

ESTRUCTURA DE DATOS 1
Código ST0245

Laboratory practice No. 2: NOTATION BIG O

Alejandro Fernández Restrepo

Universidad Eafit
Medellín, Colombia
afernander@eafit.edu.co

Nayibe Tatiana Vélez David

Universidad Eafit
Medellín, Colombia
ntvelezd@eafit.edu.co

Adelaida Maldonado Esguerra

Universidad Eafit
Medellín, Colombia
amaldonade@eafit.edu.co

3) Practice for final project defense presentation

3.1 Time measurement for 20 different sizes taking into account that the measurement will be made in milliseconds; This will be done with two types of algorithms: INSERTION SORT AND MERGE SORT.

INSERTION SORT

Size	Time
5000	23
10000	18
15000	28
20000	46
25000	111
30000	146
35000	218
40000	284
45000	353
50000	338
55000	254
60000	307
65000	349
70000	394
75000	452
80000	493
85000	560
90000	653
95000	714
100000	785

MERGE SORT

SIZE	TIME
1000000	132
2000000	263
3000000	265
4000000	324
5000000	334
6000000	398
7000000	477
8000000	526
9000000	588
10000000	689
11000000	782
12000000	826
13000000	916
14000000	980
15000000	967
16000000	1042
17000000	1141
18000000	1277
19000000	1267
20000000	1355

PhD. Mauricio Toro Bermúdez

Professor | School of Engineering | Informatics and Systems

Email: mtorobe@eafit.edu.co | Office: Building 19 – 627

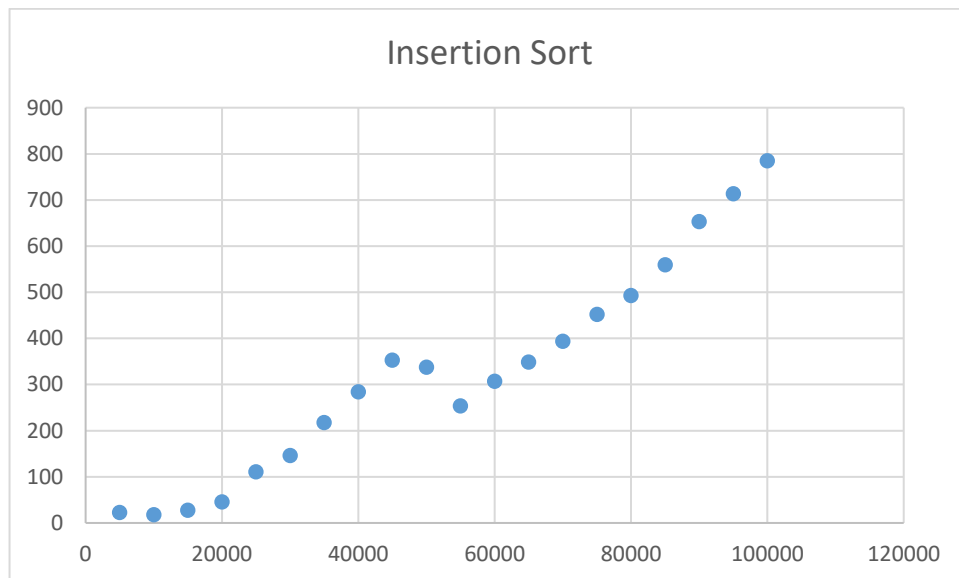
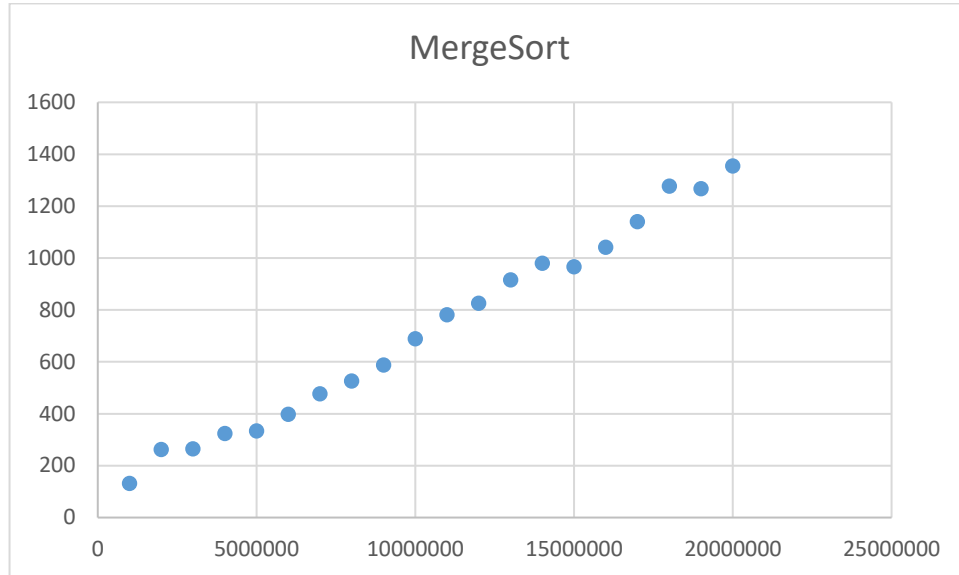
Phone: (+57) (4) 261 95 00 Ext. 9473



ESTRUCTURA DE DATOS 1

Código ST0245

3.2 Graphs where we can define the size and execution time



PhD. Mauricio Toro Bermúdez

Professor | School of Engineering | Informatics and Systems

Email: mtorobe@eafit.edu.co | Office: Building 19 – 627

Phone: (+57) (4) 261 95 00 Ext. 9473

3.3 Based on the case of large arrays, it is more efficient to use the mergesort because it separates the elements of the array into several parts that allow optimal and more efficient use than the insertionsort, as this visits each position of arrays, having a slower runtime, which makes it possible to use mergesort for large arrays.

3.4 Using insertionsort in a video game would not be optimal, since it takes a long time to execute, which leads to occupy a large space of memory, and a late execution for the game.

3.5 To ensure that the use of the insertion sort in large arrays has efficiency, it is required that the array elements be ordered, to allow efficient use of it.

3.6 Explanation of the algorithm MaxSpan:

MaxSpan is an algorithm which, given an array of integers, the algorithm searches which is the largest space that exists in the array between two equal numbers and will return the number of elements between these two numbers included.

Functionality:

For the algorithm to search for this space, we first have two variables in our case, which will be a temporary counter which will have the value of the space between the numbers that will be evaluated at the moment and that will be the final value of the largest space, which will come back. The algorithm works through two cycles for nests, the first one will go from 0 to the length of the array with a variable i and the second one will go from the length of the array to 0 with the variable j , this will automatically evaluate all possible combinations of spaces between the arrangement from the first to the last position. Then we have a yes which will verify if the number of the i is equal to that of the j , if it is the case that this condition is met our variable s (temporary container) will be equal to the subtraction of the positions plus 1, which it is the size between the two equal numbers, and then it will be another if it is verified if the evaluated vector is larger than the previous vector, if s is larger am , if it is the case m will be equal to that it is the new largest size found.

In the end the algorithm returns the value of m which is the largest size between two equal numbers in the array.

PhD. Mauricio Toro Bermúdez

Professor | School of Engineering | Informatics and Systems

Email: mtorobe@eafit.edu.co | Office: Building 19 – 627

Phone: (+57) (4) 261 95 00 Ext. 9473

3.7) Complexity Big O Array2 and Array3

ARRAY2

•

```

public int countEvens(int[] nums) {

    int cont = 0;

    for(int i = 0; i<nums.length; i++){

        if(nums[i]%2==0)
            cont++;
    }

    return cont;
}

```

The notation Big O is= O(n)

•

```

public int sum13(int[] nums) {

    int a = 0;

    for(int i = 0; i<nums.length; i++){

        if(nums[i]==13 && i<nums.length-1){

            nums[i+1]=0;
        }

        if(nums[i]!=13){

            a+=nums[i];
        }
    }

    return a;
}

```

PhD. Mauricio Toro Bermúdez

Professor | School of Engineering | Informatics and Systems

Email: mtorobe@eafit.edu.co | Office: Building 19 – 627

Phone: (+57) (4) 261 95 00 Ext. 9473

The notation Big O is= $O(n)$

-

```
public int bigDiff(int[] nums) {
    int a = nums[0];
    int b = nums[0];

    for(int i = 0; i<nums.length; i++){
        a = Math.max(a, nums[i]);
        b = Math.min(b, nums[i]);
    }
    return a-b;
}
```

The notation Big O is= $O(n)$

-

```
public boolean sum28(int[] nums) {
    int x =0;

    for(int i= 0; i < nums.length; i++){
        if(nums[i]==2){
            x+=2;
        }
    }

    return x == 8;
}
```

The notation Big O is= $O(n)$

PhD. Mauricio Toro Bermúdez

Professor | School of Engineering | Informatics and Systems

Email: mtorobe@eafit.edu.co | Office: Building 19 – 627

Phone: (+57) (4) 261 95 00 Ext. 9473

ESTRUCTURA DE DATOS 1

Código ST0245

•

```
public boolean has12(int[] nums) {

    boolean h1 = false;
    boolean h2 = false;

    for(int i = 0; i < nums.length; i++){
        if(nums[i]==1){
            h1 = true;
        }
        if(nums[i]==2 && h1){
            h2 = true;
        }
    }

    return h2;
}
```

The notation Big O is= $O(n)$

ARRAY3

•

```
public int maxSpan(int[] nums) {

    int m = 0;
    int s = 0;

    for(int i = 0; i<nums.length; i++){
        for(int j = nums.length-1; j>=0; j--){

            if(nums[i]==nums[j])

                s = j-i+1;

            if (s>m) m =s;

        }

    }

    return m;
}
```

The notation Big O is= $O(n^2)$

PhD. Mauricio Toro Bermúdez

Professor | School of Engineering | Informatics and Systems

Email: mtorobe@eafit.edu.co | Office: Building 19 – 627

Phone: (+57) (4) 261 95 00 Ext. 9473

ESTRUCTURA DE DATOS 1

Código ST0245

```

public boolean linearIn(int[] outer, int[] inner) {

    int a = 0;

    for(int i =0; i<inner.length; i++){
        for(int j =0; j<outer.length; j++){

            if(i<inner.length){
                if(inner[i]==outer[j]){

                    a++;
                    i++;
                }
            }
        }
    }
    return a==inner.length;
}

```

*The notation Big O is= $O(i*o)$*

```

public int[] squareUp(int n) {

    int [] a = new int [n*n];

    int b = 0;

    for(int i =1; i<=n; i++){
        for(int j = n; j>=1; j--){

            a[b++]=j;

            if(i<j)
                a[b-1] = 0;

        }
    }

    return a;
}

```

The notation Big O is= $O(m^2)$

PhD. Mauricio Toro Bermúdez

Professor | School of Engineering | Informatics and Systems

Email: mtorobe@eafit.edu.co | Office: Building 19 – 627

Phone: (+57) (4) 261 95 00 Ext. 9473

ESTRUCTURA DE DATOS 1

Código ST0245

```

public int[] seriesUp(int n) {
    int [] a = new int [n*(n + 1)/2];
    int b = 0;
    for(int i = 1; i<=n; i++){
        for(int j = 1; j<=i; j++){
            a[b++]=j;
        }
    }
    return a;
}

```

The notation Big O is= $O(m^2)$

```

public int countClumps(int[] nums) {
    int a = 0;
    int b = -1;
    for(int i = 0; i < nums.length - 1; i++) {
        if(nums[i] == nums[i + 1] && nums[i] != b) {
            b = nums[i];
            a++;
        }
    }

    for(int j = 0; j < nums.length - 1; j++) {
        if(nums[j] != b) {
            b = -1;
        }
    }
    return a;
}

```

The notation Big O is= $O(n^2)$

PhD. Mauricio Toro Bermúdez

Professor | School of Engineering | Informatics and Systems

Email: mtorobe@eafit.edu.co | Office: Building 19 – 627

Phone: (+57) (4) 261 95 00 Ext. 9473

3.8) The variables that are considered in the notation Big O, they directly influence the algorithms; its meaning is:

n = Array length

m = Variable for the array creation

o = Outer length

i = Inner length

4) Practice for midterms

4.1 C

4.2 B

4.3 B

4.4 A

4.5 D

4.6 A

4.7 Q

4.7.1 $T(n) = T(n - 1) + c$

4.7.2 $O(n)$

4.8 A

4.9 D

4.10 C

4.11 C

4.12 B

4.13 C

4.14 A

PhD. Mauricio Toro Bermúdez

Professor | School of Engineering | Informatics and Systems

Email: mtorobe@eafit.edu.co | Office: Building 19 – 627

Phone: (+57) (4) 261 95 00 Ext. 9473