Examen Parcial

Materia: Programación Superior IMT-231

Docente: Francisco Suárez

Problema 1

Defina en sus propias palabras qué es un **diccionario** y mencione **tres métodos** comúnmente encontrados en diccionarios, explicando los argumentos que aceptan y los tipos de datos que devuelven.

Problema 2

Defina en sus propias palabras qué es un **set** y mencione **tres métodos** comúnmente encontrados en sets, explicando los argumentos que aceptan y los tipos de datos que devuelven.

Problema 3

Escriba una función llamada **es_1_a_1** que acepte un **diccionario** cuyas llaves y valores son strings y retorne **True** si no hay dos llaves que estén mapeadas al mismo valor. Por ejemplo, para el siguiente diccionario tu función debería retornar **False** porque "Hawking" y "Newton" mapean al mismo valor:

{"Marty": "206-9024",

"Hawking": "123-4567",

"Smith": "949-0504",

"Newton": "123-4567"}

Pero para el siguiente diccionario debe retornar **True** porque cada llave está mapeada a un valor único:

{"Marty": "206-9024",

"Hawking": "555-1234",

"Smith": "949-0504",

"Newton": "123-4567"}

Un diccionario vacío es considerado 1-a-1 y retorna **True.**

Problema 4

Si bien Python provee el tipo **list** para construir y manejar secuencias mutables, muchos lenguajes no tienen una estructura similar, al menos no como parte del lenguaje. Implemente la estructura de datos **Vector** usando la clase **Array** implementada en clase. Su implementación debe producir una secuencia mutable que funcione como una lista de Python. Cuando el array interno necesite ser expandido, el nuevo arreglo debe duplicar el tamaño del original. Las operaciones que pueden ser realizadas en este ADT están descritas a continuación. Asuma que el array interno nunca disminuye de tamaño.

Vector()	Crea un nuevo vector vacío con una capacidad inicial de **dos elementos**
length()	Retorna el número de elementos contenidos en el vector
contains(item)	Determina si el argumento dado está contenido en el vector
getitem(index)	Retorna el valor almacenado en el elemento especificado por index. index debe estar dentro del rango válido
setitem(index, item)	Modifica el valor almacenado en el elemento especificado por index para almacenar item. index debe estar dentro del rango válido, e lcual incluye la primera posición después del último elemento
append(item)	Agrega el item dado al final de la lista
insert(index, item)	Inserta el item dado en la posición especificada por index. Los elementos en la posición de index y siguientes deben ser desplazados para hacer espacio para el nuevo elemento. index debe estar dentro del rango válido
remove(index)	Remueve y retorna el item en la posición dada por index. Los elementos en la posición de index y siguientes deben ser desplazados para ocupar el vacío creado por el item removido. index debe estar dentro del rango válido
indexOf(item)	Retorna el índice del elemento que contenga item. item debe estar en la lista
extend(otroVector)	Extiende este vector agregando todos los elementos de otroVector
subVector(inicio, fin)	Crea y retorna un nuevo vector que contiene una subsecuencia de elementos en el vector entre las posiciones inicio y fin, incluyendo esos elementos. inicio y fin deben estar dentro del rango válido
iterator()	Crea y retorna un iterador que puede ser usado para iterar sobre los elementos del vector

Problema 5

En un **Vector** típico el tamaño del array interno disminuye luego de que un número suficiente de elementos hayan sido removidos. Implemente una estrategia para disminuir el tamaño del array a medida que los elementos son removidos. Modifique su implementación de la anterior pregunta para incluir esta estrategia.

Problema 6

Complete el tipo de dato **Set** implementado en clase, implementado los métodos **intersect()** y **difference()**.

Problema 7

Modifique el tipo de dato **Array2D** de modo que el array interno unidimensional almacene los valores dando preferencia a las columnas en lugar de filas.

Problema 8

Determine la complejidad de los métodos **intersect** y **difference** que implementó en el problema 6.

Problema 9

Evalúe cada uno de los siguientes bloques de código y determine la complejidad O(·) para los mejores y peores escenarios.

a)

```
suma = 0
for I in range(n):
    if i % 2 == 0:
        suma += i
```

b)

```
suma = 0
i = n
while i > 0:
    suma += i
    i = i / 2
```

c)

```
for i in range(n):
    if i % 3 == 0:
        for j in range(n / 2):
            suma += j
    elif i % 2 == 0:
        for j in range(5):
            suma += j
    else:
        for j in range(n)
            suma += j
```

Problema 10

Modifique el algoritmo de búsqueda binaria implementado en clase para encontrar la primera ocurrencia de un valor que puede estar múltiples veces en una lista ordenada. Asegúrese que su solución sigue siendo O(log n).