**DESARROLLO DE ALGORITMOS**

**Trabajo Práctico Tiempos de Ejecución y Orden**

1. **Cálculo de orden. Dadas las siguientes funciones:**
2. f(n)=100
3. f(n)=60 \* n
4. f(n)=n\*log(n)
5. f(n)=2n+n2
6. f(n)=n+log(n)

1. Representarlas gráficamente utilizando algún graficador.
2. Ordenarlas en forma decreciente de acuerdo a su velocidad de crecimiento.
3. Indicar a qué Orden pertenece cada una.
4. ¿Para qué tamaño de la muestra (n) un algoritmo cuya eficiencia respeta la función 2) es mejor que la de 1)?
5. ¿Para qué tamaño de la muestra (n) un algoritmo cuya eficiencia respeta la función 2) es mejor que la de 3)? Ayuda: es MUY grande
6. ¿Para qué tamaño de la muestra (n) un algoritmo cuya eficiencia respeta la función 2) es mejor que la de 4)?

1. **Calculando tiempos de ejecución.**

Para cada uno de los siguientes subprogramas o segmentos de código:

* Expresar en función de ***n*** el tiempo de ejecución.
* Establecer el Orden utilizando las reglas de la suma y del producto.

a)

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | public static void multiplo()  {  int n=1;  while (!(n == 10))  n = n+1;  System.out.println("Se iteraron: "+n+ " veces ");  } |
|  |  |

b)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | ALGORITMO Ejemplo  LEER (numero)  PARA i<-1 HASTA numero PASO 1 HACER  SI (numero mod i=0) ENTONCES  ESCRIBIR ( i,’Es divisor’)  FIN SI  FIN PARA  FIN ALGORITMO Ejemplo |

b

c)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | public static void mostrar(int[] arreglo)  {  int largo = arreglo.length;  for (int i=0; i<largo; i++)  {       System.out.print(arreglo[i]+" - ");  }  } |

d)

|  |  |  |  |
| --- | --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | | ALGORITMO Ejemplo2  LEER (letra)  SI (letra='A' OR letra='E' OR letra='I' OR letra='O'OR letra='U') ENTONCES  Si (letra='A') ENTONCES  ESCRIBIR ( ’Es la vocal A’)  SINO  Si (letra='E') ENTONCES  ESCRIBIR ( ’Es la vocal E’)  SINO  ESCRIBIR ( ’Es otra vocal’)  FIN SI  FIN SI  SINO  Escribir( ’No es vocal’)  FIN SI  FIN ALGORITMO Ejemplo2 | |
|  |  | |  |

e)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | public static boolean verifica(int[] arreglo, int elemento)  {  int largo = arreglo.length;  int i=0;  boolean esta=false;  while ( i<largo && !esta)  {  if (arreglo[i]==elemento)  esta= true;  i++;  }  return esta;  } |

f)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | public static void reemplaza(int[] arreglo,int num, int nuevo)  {  int largo = arreglo.length;  for (int i=0; i<largo; i++)  {  if (arreglo[i]==num)  arreglo[i]=nuevo;  }  } |

g)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | MÓDULO Misterio(ENTERO  n) RETORNA VACÍO  ENTERO i,j,k,s  S <-0  PARA i<- HASTA n-1 PASO 1 HACER  PARA j<- 1 HASTA n PASO  1 HACER  s<-s+2  FIN PARA  FIN PARA  ESCRIBIR(“s:”+s)  FIN MÓDULO Misterio |

h)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | public static void mostrar(int[][] matriz)  {  for (int f=0; f<matriz.length;f++)  {  for (int col=0; col<matriz[0].length; col++)  {  System.out.print( matriz[f][col]+" ");  }  System.out.println("");  }  } |

i)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | private static void misterio (int n)  {  int i, j, k=0;  for (i = 1; i <= n-1; i++)  {  for (j = 1; j < i; j++)  {  k = i \* j;  }  }  } |

j)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | private static int calculo(int n)  {  int i, j, cantA = 0, cantB = 0;  for(i=1; i<=n; i++)  {  for(j=1; j<i; j++)  {   cantA++;   cantB++;   }   }   return cantB;  } |

k)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | private static int queHace(int[] a)  {  //n es la dimensión del arreglo a  int n, max, cont, i, j; n = a.length; max = 0;  for(i=0; i<n; i++)  {  cont = 1;  j = i + 1;  while ( ( j < n ) && ( a[i] <= a[j] ) )  {  j++;  cont++;  }  if (cont > max)  max= cont;  }  return max;  } |

l)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | j=1;  while (j <= n)  {  if (a[j] <= a[n] )  a [n]= a[j];  j=j\*2;  } |