UNIVERSIDAD DE GRANADA

Departamento de Ciencias de la Computación e Inteligencia Artificial

Reto 1: Eficiencia

J. Fdez-Valdivia
Dpto. Ciencias de la Computación e Inteligencia Artificial
E.T.S. de Ingenierías Informática y de Telecomunicación
Universidad de Granada

Estructuras de Datos

Grado en Ingeniería Informática
Doble Grado en Ingeniería Informática y Matemáticas
Doble Grado en Ingeniería Informática y ADE

*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++;
}
```

O(1) | O(log2(n)) | O(n/4 * log2(n)) | O(1) | O([(n+1)/4] * log2(n)) → O([n/4] * log2(n)) → O(n*log2(n))

(b)

```
  int eficiencia2 (bool existe)
   {
      int sum2=0; int k,j,n;

      if (existe)
            for(k=1; k<=n; k*=2)
                 for(j=1; j<=k; j++)
                        sum2++;
      else
             for(k=1; k<=n; k*=2)
               for(j=1; j<=n; j++)
                        sum2++;
    return sum2;
   }
```

O(1) | O(k) | O(k*log2(n)) | O(1) | O(n) | O(n*log2(n)) | O(1) | O(n*log2(n))

(c)

```
 void eficiencia3 (int n)
{
  int j; int i=1; int x=0;
   do{
      j=1;
      while (j <= n){
         j=j*2;
         x++;
      }
      i++;
    }while (i<=n);
}
```

O(1) | O(1) | O(1) | O(log2(n)) | O(1) | O(1) | O(n*log2(n)) → O(n*log2(n))

```
 void eficiencia4 (int n)
  {
  int j; int i=2; int x=0;
   do{
      j=1;
      while (j <= i){
          j=j*2;
          x++;
      }
      i++;
    }while (i<=n);
  }
```

O(1) | O(1) | O(1) | O(log2(i)) → Sum(n,i=0)--> [(n-2) * ((n-2)+1)/2] → O(n*log2(n²)) | O(1)

*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++;
              }
       }
```

O(1) | O(1) | O(1) | O(1) | O(n*n) = O(n²) | O(n²)

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?

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

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

a - b) Hay que hacer un entero para fin, para que vaya haciendo fin_aux --; para cuando se llame a la funcion fin y esta termine siendo un O(l).

c) Al ser todas las funciones constantes, no se puede mejorar su eficiencia.

Consideraciones:

1.- El reto es ***individual***

2.- la solución deberá entregarse obligatoriamente en un fichero pdf (se sugiere como nombre reto1.pdf)

3.- Si la solución es correcta, se puntuará con 0.2 para la evaluación contínua

4.- El plazo límite de entrega es el 3 de Octubre a las 23.55h