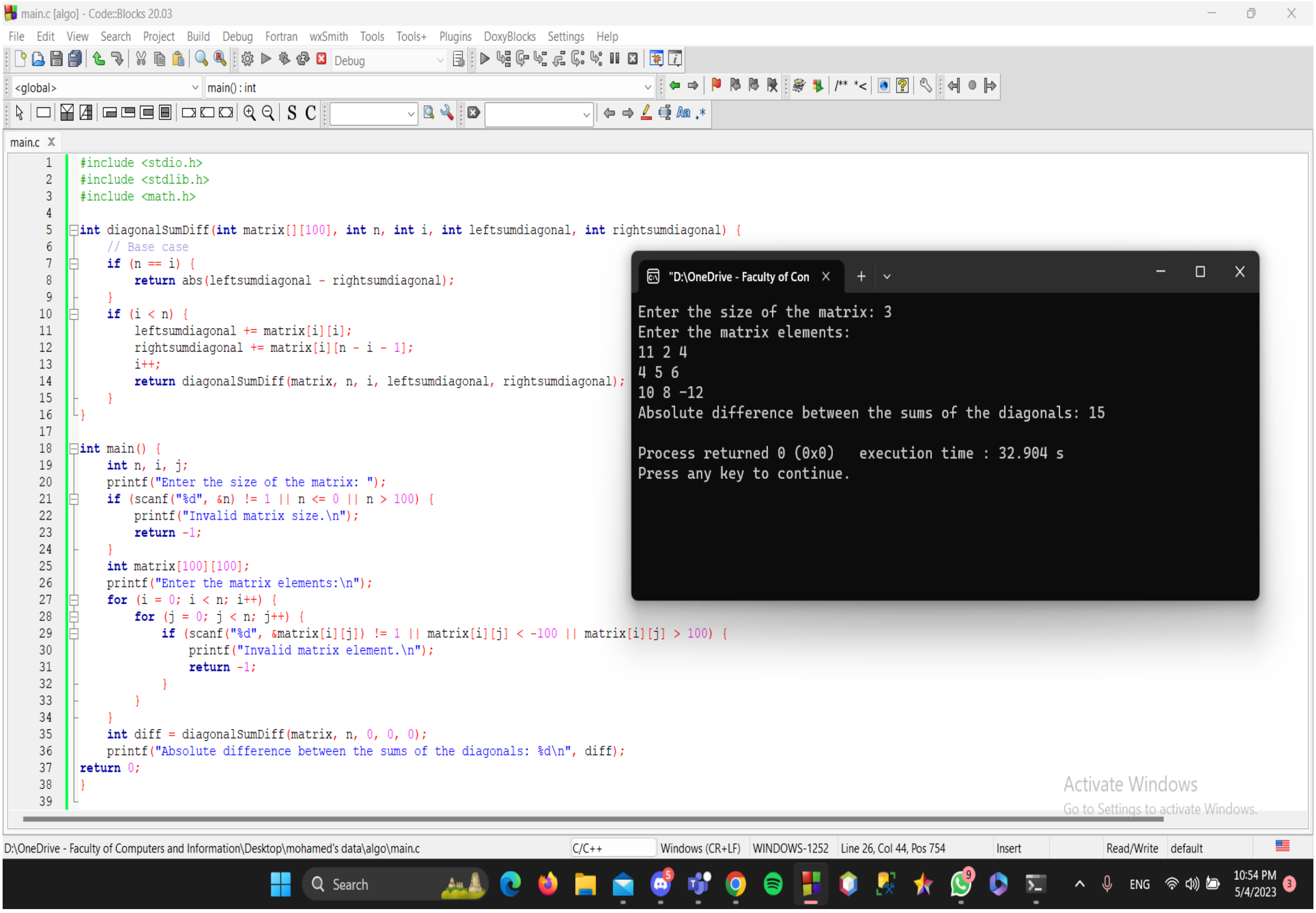
substitute in (1)

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

T(n) = T(1) + (n^2)

So T(n)=o(n^2)

- ## The code



```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int diagonalSumDiff(int matrix[][100], int n, int i, int leftsumdiagonal, int rightsumdiagonal) {
    // Base case
    if (n == i) {
        return abs(leftsumdiagonal - rightsumdiagonal);
    }
    if (i < n) {
        leftsumdiagonal += matrix[i][i];
        rightsumdiagonal += matrix[i][n - i - 1];
        i++;
        return diagonalSumDiff(matrix, n, i, leftsumdiagonal, rightsumdiagonal);
    }
}

int main() {
    int n, i, j;
    printf("Enter the size of the matrix: ");
    if (scanf("%d", &n) != 1 || n <= 0 || n > 100) {
        printf("Invalid matrix size.\n");
        return -1;
    }
    int matrix[100][100];
    printf("Enter the matrix elements:\n");
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            if (scanf("%d", &matrix[i][j]) != 1 || matrix[i][j] < -100 || matrix[i][j] > 100) {
                printf("Invalid matrix element.\n");
                return -1;
            }
        }
    }
    int diff = diagonalSumDiff(matrix, n, 0, 0, 0);
    printf("Absolute difference between the sums of the diagonals: %d\n", diff);
return 0;
}
```

```
Enter the size of the matrix: 3
Enter the matrix elements:
11 2 4
4 5 6
10 8 -12
Absolute difference between the sums of the diagonals: 15

Process returned 0 (0x0)   execution time : 32.904 s
Press any key to continue.
```

The comparison between the two algorithms

comparison	The first algorithm(non-recursive)	The second algorithm(recursive)
The best case	O(1)	O(1)
	when the input matrix has a size of 1, which is also the smallest possible size for a square matrix.	when the input matrix has a size of 1, which is also the base case of the diagonalSumDiff function. In this case, the function will simply return 0 without performing any calculations.
The avg case	O(n^2)	O(n^2)
The worst case	O(n^2)	O(n^2)
	when the input matrix is a large square matrix of size n x n, where n is a very large number	when the input matrix is a large square matrix where the size of the matrix is a multiple of 2.
The most preferred algorithm	**both**	