Laboratorio Nro. 2: notación O grande

Eduard Damiam Londoño Garcia
Universidad Eafit
Medellín, Colombia
edlondonog@eafit.edu.co

Esteban Osorio
Universidad Eafit
Medellín, Colombia
eosoriof@eafit.edu.co

3) Simulacro de preguntas de sustentación de Proyectos

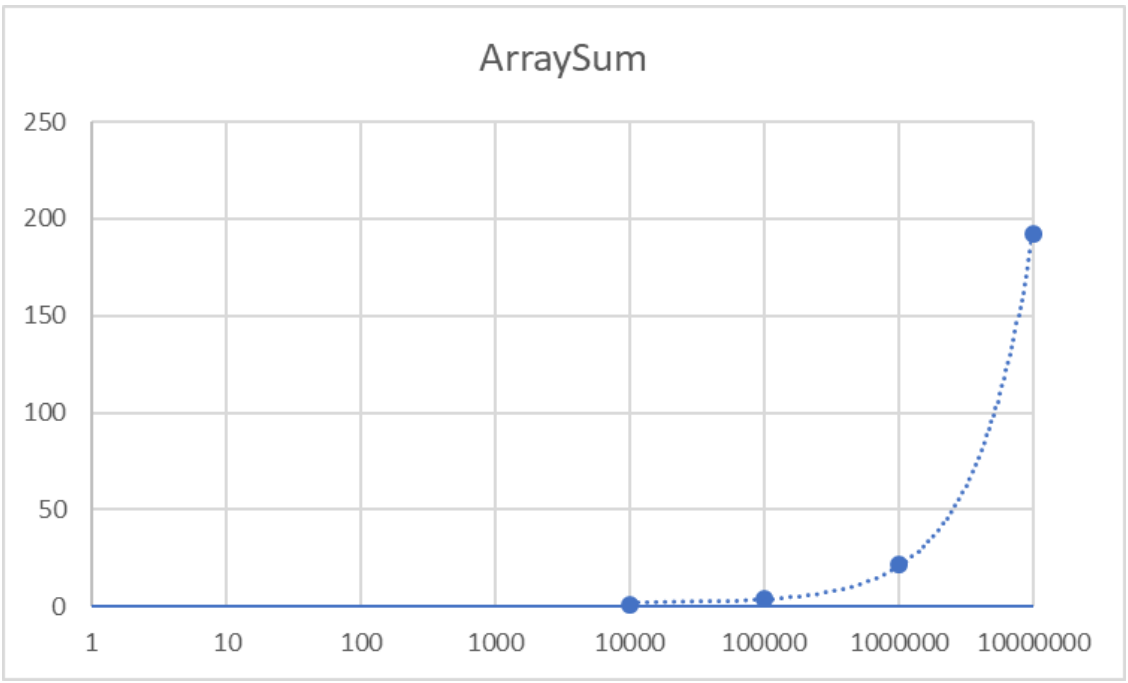
1.

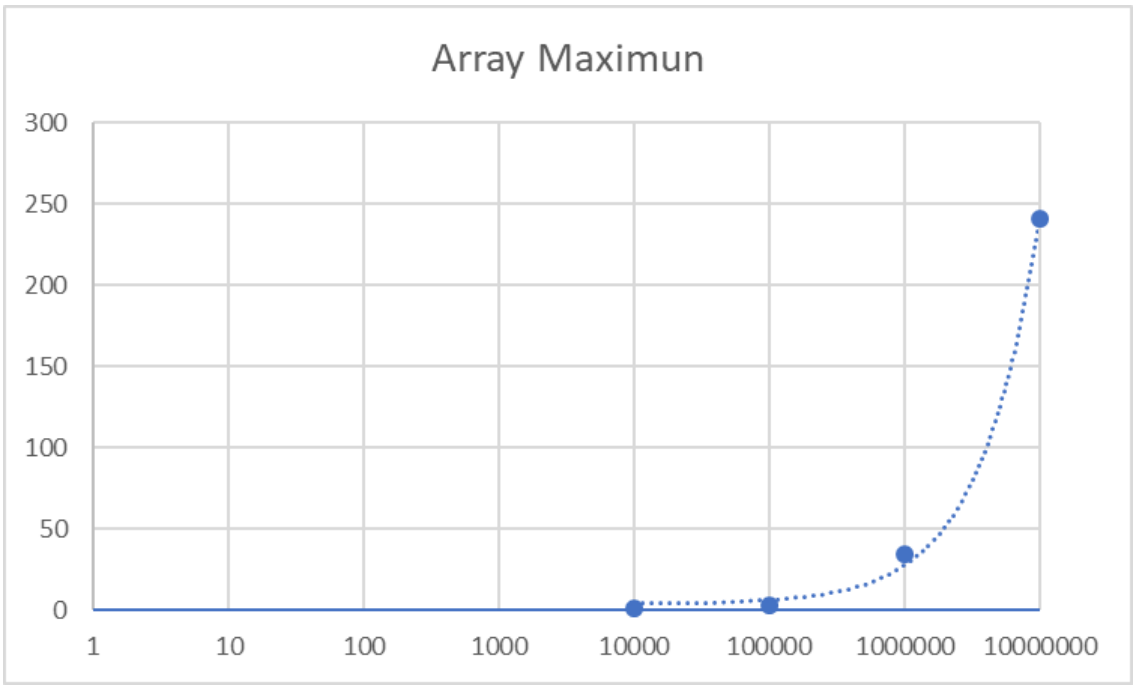
	N = 10.000	N= 100.000	N= 1'.000.000	N= 10'000.000
Array Sum	**1**	**4**	**22**	**192**
Array Maximun	**1**	**3**	**35**	**241**

	N = 1.000	N = 10.000	N = 100.000	N = 1'000.000
Insertion sort	**4**	**134**	**5337**	**375702**

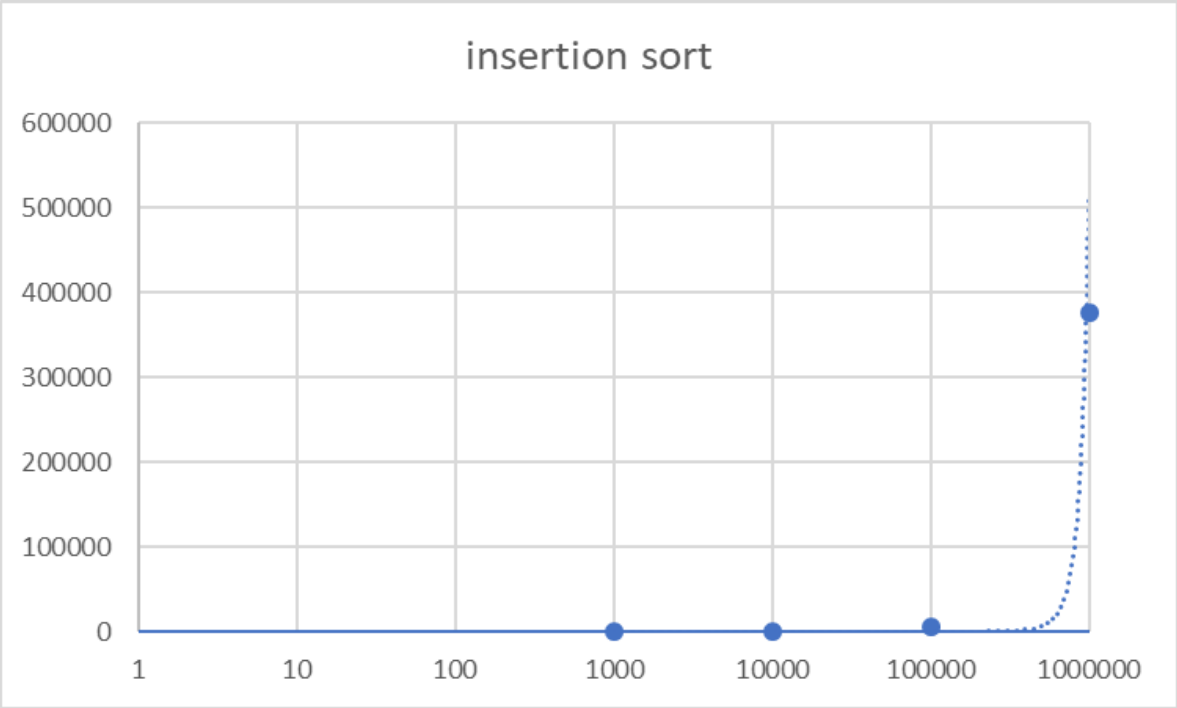
	N= 100.000	N= 1'000.000	N= 10'000.000	N= 100'000.000
Merge sort	**27**	**179**	**1440**	**15343**

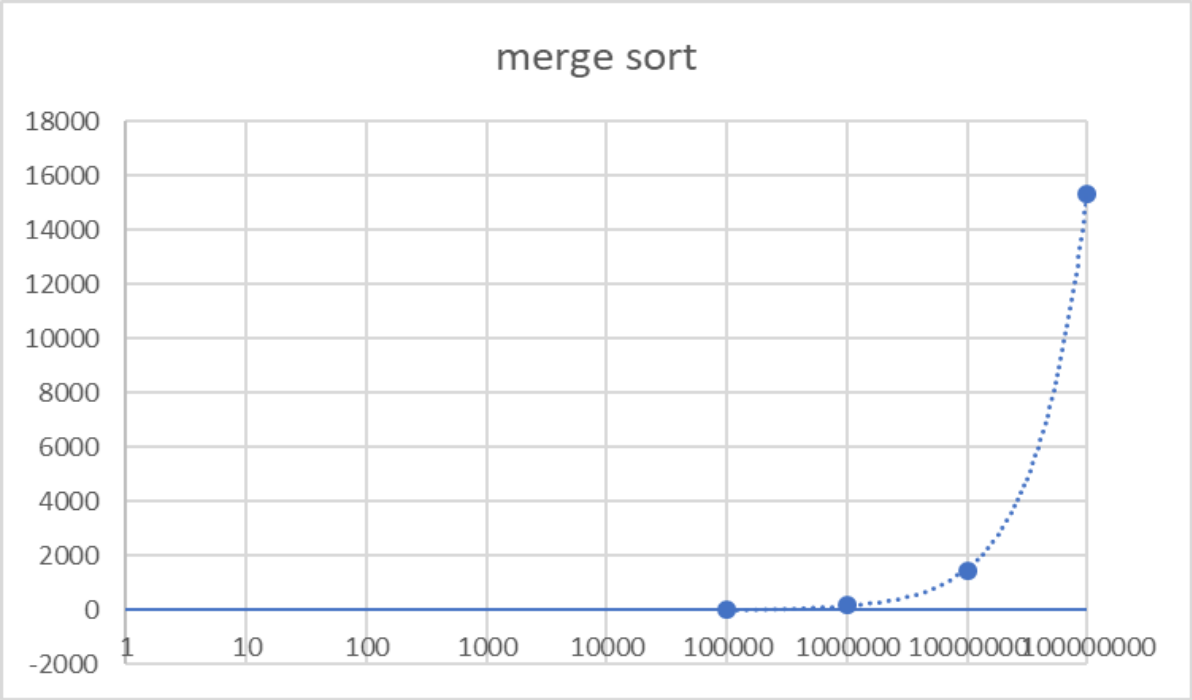
2.





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. Tanto array sum y array Maximun tenían el mismo resultado teórico el cual fue O(n) y las gráficas corroboraron el resultado teórico ya que las dos dieron una línea recta además a pesar de que array Maximun daba unos valores un poco más grandes, la gráfica para valores grandes era casi la misma, las gráficas también corroboraron los resultados teóricos de Insertion sort el cual dio $O(n^2)$ y el Merge sort que dio O(n log n).

4. Insertion sort para valores grandes es muy poco eficiente, por ejemplo, cuando lo intentamos para 10'000.000 duro más 3 min una gran cantidad de tiempo, además de un gasto bastante grande de memoria por parte del PC mientras realiza el ordenamiento.

5. Para array sum el tiempo no crece tan rápido como con Insertion sort, la razón de esto es que Insertion sort tiene complejidad $O(n^2)$ porque tiene dos ciclos, mientras que array sum es solo O(n) porque tiene un solo llamado recursivo.

6. Merge sort es mucho más eficiente para valores grandes, cuando probamos Insertion sort con 10'000.000 pasaron 5 min y seguía sin terminar la ejecución, mientras que cuando probamos 100'000.000 con Merge sort tardo solo 15343 ms una diferencia abismal de tiempo, en cuanto a valores pequeños Merge sort también es más eficiente que Insertion sort, para 10.000 Insertion sort tardo 134 ms mientras que Merge sort tardo solo 4.

7. Primero se mira que el tamaño del arreglo sea mayor a 0, si es cero se retorna 0, en caso de ser mayor a 0 se inicia la variable para contar el Span en 1, se empieza un ciclo desde 0 hasta el tamaño del arreglo, dentro de este ciclo hay otro ciclo que empieza desde el tamaño del arreglo -1 hasta i, si la posición i y la posición j son iguales se crea una variable igual j-i + 1, el +1 es porque el span es inclusivo, si la variable es mayor al Span actual se cambia el Span por la variable y se rompe el ciclo, al final se retorna el Span.

8.
```
public int countEvens(int[] nums) {
int cont = 0; //C1
for(int i = 0; i < nums.length;i++){ //C2*n
  if(nums[i] % 2 == 0)    //C3
    cont++;   //C4
}
return cont; //C5
}
```

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

T(n)= C1+C3+C4+C5+C2*n
T(n) es O(C1+C3+C4+C5+C2*n)
T(n) es O(C2*n)
T(n) es O(n)

```
public int bigDiff(int[] nums) {
        int max = nums[0];    //C1
        int min = nums[0];    //C2
        for(int i = 1; i < nums.length;i++){  //C3*n
        min = Math.min(min,nums[i]);    //C4
        max = Math.max(max,nums[i]);    //C5
       }
      return max-min;   //C6
}
```

T(n)= C1+C2+C4+C5+C3*n
T(n) es O(C1+C2+C4+C5+C3*n)
T(n) es O(C3*n)
T(n) es O(n)

```
public int centeredAverage(int[] nums) {
    int max = nums[0];    //C1
    int min = nums[0];    //C2
    int indMax = 0;       //C3
    int indMin = 0;       //C4
    for(int i = 1; i < nums.length;i++){  //  C5*n
        If (Math.min(min,nums[i]) < min){   //C6*n
          min = Math.min(min,nums[i]);    //C7*n
          indMin = i;                     //C8*n
        }
       if(Math.max(max,nums[i]) > max){    //C9*n
          max = Math.max(max,nums[i]);    //C10*n
          indMax = i;                     //C11*n
       }
     }
     nums[indMin] = 0;         //C12
  nums[indMax] = 0;         //C13
  int cont = 0;             //C14
  int acom = 0;             //C15
  for(int i = 0; i < nums.length;i++){   //C16*n
```

```
        if(nums[i] != 0){                  //C17*n
            cont++;                        //C18*n
           acom = acom + nums[i];              //C19*n
        }
     }

    return acom/cont;                    //C20
   }
```

T(n)= C1+C2 +C3+ C4+ C12+ C13+C14+C15+C20+(C5+C6+C7+C8+C9+C10+C11 +C16+17+C18+C19)n
T(n) es O(C1+C2+C3+C4+C12+C13+C14+C15+C20+(C5+C6+C7+C8+C9+C10+ C11+C16+17+C18+C19)n)
T(n) es O((C5+C6+C7+C8+C9+C10+C11+C16+17+C18+C19)n)
T(n) es O(n)

```
      public int sum13(int[] nums) {
      int cont = 0;                        //C1
      for(int i = 0; i < nums.length;i++){  //C2*n
        if(nums[i] != 13){                 //C3*n
        cont = cont +  nums[i];            //C4*n
        }
        else if(nums[i]==13 && i < nums.length - 1){ //C5*n
          nums[i] = 0;                              //C6*n
          nums[i+1]=0;                               //C7*n
        }
      }
      return cont;                 //C8
     }
```

T(n) = C1+C8(C2+C3+C4+C5+C6+C7)n
T(n) es O(C1+C8(C2+C3+C4+C5+C6+C7)n)
T(n) es O((C2+C3+C4+C5+C6+C7)n)
T(n) es O(n)

```
    public boolean has22(int[] nums) {
       If(nums.length <= 1){     //C1
         return false;         //C2
       }
      for(int i = 0;i < nums.length-1;i++){    //C3*n
```

```
            if(nums[i]==2 && nums[i+1]==2){        //C4*n
              return true;                         //C5*n
            }
          }
          return false;                            //C6
      }
```

T(n) = C1+C2+C6+(C3+C4+C5)n
T(n) es O(C1+C2+C6+(C3+C4+C5)n)
T(n) es O(C3+C4+C5)n)
T(n) es O(n)

```
    public int maxSpan(int[] nums) {
        if (nums.length > 0) { //C1
            int maxSpan = 1;  //C2
            for (int i = 0; i < nums.length; i++){  C3*n
                for (int j = nums.length - 1; j > i; j--){  C4*n^2
                    if (nums[j] == nums[i]) { C5
                        int count = (j - i) + 1; C6
                        if (count > maxSpan) maxSpan = count; C7
                        break; C8
                        }
                    }
                }
                return maxSpan; //C9
            } else return 0; //C10
        }
```

T(n)= C+n+n^2
T(n) es O(C+n+n^2)
T(n) es O(n+n^2)
T(n) es O(n^2)

```
     public int[] fix34(int[] nums) {
        for (int i = 0; i < nums.length; i++) //C1+n
            if (nums[i] == 3) { //C2
                int cont = nums[i + 1]; //C3
                nums[i + 1] = 4; //C4
                for (int j = i + 2; j < nums.length; j++) //C5+n^2
                    if (nums[j] == 4) nums[j] = cont; //C6
            }
        return nums; //C7
```

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

```
    }
```

$T(n)= C+n+ n^2$
$T(n)$ es $O(C+n+n^2)$
$T(n)$ es $O(n+n^2)$
$T(n)$ es $O(n^2)$

```
    public boolean canBalance(int[] nums) {
        for (int i = 0; i < nums.length; i++) { //C1*n
            int sum = 0; //C2
            for (int j = 0; j < i; j++){ //C4*m*n
                sum += nums[j]; //C5
            }
            for (int j = i; j < nums.length; j++){ //C5* n^2
                sum -= nums[j]; //C6
            }
            if (sum == 0) return true; //C7
        }
        return false; //C8
    }
```

$T(n)= C+n+(m*n)+n^2)$
$T(n)$ es $O(C+n+(m*n)+n^2)$
$T(n)$ es $O(n+(m*n)+n^2)$
$T(n)$ es $O((m*n)+ n^2)$
$T(n)$ es $O(n^2)$

```
  public int[] fix45(int[] nums) {
        for (int i=0;i<nums.length;i++) //C1+n
            if (nums[i] == 5 && i == 0 || nums[i] == 5 && nums[i - 1] != 4) { //C2
                int a = i; //C3
                for (int j = 0; j < nums.length; j++) //C4*n^2
                    if (nums[j] == 4 && nums[j + 1] != 5) { //C5
                        int temp = nums[j + 1]; //C6
                        nums[j + 1] = 5; //C7
                        nums[a] = temp; //C8
                        break; //C9
                    }
            }
        return nums; //C10
    }
```

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

$T(n) = C+n+ n^2$
$T(n)$ es $O(C+n+ n^2)$
$T(n)$ es $O(n+ n^2)$
$T(n)$ es $O(n^2)$

```
public boolean linearIn(int[] outer, int[] inner) {
        int a = 0; C1
        int b = 0; C2
        while (a < inner.length && b  < outer.length) { C3*n
            if (outer[b] == inner[a]) { C4
                b++; C5
                a++; C6
            } else b++; C7
        }
        return (a == inner.length); C8
    }
```

$T(n) = C+n$
$T(n)$ es $O(C+n)$
$T(n)$ es $O(n)$

9. Las variables como n, m etc. en los cálculos de complejidad anteriores representa una variable que puede afectar el número de instrucciones que ejecuta el programa, ósea es una variable que mientras más grande sea más lento se ejecutara el programa, en todos los programas anteriores, n era el tamaño del arreglo.

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