Laboratorio Nro. 2 : Notación O grande

Luis Javier Palacio Mesa
Universidad Eafit
Medellín, Colombia
ljpalaciom@eafit.edu.co

Santiago Castrillon Galvis
Universidad Eafit
Medellín, Colombia
scastrillg@eafit.edu.co

Kevyn Santiago Gómez Patiño
Universidad Eafit
Medellín, Colombia
ksgomezp@eafit.edu.co

3) Simulacro de preguntas de sustentación de Proyectos

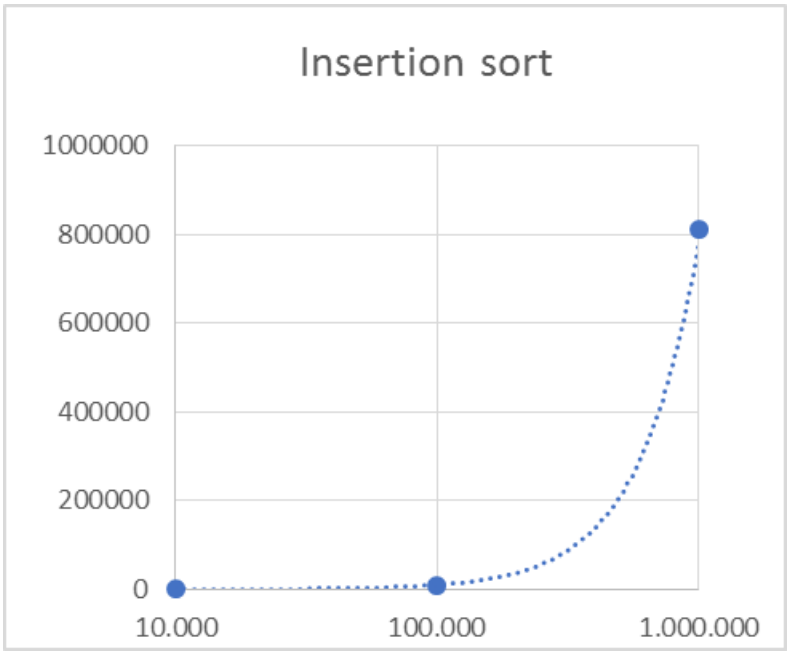
DOCENTE MAURICIO TORO BERMÚDEZ
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627
Correo: mtorobe@eafit.edu.co

3.1

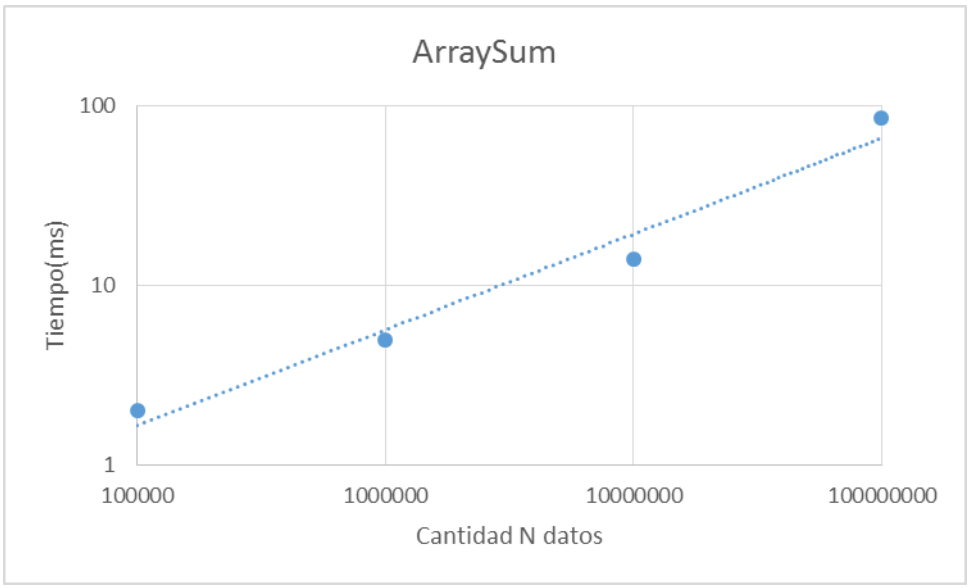
N	100000	1000000	10000000	100000000
ArraySum	2	5	14	85
ArrayMax	5	6	16	71
Insertion Sort	4119	Mas de 5 min	Mas de 5 min	Mas de 5 min
Merge Sort	44	240	2537	23762

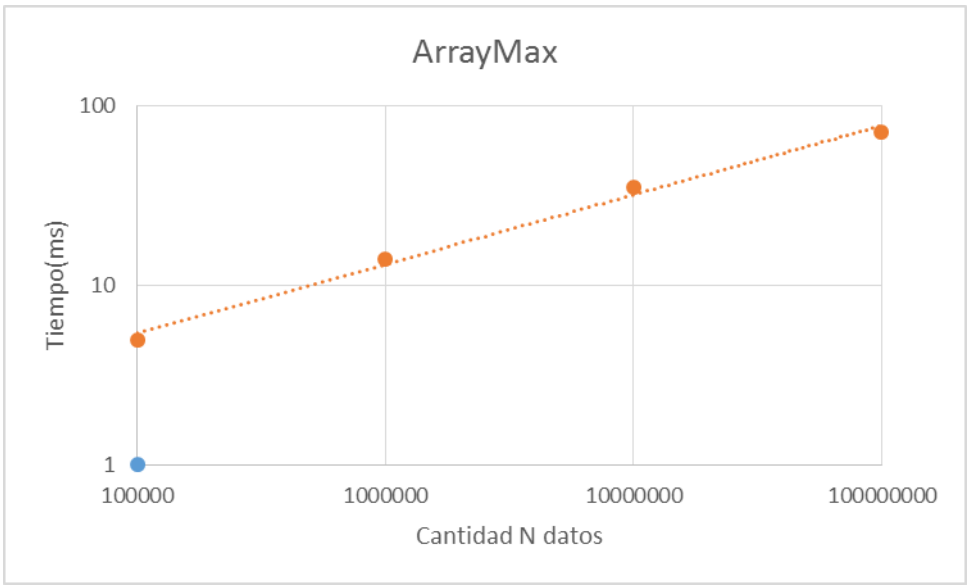
Insertion sort	10.000	134
Insertion sort	100.000	8993
Insertion sort	1.000.000	812844

3.2

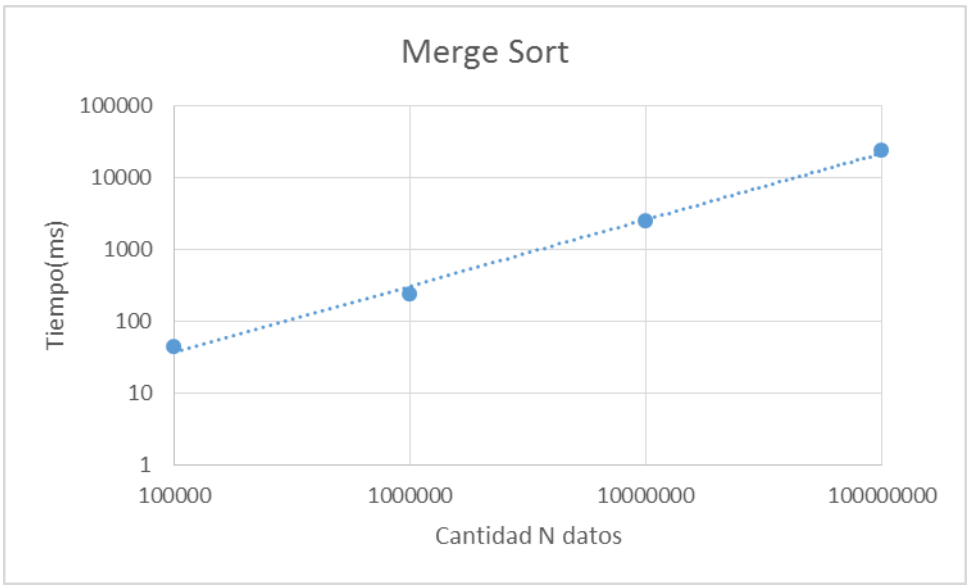


DOCENTE MAURICIO TORO BERMÚDEZ
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627
Correo: mtorobe@eafit.edu.co





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



3.3

Para cada algoritmo se obtuvo los resultados esperados ya que en la grafica se representan con una linea recta, es decir, complejidad O(n). Sin embargo, no fue posible comprobar insertion sort debido a que sus tiempos de ejecucion después de n=1000000 requerían mucho tiempo en ejecución. Esto es así puesto que la complejidad de tal algoritmo es O(n^2)

3.4

Insertion sort se demora mucho porque es un algoritmo de ordenamiento que, en el peor de los casos, analiza todos los valores del arreglo. Al pasarle un arreglo muy grande su tiempo de ejecución crece mucho debido a su complejidad O(n^2).

3.5

El tiempo de ejecución sigue siendo relativamente bajo. Esto se debe a que sólo se realiza un ciclo para el método, siendo su complejidad O(n). Esto explica por qué insertion sort, un algoritmo de complejidad O(n^2), tiene un tiempo de ejecución más alto que arraySum.

3.6

Merge sort es muchísimo más eficiente que insertion sort debido a que es un algoritmo de búsqueda binaria por lo que su complejidad es O(nlogn), esto aplica tanto para pocos datos como para muchos, aunque con pocos datos la diferencia no es tan notoria

3.7

El algoritmo se basa en un ciclo dentro de otro y un entero “span” que nos servirá para guardar la mayor distancia encontrada. Este segundo ciclo hace el recorrido de derecha a izquierda puesto lo que queremos es obtener la máxima distancia entre dos valores iguales en el arreglo. Así pues, en este segundo ciclo tenemos un condicional que verifica que los dos números que saquemos del arreglo sean iguales, y que la distancia (que obtenemos al restarle al índice j, del segundo ciclo, el índice i, del primer ciclo, y sumar 1 para compensar el hecho de que el arreglo guarda valores desde 0) sea una distancia mayor a la que tengamos en la variable span; si la condición se cumple se acualiza la variable span a esta nueva distancia que es mayor y nos salimos de este segundo ciclo gracias al break, puesto que ya encontramos la mayor distancia entre esos dos valores.

3.8

Calculen la complejidad de los ejercicios en línea, numerales 2.1 y 2.2, y agréguenla al informe PDF

Sea n = nums.length;

```
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
            }
        }
        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 $O(n)$

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

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 $O(n*C')$

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

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--;
   } // Da lo mismo si entra a la condicion de arriba o de abajo
}
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*

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

*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;
} // Suponemos que nunca entra a la condicion puesto que si entra el ciclo para antes de llegar a m }
}
return span; C5
}
*T(n) = C1 + C5 + C2*n +C3* n^2+ C4* n^2*
*T(n) =C + n^2 [C3 + C4] + C2*n*
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;
} // Suponemos que nunca entra a la ultima condición puesto que si entra el ciclo para antes de llegar a m
}
}
}
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*

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

*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;
}} // Suponemos que nunca entra a la ultima condición puesto que si entra el ciclo para antes de llegar a m
}
}
}
}
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; // Suponemos que nunca entra a la condición puesto que si entra el ciclo para antes de llegar a n
}
}
return false; C6
}

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

*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;
          } // Suponemos que nunca entra a la condición puesto que si entra el ciclo para antes de llegar a m
        }
      }
      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)*

3.9 Expliquen con sus palabras las variables (qué es 'n', qué es 'm', etc.) del cálculo de complejidad del numeral anterior.

1. n o nums.length: Este variable nos sirve para saber cual es la longitud de un arreglo y recorrerlo
2. m: En algunos problemas, en el segundo ciclo no había que recorrer totalmente el arreglo, así que me representa las veces que se hace un ciclo dentro de otro.

4) Simulacro de Parcial

1. *c*
2. *d*
3. *b*
4. *b*
5. *d*

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