RETO 1 Javier Gómez López

Friday, 17 September 2021 10:28

Ejercicio 1

(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++;
}
```

$O(1) + [O(1) \cdot O(\log n) \cdot O(4n) \cdot O(\frac{n}{4})] \Rightarrow O(n^2 \cdot \log n)$

(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;
 }
```

$\sum_{k=1}^{\log n} k = \log n \cdot \frac{\log n + 1}{2} \Rightarrow O([\log n]^2)$

El peor de los casos es $O(n \cdot \log n)$

(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);
 }
```

$O(n) \cdot [O(1) + O(\log n) \cdot [O(1) + O(1)] + O(1)] \Rightarrow O(n \cdot \log n)$

```
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);
 }
```

$O(n-1) \Rightarrow \sum_{i=2}^{n} \log i = n \cdot \frac{\log n + \log 2}{2} \Rightarrow O(n \log n)$

Ejercicio 2.

Caso 1

```
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++;
          }
   }
```

$O(1) + O(n) + O(n) \cdot [O(n) + O(n) + O(1)] \Rightarrow$

$\Rightarrow O(1) + O(n) + O(n^2) \Rightarrow O(n^2)$

Para mejorar el código, utilizaría una variable auxiliar que tome el valor $y = fin(L)$. En el for se comprobaría $p != y \Rightarrow O(1)$. La eficiencia mejoraría a $O(n)$

Caso 2

```
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++;
          }
   }
```

$O(1) + O(n) + O(n) \cdot [O(n) + O(1) + O(1) \cdot O(1)] \Rightarrow$

$\Rightarrow O(n^2)$

Para mejorar el código usaríamos la misma estrategia que en el apartado anterior, quedando la eficiencia, de nuevo, $O(n)$

Caso 3

```
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++;
          }
   }
```

$O(1) \cdot O(n) \Rightarrow O(n)$

En este caso, sobre el papel, no sería posible mejorar la eficiencia. Sin embargo, sería recomendable sacar fin(L) del bucle para evitar llamar todo el rato a la función, que es menos eficiente que una comparación.