Ejercicios de complejidad:

1. Escribir versiones recursivas para los algoritmos de ordenamiento lento y determinar su coste.
2. Determinar el coste de las implementaciones realizadas para las operaciones de altas, bajas y búsquedas en estructuras de listas ligadas con punteros (analizar el código diseñado).
3. Diseñar un algoritmo que reciba un vector de n enteros, el valor de n y un entero x, y determine si en el vector hay dos enteros que sumados dan x. El algoritmo debe tener un costen temporal O(n log n) .
4. Calcular el coste temporal de:

Int Suma (int v[], int n)

{

if (n<1) return v[0];

else return v[n]+Suma(v, n-1);

}

1. Diseñar una versión no recursiva del Quick Sort y determinar su coste temporal.
2. Calcular el coste del algoritmo que usaban los babilonios para calcular la raíz cuadrada:

float raíz (float x)

{

float b, a;

b=x;

a=1.0;

while (b!=a)

{

b=(a +b)/2;

a= (x/b);

}

return b;

}

1. Calcular el coste temporal del algoritmo para obtener la potencia entera de exponente entero mayor o igual a 0 que usa la estrategia *Divide Y Vencerás*.
2. Calcular el coste de las siguientes recurrencias básicas y dar un ejemplo de algoritmo que corresponda a cada ecuación:
   * 1. T(n)= T(n-1) +1 T(0)=1
     2. T(n)=2T(n-1)+1 T(0)=1
     3. T(n)=T(n-1) +n T(0)=1
     4. T(n)=T(n/2)+1 T(1)=1
     5. T(n)=T(n/2)+n T(1)=1
     6. T(n)=2T(n/2)+1 T(1)=1
     7. T(n)=2T(n/2)+n T(1)=1