# Laboratory practice No. 2: Big O Notation

**Santiago Isaza Cadavid**
Universidad EAFIT
Medellín, Colombia
sisazac@eafit.edu.co

**Hamilton Smith Gómez Osorio**
Universidad EAFIT
Medellín, Colombia
hsgomezo@eafit.edu.co

August 31, 2018

## 1) *GitHub's codes*

## 2) *Project Questions Simulation*

### 2.a. *Algorithms's chart*

### 2.b. *Algorithms's graphics*

### 2.c. *Given the above information, how efficient is merge sort compared with insertion sort for large arrays? Is it appropriate to use insertion sort for a data base with millions of elements?*

### 2.d. *Explain with your own words how does the Codingbat's Array3 exercise maxSpan works. Why?*

### 2.e. *Calculate the complexity of the on-line exercise*

**i.**

```
public int countEvens(int[] nums) {
 int n=0;
  for(int i=0;i<nums.length;i++){
      if(nums[i]%2==0) n+=1;
  }
  return n;
}
```

**ii.**

```
public boolean lucky13(int[] nums) {
   for(int i=0;i<nums.length;i++){
       if(nums[i]==3 || nums[i]==1) return false;
   }
   return true;
}
```

**iii.**

```
    public boolean isEverywhere(int[] nums, int val) {
     for(int i=0;i<nums.length-1;i++){
         if(nums[i]!=val && nums[i+1]!=val) return false;
     }
     return true;
}
```

**iv.**

```
     public boolean modThree(int[] nums) {
     for(int i=0;i<nums.length-2;i++){
         if(nums[i]%2==0 && nums[i+1]%2==0 && nums[i+2]%2==0) return true;
         if(nums[i]%2==1 && nums[i+1]%2==1 && nums[i+2]%2==1) return true;
     }
     return false;
}
```

**v.**

```
     public boolean tripleUp(int[] nums) {
     for(int i=0;i<nums.length-2;i++){
         if(nums[i+1]==nums[i]+1 && nums[i+2]==nums[i]+2) return true;
     }
     return false;
}
```

### *2.f. Explain what the variable n means in the previous exercises*

## *3) Midterm Simulation*

### *3.a. Exercise 1*

c) O(n+m)

### *3.b. Exercise 2*

a) O($m * n$)

### *3.c. Exercise 3*

b) O(ancho)

### *3.d. Exercise 4*

b) O($n^3$)

### *3.e. Exercise 5*

d) O($n^2$)

## *3.f. Exercise 6*

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

## *3.g. Exercise 7*

### 3.7.1 Worst case-scenario number of steps

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

### 3.7.2 Asymptotic Complexity

O(n)

## *3.h. Exercise 8*

The mystery(n) function executes $n * \sqrt{n}$ steps

## *3.i. Exercise 9*

d) Executes more than $n^2 + n * m$

## *3.j. Exercise 10*

a) Executes less than $n * \log n$ steps

## *3.k. Exercise 11*

c) Executes T(n) = T(n-1)+T(n-2)+C steps

## *3.l. Exercise 12*

b) O($m\sqrt{n}$)

## *3.m. Exercise 13*

a) O($n^3$)

## *4) Recommended reading*