UNIVERSIDAD DE GRANADA

Departamento de Ciencias de la Computación e Inteligencia Artificial

Reto 1: Eficiencia

Yeray López Ramírez
E.T.S. de Ingenierías Informática y de Telecomunicación
Universidad de Granada

Estructuras de Datos

Grado en Ingeniería Informática

1.- Usando la ***notación O,*** *determinar la eficiencia de las siguientes* `funciones`:

(a)

```
  void eficiencia1(int n)
{
  int x=0; int i,j,k;                          ]O(1)
           for(i=1; i<=n; i+=4)                                             ]O(!4*log2(n/4!))
                for(j=1; j<=n; j+=[n/4])                    ]O(4*log2n)          ∈
                     for(k=1; k<=n; k*=2)          ]O(log2n)                 O(log2n!)
                             X++;              ]O(1)                               ∈
}                                                                            O(n*log2n)
```

(b)

int eficiencia2 (bool existe)

```
   {
      int sum2=0; int k,j,n;          ]O(1)

      if (existe)
            for(k=1; k<=n; k*=2)                     ]O(n*log2n)
                 for(j=1; j<=k; j++)        ]O(n)                   ]O(2n*log2n)
                        sum2++;       ]O(1)                              ∈
      else                                                          O(n*log2n)
             for(k=1; k<=n; k*=2)
               for(j=1; j<=n; j++)          ]O(n)    ]O(n*log2n)
                        Sum2++;       ]O(1)
    return sum2;
   }
```

(c)

```
 void eficiencia3 (int n)                      |     void eficiencia4 (int n)
{                                              |       {
  int j; int i=1; int x=0;                     |       int j; int i=2; int x=0;
   do{                                         |        do{
      j=1;      ]O(1)                          |            j=1;      ]O(1)
      while (j <= n){          ]O(log2n)  ]O(n*log2n)+n     while (j <= i){     ]O(log2i)  ]O(log2n!)+n
         j=j*2; ]O(1)                            ∈              j=j*2;]O(1)                      ∈
         x++;   ]O(1)                      O(n*log2n)           x++;  ]O(1)                O(n*log2n)
      }                                                     }
      i++;      ]O(1)                          |            i++;      ]O(1)
    }while (i<=n);                             |          }while (i<=n);
}                                              |        }
```

*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 ligero cambio en el código?*

```
void eliminar (Lista L, int x)
   {
      int aux, p;                          ]O(1)
      for (p=primero(L); p!=fin(L);)       ]O(n)
          {
             aux=elemento (p,L);           ]O(n)
             if (aux==x)                              ] O(n)
                borrar (p,L);              ]O(n)
             else p++;                     ]O(1)
          }
   }
```

O(3n²) ∈ O(n²)

Dado que dentro del bucle for se encuentra 3 O(n) entonces la eficiencia del algoritmo es O(3n²). Con un ligero cambio no podemos mejorar su eficiencia ya que implicaría sacar a elemento() del bucle lo cual es imposible si queremos acceder al resto de elementos.

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

```
void eliminar (Lista L, int x)
   {
      int aux, p;                          ]O(1)
      for (p=primero(L); p!=fin(L);)       ]O(n)
          {
             aux=elemento (p,L);           ]O(1)
             if (aux==x)                              ] O(1)
                borrar (p,L);              ]O(1)
             else p++;                     ]O(1)
          }
   }
```

O(n²+2n) ∈ O(n²)

A pesar de que hemos mejorado la eficiencia de 2 funciones, la eficiencia del algoritmo es similar al anterior por culpa de fin(). Se podría mejorar si sacamos fin(L) del bucle e introducimos su valor en otra variable auxiliar(int ultimo = fin(L)) Pasando a ser O(3n) ∈ O(n) que es mucho más eficiente.

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

```
void eliminar (Lista L, int x)
   {
      int aux, p;                          ]O(1)
      for (p=primero(L); p!=fin(L);)       ]O(n)
          {
             aux=elemento (p,L);           ]O(1)
             if (aux==x)                              ] O(1)
                borrar (p,L);              ]O(1)
             else p++;                     ]O(1)
          }
   }
```

O(n)

Incluso siendo todo O(1), la eficiencia del algoritmo es O(n) debido al bucle **for**. La única forma de mejorar el código sería no usando bucles lo cual es imposible si queremos buscar un elemento en una lista o vector.