**Programación 2 (INF210])** **Práctico #2** **Gestión 2017**

# Tema: Algoritmia

1. Crear un algoritmo **eficiente** que genere la serie de Fibonacci. El algoritmo será eficiente será según logre el menor tiempo de ejecución.

2. Hacer un algoritmo que, a partir de un número N, obtenga las ternas pitagóricas (A, B, C) en las que A, B y C son Todos enteros positivos y menores que N. Para que una terna (A, B, C) sea pitagórica debe cumplir que $A^2+B^2=C^2$.

3. En una computadora con velocidad de 0.003s por instrucción, determine para que tamaños de entrada es más rápido un algoritmo con función de coste $\mathbf{10N^3}$ que otra función de coste $\mathbf{2^N}$

4. Escribe dos algoritmos que eliminen los elementos de un vector de uno en uno. El primero que elimina los elementos del final, el segundo que elimina los elementos del primer lugar. Calcule el tiempo de cada uno de los algoritmos. Luego indica cuál de los dos algoritmos es más eficiente.

5. El algoritmo de ordenación tal vez más sencillo sea el denominado de intercambio que ordena los elementos de una lista en orden ascendente. Este algoritmo se basa en el recorrido secuencial del vector a ordenar, comparando el primer elemento del vector con los restantes y efectuando intercambio de posiciones cuando el orden de la comparación no sea el correcto. El algoritmo se ilustra con el vector original [8, 4, 6, 2] que se convertirá en el vector ordenado [2, 4, 6, 8]. El algoritmo realiza n − 1 pasadas (3 en el ejemplo), siendo n el número de elementos, y ejecuta las siguientes operaciones.
   El elemento de la posición 0 (a[0]) se compara con cada elemento posterior del vector de posiciones 1, 2 y 3. En cada comparación se comprueba si el elemento siguiente es más pequeño que el elemento de la posición 0, en ese caso se intercambian. Después de terminar todas las comparaciones, el elemento más pequeño se localiza en la posición 0.

   **Pasada 1**

   El elemento menor ya está colocado la posición 0, quedando el vector [2, 8, 6, 4]. El algoritmo continúa comparando el elemento de la posición 1 con el resto de los elementos de la derecha.
   Por cada comparación, si el elemento mayor está en la posición 1 se intercambian los elementos. Después de hacer todas las comparaciones, el segundo elemento menor del vector se almacena en la posición 1.

   **Pasada 2**

   El vector a ordenar ahora es [2, 4, 8, 6]. Ahora se realiza una sola comparación entre los dos últimos elementos del vector y después del intercambio queda [2, 4, 6, 8].

a) Escribir un algoritmo que ordene los elementos de un vector de la forma descrita anteriormente.
b) Calcule el tiempo de ejecución realizando su respectiva tabla de conteo.
c) Indique, por su forma de trabajar, a cuál de los algoritmos analizados en clases se parece más.
d) Indicar a cuales algoritmos de ordenamiento analizados anteriormente es mejor
e) Indicar a cuales algoritmos de ordenamiento analizados anteriormente es peor.

**6. Para los siguientes ejercicios calcular el T(n)**

**Ejercicio 1:**

```
void Pro1(Word n)
{  for (int i=1  ;  i<=n ; i++)  {
            for (int j=n ;  j>= i+1 ; j-- ) {
                if  (  a[j-1] >  a[j] ) {
                     int t temp= a[j − 1];
                     a[j − 1] = a[j];
                     a[j] = temp;
                 }
            }
    }
}
```

**Ejercicio 2:**

```
void Proceso2(Word n)
{ int cont,j, max = 0;
  for (int i = 1; i<=n; i++) {
      cont = 1;     j = i + 1;
      while (a[i] <= a[j]) {
         j = j + 1;
         cont = cont + ;
      }
      if (cont > max)
         max = cont;
  }
 }
```

**Ejercicio 3:**

```
Cardinal Proceso3(int a[], Word n, int c)
{ Cardinal inf,sup, i, p ;
  p=0; inf=1;   sup=n;
  while (sup>=inf) {
     i=(inf+sup) / 2;
     if (a[i]==c)
        p= i;
    else {
       if (c<a[i]) {
          sup = i -1 ;
       else
          inf =i +1
   }
  }
  return p;
}
```

**Ejercicio 4:**

```
int  Computa4(int N)
{ int A,B,i,j;
   A=0;    i=1;
   while ( i <= N ) {
        B=1;
        j=1;
        while ( j <= 3 ) {
            B=B*i;
            j++;
        }
        A=A+B;
        i++;
  }
  return  A;
}
```

**Ejercicio 5:**

```
Cardinal Euclides(m,n:CARDINAL)
{ Cardinal temp;
   while (m>0) {
       temp:=m;
       m=n mod m;
       n=temp;
   }
   return n ;
}
```

**Ejercicio 6:**

```
void   Complex(Cardinal n)
{ int   m=1;
   for (int a=1; a<= (n -1); a++) {
        for (int b=a+1; b<=n; b++) {
            for (int c=1; c<=b; c++) {
                m=m*2;
            }
        }
   }
}
```

**7. Para los siguientes ejercicios calcular el T(n)**

**Ejercicio 1:**

```
int Pro71(int v[], int a, int b)
{ int  p, n;
  n = b - a+1;
  if (n==0)
     p=0;
  else if (n==1)
     p=v[0];
  else
     p=Pro71(v,a+1,b-1)+v[a]+v[b];
  return p;
}
```

**Ejercicio 2:**

```
void   Pro72(Cardinal n)
{
  if (n>2){
     int a=n-1;
     int b=(a + 1)/2;
    Pro72(b);
  }
}
```

**Ejercicio 3:**

```
void   Pro73(Cardinal n)
{
  if (n>2){
     Pro72(n);
     Pro73(n-1);
  }
}
```

**Ejercicio 4:**

```
void   Pro74(int v[], Cardinal n)
{
  if (n>2){
     Pro71(v, n);
     n = n – 1;
     Pro74(v, n);
  }
```

**Ejercicio 5:**

```
void   Pro75(int v[], Cardinal n)
{
  if (n>2){
     Pro75(v, n-1);
     n = n – 1;
     Pro74(v, n+1);
  }
```