UNLP. Facultad de Informática.

Algoritmos y Estructuras de Datos 2007. **Trabajo Práctico 2 – Parte C**

Tiempos de Ejecución - Recurrencias

Objetivos

- Expresar la función recurrente de un algoritmo
- Resolver la recurrencia

Ejercicio 1.

i)

```
int rec2 (int n) {
   if (n <= 1)
      rec2 = 1;
   else
      rec2 =  2 * rec2 (n-1);
   return rec2;
}
```

ii)

```
int rec1 (int n) {
   if (n <= 1)
      rec1 = 1;
   else
     rec 1 =  rec1 (n-1) + rec1 (n-1);
   return rec1;
}
```

iii)

```
int  f (int n) {
  if  ( n == 0 )
     f = 0;
  else { if ( n == 1 )
            f = 1;
          else  f = f (n-2) x f
(n-2);
        }
  return f;
}
```

iv)

```
int  potencia_iter (int x, n) {
   int potencia
    if  ( n == 0 )
         potencia = 1;
    else { if  ( n == 1)
              potencia  =  x:
          else  {
               potencia  = x;
               for  (int  i  = 2 ; i ≤  n ; i++) {
                  potencia = potencia * x ;
                }
         }
    }
  return  potencia;
}
```

v)

```
int  potencia_rec (int x, n) {
  if  ( n == 0 )
     potencia_rec = 1;
  else  {if  ( n == 1)
           potencia_rec  =x;
          else {
            if ( n  %  2 ) = 0
                potencia_rec =  potencia_rec (x * x, n div 2 );
           else  potencia_rec =  potencia_rec (x * x, n div 2) * x;
         }
   }
  return potencia_rec;
}
```

a) Para cada uno de los algoritmos presentados:
- **Expresar** en función de n el tiempo de ejecución.
- **Analizar** y **resolver** la correspondiente recurrencia.
- **Determinar** el orden de las funciones obtenidas usando notación *big-Oh*.

d) **Comparar** el tiempo de ejecución de la función 'rec2' con el de la función 'rec1'.

e) **Implementar** un algoritmo más eficiente que la función f (es decir que el T(n) sea menor).

f) **Implemente** una clase llamada **Algoritmos** en el paquete **estructurasdedatos**, que contenga los métodos dados (declarados como métodos de clase). Incluya la implementación del inciso e)

Ejercicio 2.

Resolver las siguientes recurrencias :

a)

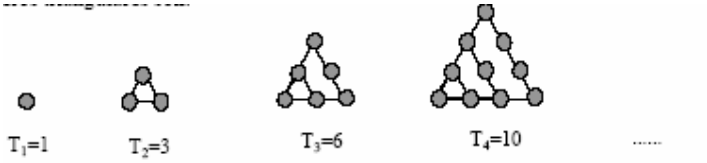
$$T(n) = \begin{cases} 1 & \text{si } n = 1 \\ 8\,T(n/2) + n^3 & \text{si } n \geq 2 \end{cases}$$

b)

$$T(n) = \begin{cases} c & \text{si } n = 0 \\ d & \text{si } n = 1 \\ [T(n-2)]^2 & \text{si } n \geq 2 \end{cases}$$

c) $T(n) = \begin{cases} 2 & \text{si } n = 1 \\ T(n-1) + n & \text{si } n \geq 2 \end{cases}$	d) $T(n) = \begin{cases} 2 & \text{si } n = 1 \\ T(n-1) + n/2 & \text{si } n \geq 2 \end{cases}$

Ejecicio 3.
Los números triangulares son:



Encontrar la función de recurrencia para T(n) y resolverla.

Ejercicio 4.
Considerar la estrategia *mergesort*, que permite ordenar un arreglo recursivamente. Se divide el vector en dos partes, se ordena con *mergesort* cada una de ellas, y luego se combinan ambas partes ordenadas.
a) Define una clase **OrdenArreglo** en el paquete **estructurasdedatos.utiles** e implemente el método mergesort como un método de clase.
b) **Hallar** la función del tiempo de ejecución del algoritmo planteado.