Curso de nivelación de algoritmos

Taller - Clase 3

Ejercicio 1

Implementar en Python las siguientes funciones de búsqueda, suponiendo que las listas que reciben como parámetro no contienen elementos repetidos.

a) `def busquedaLineal(x, l):` `# devuelve -1 si x no está en l`

b) `def busquedaBinaria(x, l):` `# devuelve -1 si x no está en l y l está ordenada`

Algoritmos de ordenamiento

```
def selectionSort(lista):
    for i in range(0,len(lista)):
        posmin = i
        for j in range(i+1,len(lista)):
            if lista[j]<lista[posmin]:
                posmin = j

        temp = lista[i]
        lista[i] = lista[posmin]
        lista[posmin] = temp
```

Algoritmos de ordenamiento

```
def insertionSort(lista):
    for i in range(1,len(lista)):
        valor_actual = lista[i]
        pos = i
        while pos>0 and lista[pos-1]>valor_actual:
            lista[pos] = lista[pos-1]
            pos = pos-1

        lista[pos] = valor_actual
```

Algoritmos de ordenamiento

```
def bubbleSort(lista):
    for i in range(0,len(lista)):
        for j in range(0,len(lista)-1):
            if lista[j]>lista[j+1]:
                temp = lista[j]
                lista[j] = lista[j+1]
                lista[j+1] = temp
```

Ejercicio 2

Implementar en Python una función que dadas dos listas ordenadas **L1** y **L2** sin elementos repetidos devuelva la cantidad de elementos de **L1** que aparecen en **L2** en O(|**L1**| + |**L2**|).

```
def enAmbasListas(l1, l2):                # l1 y l2 están ordenadas y sin repetidos
```

Ejercicio 3

Dado una lista cuyos valores están entre 0 y 9, implementar un algoritmo que lo ordena en O(|**L**|).