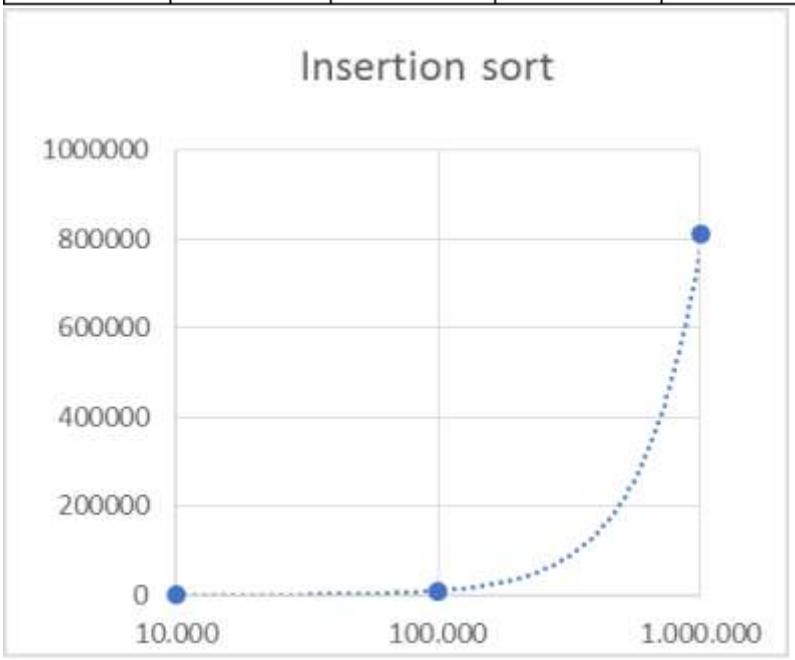
Laboratorio Nro. 2: Notación O grande

Mateo Montes Loaiza
Universidad Eafit
Medellín, Colombia
Mmontesl1@eafit.edu.co

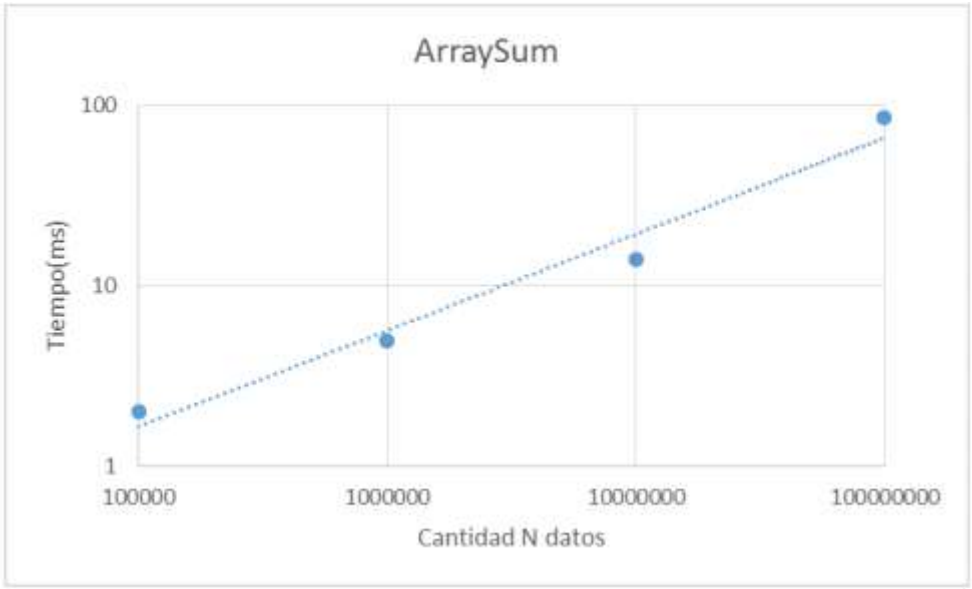
Denilson Moreno Cardona
Universidad Eafit
Medellín, Colombia
dmorenoc@eafit.edu.co@eafit.edu.co

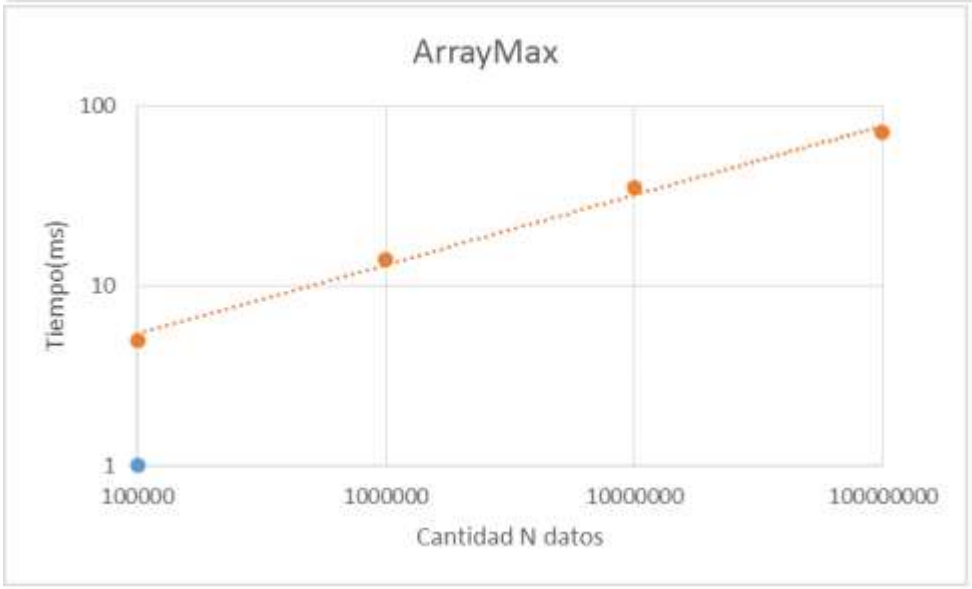
3.5)

N	100000	1000000	10000000	100000000
ArraySum	2	5	14	85
ArrayMax	5	6	16	71
Insertion Sort	4119	Mas de 5 min	Mas de 5 min	Mas de 5 min
Merge Sort	44	240	2537	23762

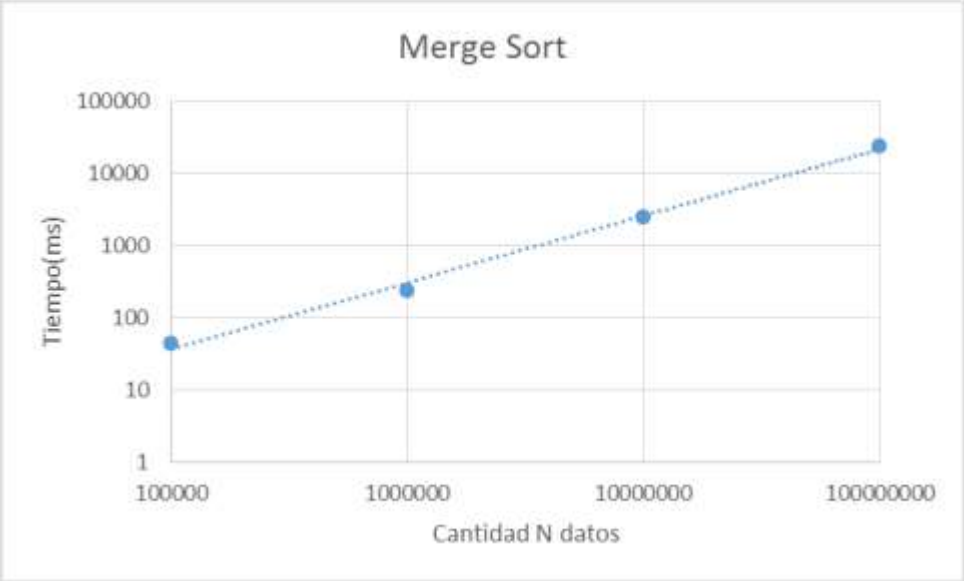


DOCENTE MAURICIO TORO BERMÚDEZ
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627
Correo: mtorobe@eafit.edu.co





DOCENTE MAURICIO TORO BERMÚDEZ
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627
Correo: mtorobe@eafit.edu.co



3.6)

Se concluye que para cada uno de los metodos se da un resultado el cual es O(n) y la grafica concuerda ya que es una linea recta.
En la grafica Insertion Sort ya que tiene tiempos de ejecucion muy largos, su complejidad es de O(n^2)

3.7)

En Insertion sort es un algoritmo el cual tarda mucho en ejecutar ya que en el peor de los casos analiza cada uno de los valores. Si se pasa un arreglo muy grande su ejecucion crece ya que su complejidad es O(n^2)

3.8)

El tiempo de ejecución es bajo ya que solo realiza un solo ciclo siendo su complejidad O(n). esto demuestra porque en el método insertion sort, como tiene complejidad O(n^2) tiene un tiempo más alto que ArraySum

3.9)

Merge sort es más eficiente comparandose con insertion sort ya que su complejidad es de O(nlogn) y esto se da para pocos datos como para muchos.

3.11)

```
1. public int[] evenOdd(int[] nums) {
      int[] ret = new int[nums.length]; //C1
       int index = 0; // C2
            for (int i = 0; i < nums.length; i++) { // C3*n
                  if (nums[i] % 2 == 0) { // C4*n
                  ret[index] = nums[i]; // C5*n
                   index++; // C6*n
                   }
            }
           return ret; //C7
 }
```

T(n) = C1 + C2 + C7 + 2*C3*n + 2*C4*n + 2*C5*n

T(n) = C1 + C2 + 2n * [C3 + C4 + C5]
T(n) =C + 2n* C6
T(n) es O(C + 2n*C6)
T(n) es O(2n*C6)
T(n) es O(2n)
T(n) es O(n)

```
2. public String[] fizzBuzz(int start, int end) {
        int n = end - start; // C1
         String[] ret = new String[n]; //C2
         for (int i = 0; i < n; i++) { C3*n
               if (start % 3 == 0 && start % 5 == 0) { //C4*n
               ret[i] = "FizzBuzz";
         } else if (start % 3 == 0) { //C5*n
        ret[i] = "Fizz";
        } else if (start % 5 == 0) { //C6*n
         ret[i] = "Buzz";
 } else {
        ret[i] = String.valueOf(start); //C7*n
 }
        start++; //C8*n
 }
 return ret;// C9
 }
```

T(n) = C1 + C2 + C9 + C3*n + C4*n + C5*n + C6*n + C7*n +C8*n
T(n) =C + n[C3 + C4 + C5 + C6 +C7 + C8]
T(n) =C + C' *n
T(n) es O(C + n*C')
T(n) es O(n*C')
T(n) es O(n)

```
3. public int[] withoutTen(int[] nums) {
         int index = 0; //C1
        int[] ret = new int[nums.length];// C2
         for (int i = 0; i < nums.length; i++) { C3*n
               if (nums[i] != 10) { C4*n
                ret[index] = nums[i]; C5 *n
                index++; C6 *n
                }
        }
        return ret; //C7
 }
```

T(n) = C1 + C2 + C7 + C3*n + C4*n +C5*n + C6* n
T(n) =C + n[C3 + C4 + C5 + C6]
T(n) =C + C' *n
T(n) es O(C + n*C')
T(n) es O(n*C')
T(n) es O(n)

```
4. public int[] zeroFront(int[] nums) {
        int cero = 0; // C1
        int uno = nums.length - 1;// C2
        int[] ret = new int[nums.length];// C3
```

```
for (int i = 0; i < nums.length; i++) { //C4*n
      if (nums[i] == 0) { //C5*n
      ret[cero] = nums[i]; //C6*n
      cero++;//C7*n
} else {
       ret[uno] = nums[i];
      uno--;
}
}
return ret; // C8
}
```

T(n) = C1 + C2 + C8 +C3 + C4*n +C5*n + C6* n + C7* n
T(n) =C + n[C4 + C5 + C6 + C7]
T(n) =C + C'*n
T(n) es O(C + n*C')
T(n) es O(n*C')
T(n) es O(n)

```
5. public int[] notAlone(int[] nums, int val) {
       for (int i = 1; i < nums.length - 1; i++) { //C1*n
             if (nums[i] == val) { //C2*n
             int mayor = 0; //C3*n
       if (nums[i - 1] != val && nums[i + 1] != val) {//C4*n
             mayor = nums[i - 1] > nums[i + 1] ? nums[i - 1] : nums[i + 1]; //C5*n
       if (mayor > nums[i]) {//C6*n
            nums[i] = mayor; //C7*n
       }
       }
       }
       }
return nums; // C8
}
```

T(n) = C1*n + C2*n + C8 +C3*n + C4*n +C5*n + C6* n + C7* n
T(n) =C8 + n[C1 + C2 + C3 + C4 + C5 + C6 + C7]
T(n) =C8 + C'*n
T(n) es O(C8 + n*C')
T(n) es O(n*C')
T(n) es O(n)

```
6. public int maxSpan(int[] nums) {
      int span = 0; // C1
      for (int i = 0; i < nums.length; i++) { // C2 *n
      for (int j = nums.length - 1; j >= 0; j--) { // C3 * n^2
            If (nums[i] == nums[j] && j - i + 1 > span) { // C4 * n^2
                  span = j - i + 1; *
      break;
      }
}
return span; C5
}
```

T(n) = C1 + C5 + C2*n +C3* n^2+ C4* n^2
T(n) =C + n^2 [C3 + C4] + C2*n

DOCENTE MAURICIO TORO BERMÚDEZ
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627
Correo: mtorobe@eafit.edu.co

T(n) =C + C'* n^2 + C2*n
T(n) es O(C + n^2*C'+ C2*n)
T(n) es O(n^2*C')
T(n) es O(n^2)

```
7 public int[] fix34(int[] nums) {
 for (int i = 0; i < nums.length; i++) { C1*n
       if (nums[i] == 3 && nums[i + 1] != 4) {C2*n
       for (int j = nums.length - 1; j > 0; j--) { C3*n*m
             if (nums[j] == 4 && nums[j - 1] != 3) { C4* n*m
             nums[j] = nums[i + 1
             nums[i + 1] = 4;
 break;
 }
}
 }
 }
 return nums; C5
 }
```

T(n) = C1*n + C5 + C2*n +C3*n*m + C4*n*m
T(n) =C5 + n*m[C3 + C4] + n[C2 +C1]
T(n) =C5 + C'*n*m + C''*n
T(n) es O(C5 + n*C'*m+ C''*n)
T(n) es O(n*m*C')
T(n) es O(n*m)

```
8. public int[] fix45(int[] nums) {
       for (int i = 0; i < nums.length; i++) { C1*n
             if (nums[i] == 4 && nums[i + 1] != 5) { C2*n
       for (int j = nums.length - 1; j >= 0; j--) { C3* n^2
             if (nums[j] == 5) { C4 * n^2
             if (j == 0 || nums[j - 1] != 4) {
             nums[j] = nums[i + 1];
             nums[i + 1] = 5;
 break;
 }}
 }
 }
 }
 }
 return nums; C5
 }
```

T(n) = C1*n + C5 + C2*n +C3* n^2+ C4* n^2
T(n) =C5 + n^2 [C3 + C4] + n[C2 +C1]
T(n) =C5 + C'* n^2+ C''*n
T(n) es O(C5 + n^2*C'+ C''*n)
T(n) es O(n^2*C')
T(n) es O(n^2)

```
9. public boolean canBalance(int[] nums) {
       for (int i = 0; i < nums.length; i++) { C1*n
       int sum = 0; C2*n
       for (int j = 0; j < nums.length; j++) { C3*n^2
```

```
            sum = i < j ? sum + nums[j] : sum - nums[j];C4*n^2
        }
            if (sum == 0) {C5*n
 return true;
 }
 }
 return false; C6
 }
```

T(n) = C1*n + C6 + C2*n +C3*n^2 + C4*n^2 +C5*n +C6
T(n) =C6 + n^2*[C3 + C4] + n[C2 +C1+ C5]
T(n) =C6 + C'*n^2 + C''*n
T(n) es O(C5 + n^2*C + C''*n)
T(n) es O(n^2*C')
T(n) es O(n^2)

```
10. public boolean linearIn(int[] outer, int[] inner) {
        int count = 0; // C1
        for (int i = 0; i < inner.length; i++) { C2*n
        for (int j = 0; j < outer.length; j++) {C3*n*m
        if (outer[j] == inner[i]) { C4*n*m
        count++;
        break;
 }
 }
 }
 return count == inner.length; C5
 }
```

T(n) = C1 + C5 + C2*n +C3*n*m + C4*n*m
T(n) =C5 + n*m[C3 + C4] + n*C2
T(n) =C5 + C'*n*m + C2*n
T(n) es O(C5 + n*C'*m+ C2*n)
T(n) es O(n*m*C')
T(n) es O(n*m)

3.12)

n sirve para saber la longitude del arreglo y recorrerlo
m representa las veces que se hace un ciclo dentro de otro ciclo.

4)

1) c
2) d
3) b
4) b
5) d
6) a
7.1) T(n)= C2+T(n-1)
7.2) O(O(n))

DOCENTE MAURICIO TORO BERMÚDEZ
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627
Correo: mtorobe@eafit.edu.co