Laboratorio Nro. 2: Notación O grande

Manuela Valencia Toro
Universidad Eafit
Medellín, Colombia
mvalenciat@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
ArraySum	**1001**	**1003**	**1004**	**1032**

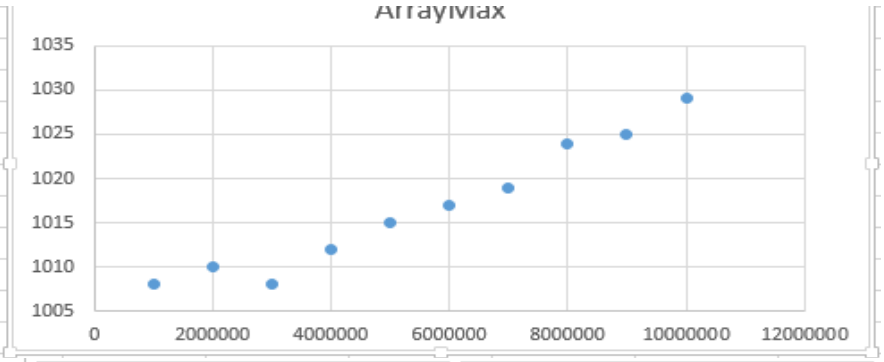
	10.000	1.000.000	10.000.000	20.000.000
ArrayMax	**1001**	**1002**	**1028**	**1047**

	10.000	100.000	1.000.000	10.000.00
InsertionSort	**147**	**9863**	**Más de 5 min**	**Se llena la memoria**

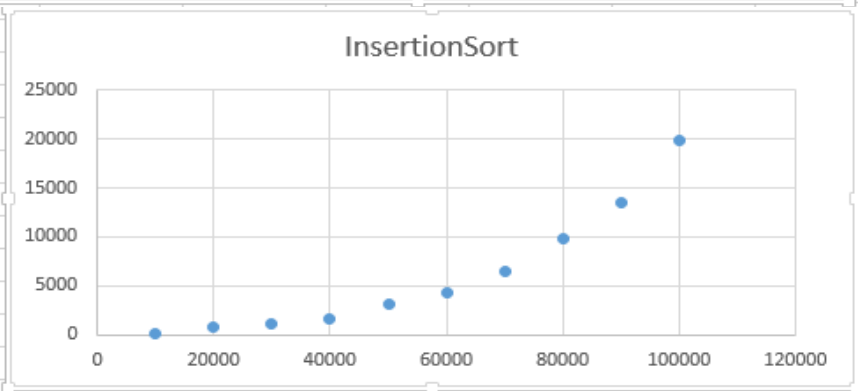
	10.000	100.000	1.000.000	10.000.000
MergeSort	**8**	**63**	**219**	**2107**

2.

000000	1008
000000	1010
000000	1008
000000	1012
000000	1015
000000	1017
000000	1019
000000	1024
000000	1025
000000	1029

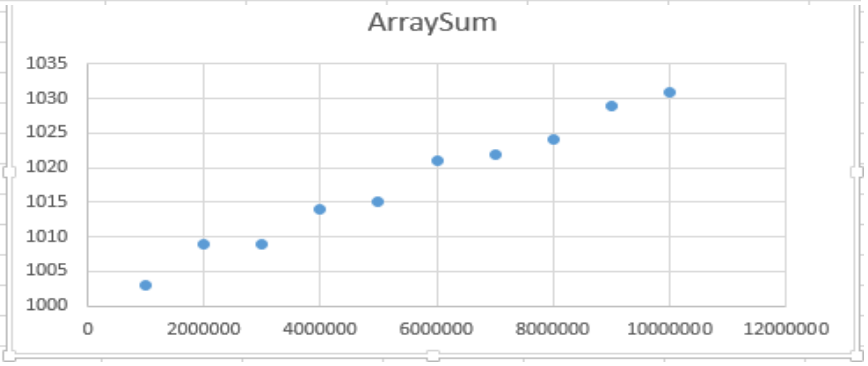


10000	140
20000	707
30000	1037
40000	1563
50000	3057
60000	4341
70000	6384
80000	9747
90000	13576
100000	19822

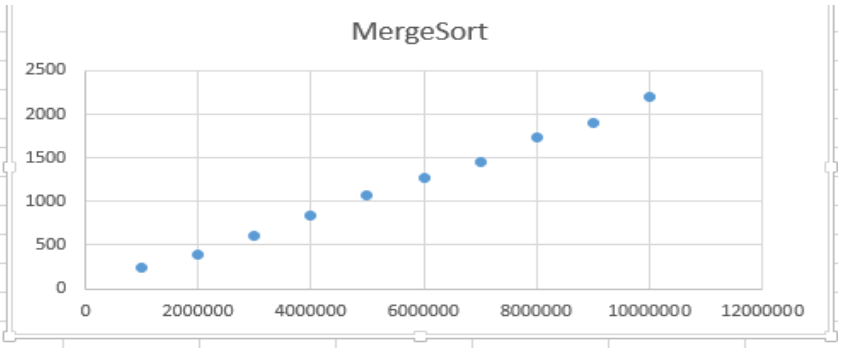


1000000

1000000	1003
2000000	1009
3000000	1009
4000000	1014
5000000	1015
6000000	1021
7000000	1022
8000000	1024
9000000	1029
10000000	1031



1000000	236
2000000	398
3000000	602
4000000	835
5000000	1071
6000000	1265
7000000	1452
8000000	1736
9000000	1897
10000000	2199



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

3. Según la complejidad y las gráficas de estos algoritmos, tanto arraySum, como ArrayMax, y ArraySum puden alcanzar a ejecutarse con grandes valores en muy poco tiempo, pues su complejidad es o(n), y por otro lado el algoritmo MergeSort presenta una complejidad O(n log n),lo que puede demostrarse con las gráficas, mientras que el algoritmo InsertionSort tiene una complejidad mayor, haciendo que sea más lento, y no sea posible usarlo con valores muy grandes.

4. *El algoritmo InsertionSort, para valores muy grandes, aparte de que se demora muchísimo tiempo ejecutandose, en un computador con poca memoria, esta se llenaría e impediría su ejecución. Esto se debe a su complejidad O(n^2).*

5. *Para ArraySum con valores muy grandes, lo único que pasaría es que se llene la memoria y esto impida su ejecución, puesto que su complejidad es O(n), este algoritmo de ejecuta eficientemente aun para valores grandes.*

6. *Para arreglos grandes, MergeSort muchísimo más eficiente que InsertionSort, pues los tiempos del algoritmo de insertion sort crecen rapidamente, lo que lo hace más lento y por esto más ineficiente a la hora de trabajar con valores grandes. El algoritmo insetion sort se usa principalmente con valores pequeños, debido a que en estos su comportamiento es mejor.*

7.

```
public int maxSpan(int[] nums) {
   if (nums.length > 0) {
      int maxSpan = 1;
      for (int i = 0; i < nums.length; i++)
         for (int j = nums.length - 1; j > i; j--)
            if (nums[j] == nums[i]) {
               int count = (j - i) + 1;
               if (count > maxSpan) maxSpan = count;
               break;
            }
      return maxSpan;
   } else return 0;
}
```

Algoritmo tomado de: http://gregorulm.com/codingbat-java-array-3-part-i/

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

En el algoritmo maxSpan, primero se verifica si la longitud es mayor que cero, si lo es, se ejecuta un ciclo que va desde la primera posición del arreglo hasta la última, y por cada ejecución de este se ejecutará otro ciclo que irá desde la última posición hasta la posición actual del primer arreglo comprobando si el valor de la posición del primer ciclo es igual a la del ultimo, si esto ocurre en la variable count se almacena la diferencia entre estas dos posiciones, mas uno ya que la última posición se cuenta, seguido de estos e verifica si count es mayor que maxspan, es decir si es mayor que 1, si es mayor maxSpan tomará la el valor de count. Este procedimiento se realizará hasta que el primer ciclo alcance la longitud del arreglo. Terminado esto se retornará el valor almacenado en maxspan. Por otro lado si la longitud es cero, se retornara cero.

8. Array-2

```
public int[] fizzArray(int n) {
  int [] array= new int[n];  //c1
  for(int i=0; i<n;i++){
    array[i]=i;             //c2 + n
  }
  return array;
 }
```

T(n)=c2 + n
O(n)= O(n)

```
  public int bigDiff(int[] nums)
{
  Arrays.sort(nums); //c1
  if(nums.length>1){
  return (Math.abs(nums[0]-nums[nums.length-1]));   //c2
}
  else{
    return 0;
  }
}
```

T(n)=C
O(n)= O(1)

```
public int countEvens(int[] nums) {
  int contador=0;   //c1
  for(int i=0; i<nums.length;i++){
    if(nums[i]%2==0){  //c2 + n
      contador++;    //c3 + n
    }
  }return contador;
}
 T(n)=C+n
 O(n)=O(n)



public int sum13(int[] nums) {
  int a=0;   //c1
 for(int i=0;i<nums.length;i++){
   if(nums[i]!=13){    //c2 + n
     a+=nums[i];   c3 + n
   }else{
     i++;    //c4 + n
   }
 }return a;
}
T(n)= C + n
O(n)=O(n)


public boolean has22(int[] nums) {
  boolean boleano=false;   //c1
  for(int i=0;i<nums.length-1;i++){
    if(nums[i]==2 && nums[i+1]==2){ //c2 + n
      boleano= true;}   //c3 + n

  }return boleano;
}
T(n)=C + n
O(n)=O(n)
```

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

```
        public boolean sum28(int[] nums) {
          int suma=0;   //c1
          for(int i=0;i<nums.length;i++){
            if(nums[i]==2){  //c2 + n
              suma+=2;  //c3 + n
            }
          }if(suma==8){
            return true; //c4
          }else{
            return false;  // c5
          }
}
```

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

Array-3

```
        public  boolean canBalance(int[] nums) {
          int a=0;   //c1
          int b=0;  //c2
          for(int i=0;i<nums.length-1;i++){
              for (int j=0;j<=i;j++){
              a= a + nums[j];  //c3 + n*m
              }
              for(int k =i+1 ; k<nums.length;k++){
                b=b + nums[k]; //c4 + n
              }
              if (a==b){
                return true; c5
              }
              a=0; //c6
              b=0; //c7
          }
          return false;
}
```

T(n)=C + m*n
O(n)=O(m*n)

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

```
        public int[] seriesUp(int n) {
          int [] matriz= new int [n*(n+1)/2]; //c1
          for (int i=matriz.length-1; i>=0;i--){
          for (int j=n;j>0;j--){
            matriz [i]=j; //c2 + m*n
            i=i-1;  //c3 + m*n
         }
          n=n-1;   // c4 * m
        i=i+1; //c5 *¨m
        }
           return matriz;
  }
T(n)=C + m*n
O(n)=O(m*n)


        public int[] fix34(int[] nums) {
            for(int i=0;i<nums.length;i++){
              if(nums[i]==3 ){  // c1 + n
                int a= nums[i+1];  // c2 +n
                nums[i+1]=4;   //c3+ n
                for(int t=i;t<nums.length;t++){
                  if (nums[t]==4 && nums[t-1]!=3){ //c5 + m*n
                    nums[t]=a; //c6 + m*n
                  }
                }
              }
            }
            return nums;
}
T(n)=C + m*n
O(n)=O(m*n)
```

```
public boolean linearIn(int[] outer, int[] inner) {
  int algo=0;  //c1
  boolean alg=false;  //c2
  for(int i=0;i<inner.length;i++){
    alg=false;  //c3 + n
    for (int e=0;e<outer.length;e++){
      if(inner[i]==outer[e]){  //c4 + m*n
        alg=true; //c5 + m*n
      }     }
    if (alg==true){
      algo++;  //c6
    }
  }
  if(algo==inner.length){
    return true;   //c7
  }
  else{
    return false;  //c8
  }}
T(n)= C  + m*n
O(n)=O(m*n)



public int countClumps(int[] nums) {
  int count=0;   //c1
  for(int i=0; i<nums.length-1;i++){
    if(i>=1 && nums[i]==nums[i-1]){
continue; //c2 + n
}
    if(nums.length>1 && nums[i]==nums[i+1]){
      count=count + 1; //c3 + n
    i=i+1; //c4
    }
  }
  return count;
}
T(n)=C + n
O(n)=O(n)
```

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

9. . tanto los algoritmos de array 2 como los de array 3 presentan una complejidad, En cada uno de estos la “n” representa la variable que hará que el tiempo que se demore en ejecutarse sea mucho o poco, en la mayoría de estos la n representa el tamaño del arreglo ingresado, pero puede variar dependiendo de las los pasos que se vayan a realizar.

4) Simulacro de Parcial

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

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