Reto 1: Estructura de Datos

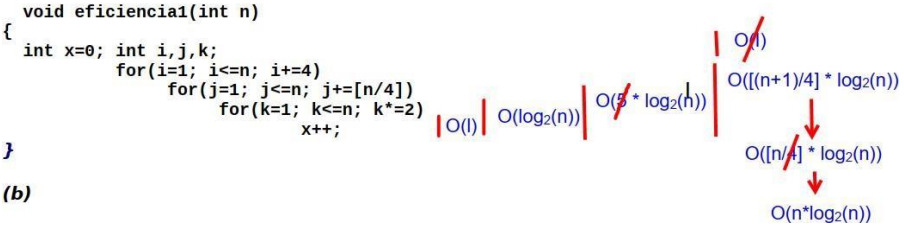
David Martínez Díaz GII-ADE

Ejercicio 1: Usando la **notación O,** determinar la eficiencia de las siguientes funciones:

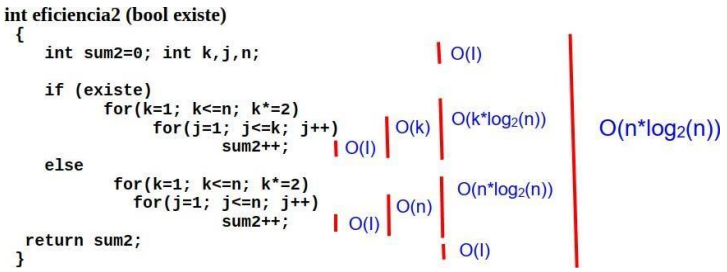
(a)

```
  void eficiencia1(int n)
{
  int x=0; int i,j,k;
           for(i=1; i<=n; i+=4)
                for(j=1; j<=n; j+=[n/4])
                     for(k=1; k<=n; k*=2)
                              x++;
}
```

(b)

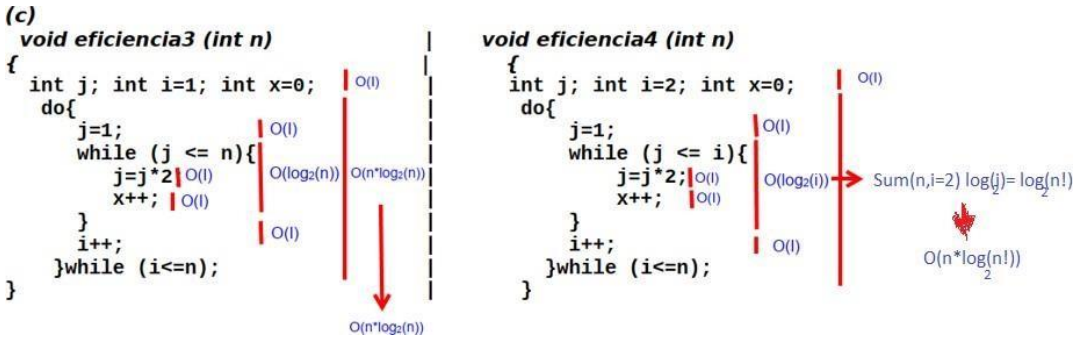


*En este caso podemos hacer dos cosas, quitamos las constantes, pudiendo simplificarlo ya que simplemente tenemos que quedarnos con la función, por otro lado podemos simplificar el (n+1)/4 quedándose una simple n.



*En el apartado b, simplemente vamos aplicando la regla del producto y por ultimo nos quedamos con la situación más compleja y nos quedamos con el mas "grande".

(c)



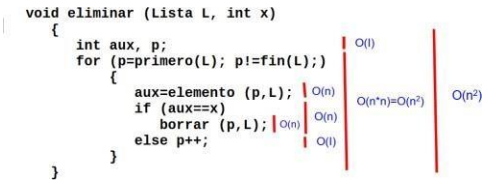
*En el apartado c, simplemente podemos ver cómo cambia la situación al cambiar lo que depende el bucle while, donde en la segunda situación tenemos que despejar la i en función de la n, utilizando una progresión aritmética, aplicando la formula.

Ejercicio 2: Considerar el siguiente segmento de código con el que se pretende buscar un entero **x** en una lista de enteros L de tamaño n (el bucle **for** se ejecuta **n veces**):

```
void eliminar (Lista L, int x)
   {
      int aux, p;
      for (p=primero(L); p!=fin(L);)
          {
             aux=elemento (p,L);
             if (aux==x)
                borrar (p,L);
             else p++;
          }
   }
```

Analizar la eficiencia de la función eliminar si:

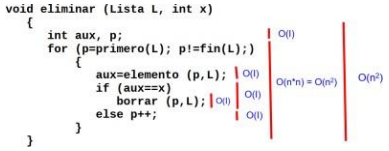
(a) primero es O(1) y fin, elemento y borrar son O(n). ¿Cómo mejorarías esa eficiencia con un solo cambio en el código?



Para mejorar la eficiencia del código, puedes crear un entero (int), que se llame final_aux, fuera del bucle for, para que la función fin(L), no se tenga que repetir más de una vez, es una pequeña mejora en el código ya que así detectamos el centinela y lo guardamos para hacer sus respectivas comparaciones, sin tener que depender de nada más. Este quedaría como condición en el bucle for:

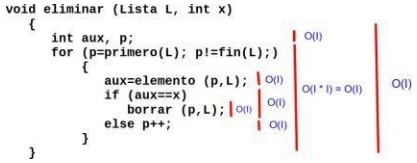
➔ final_aux = fin(L);
➔ for (p=primero(L); p!=final_aux);

(b) primero, elemento y borrar son O(1) y fin es O(n). ¿Cómo mejorarías esa eficiencia con un solo cambio en el código?



En este caso creo que podríamos utilizar la misma opción que en el apartado 1.

(c) Todas las funciones son O(1). ¿Puede en ese caso mejorarse la eficiencia con un solo cambio en el código?



En el caso de que todas las funciones son O(I), no se puede mejorar su eficiencia.