Universidad Simón Bolívar

Departamento de Computación y Tecnología de la Información

CI2691 - Laboratorio de Algoritmos y Estructuras I

Enero-Marzo 2024

Laboratorio (Semana 7)

(5 puntos)

1. Descripción de la Actividad

Se quiere que realice programas en Python que resuelvan los siguientes problemas. Todas las soluciones deberán estar en un solo script `LabSem7.py`. Serán necesarias verificaciones de los argumentos (y lanzar su correspondiente Excepción si no las cumplen), en caso de ser necesario, excepto las verificaciones de tipo.

1. **Operaciones con conjuntos** *(3 puntos)*: Supongamos que vamos a representar conjuntos de enteros usando listas. Para este problema no se podrá usar el objeto `set` de Python. Se necesita que se programen las siguientes operaciones con conjuntos. Todas las operaciones que generen un nuevo conjunto deben tener un parámetro opcional cuyo valor por defecto es `False` e indica si la operación se hace en linea. Es decir, no se genera un nuevo conjunto, si no que se modifica el primero. En caso contrario, los arreglos originales no se deben modificar.

   a) `es_conjunto` *(0.5 puntos)*: Verifica que una una lista cumple con la definición de conjunto. Es decir, todos los elementos aparecen una sola vez.
   Ejemplos de ejecución:

```
es_conjunto([0, 7])  # True
es_conjunto([0, 0])  # False
```

   b) `union` *(0.5 puntos)*: Representa la unión entre conjuntos.
   Ejemplos de ejecución:

```
a: List[int] = [0]
union(a, [1])       # [0, 1]
union(a, [2], True) # [0, 2]
print(a)            # [0, 2]
```

   c) `interseccion` *(0.5 puntos)*: Representa la intersección entre conjuntos.
   Ejemplos de ejecución:

```
a: List[int] = [0, 1, 2]
interseccion(a, [1, 3])          # [1]
interseccion(a, [2, 4], True)    # [2]
print(a)                         # [2]
```

   d) `diferencia` *(0.5 puntos)*: Representa la diferencia entre conjuntos:
   Ejemplos de ejecución:

```
a: List[int] = [0, 1, 2]
diferencia(a, [1, 3])          # [0, 2]
diferencia(a, [2, 4], True)    # [0, 1]
print(a)                       # [0, 1]
```

e) `producto` *(0.5 puntos)*: Dados dos conjuntos `A` y `B`, devuelve un nuevo conjunto que contenga todos los valores `c = a * b` donde `a` pertenece a `A` y `b` pertenece a `B`.
Ejemplos de ejecución:

```
a: List[int] = [0, 1]
producto(a, [2])           # [0, 1]
producto(a, [2, 4], True)  # [0, 2, 4]
print(a)                   # [0, 2, 4]
```

f) `conjunto_de_sumas` *(0.5 puntos)*: Dado un conjunto `A`, devuelve un nuevo conjunto que contenga la sumatoria de cada subcojunto de `A`.
Ejemplos de ejecución:

```
a: List[int] = [1, 3]
conjunto_de_sumas([1, 2])   # [0, 1, 2, 3]
conjunto_de_sumas(a, True)  # [0, 1, 3, 4]
print(a)                    # [0, 1, 3, 4]
```

2. `producto_matricial` *(1 punto)*: Dadas dos matrices de enteros A, B de dimensiones $M \times N$ y $N \times P$ respectivamente, se quiere una función que calcule el producto matricial ente ella.
Ejemplos de ejecución:

```
a: List[List[int]] = [
    [3 , 4, -1],
    [-7, 0,  2],
    [2 , 5,  8]
]
b: List[List[int]] = [
    [6,   0, -6],
    [3,   4,  5],
    [-1, -2, -3]
]
producto_matricial(a, b)
# [[31,  18,   5],
#  [-44, -4,  36],
#  [19,   4, -11]]
```

3. `particion` *(1 punto)*: Dada una lista de enteros `a`, y un número entero `X` , se debe realizar un método que retorne un número entero, cumpliendo con la siguiente especificación:

[
const $N, X : \mathbb{Z}$;
var a : array$[0..N)$ de $\mathbb{Z}$;
var $k : \mathbb{Z}$;
$\{ \quad N > 0 \ \wedge \ a = A \quad \}$
$\langle$ particion $\rangle$
$\{ \quad (\forall z \in \mathbb{Z} : \ (\#i \in \mathbb{Z} \mid 0 \leq i < N : a[i] = z) = (\#i \in \mathbb{Z} \mid 0 \leq i < N : A[i] = z) \ \ ) \ \wedge$
$0 \leq k < N \ \wedge$
$(\forall i \in \mathbb{Z} \mid 0 \leq i < k : a[i] \leq X) \ \wedge$
$(\forall i \in \mathbb{Z} \mid k \leq i < N : a[i] > X) \ \wedge$
$(\forall i, j \in \mathbb{Z} \mid 0 \leq i < j < N \wedge ((A[i] \leq X \wedge A[j] \leq X) \vee (A[i] > X \wedge A[j] > X) :$
$(\exists p, q \in \mathbb{Z} \mid 0 \leq p < q < N : A[i] = a[p] \wedge A[j] = a[q]) \ \ )$
]

Este programa se debe basar en el programa `intercambio` cuya especificación es:

$\{ (\forall i : i \neq E \wedge i \neq F : a[i] = A[i]) \wedge a[E] = A \wedge a[F] = B \}$
`intercambio(a, E, F)`
$\{ (\forall i : i \neq E \wedge i \neq F : a[i] = A[i]) \wedge a[E] = B \wedge a[F] = A \}$

Ejemplos de ejecución:

```
a: List[int] = [0, 3, 7, 1, 2, 4, 5, 2, 1]
particion(a, 3)  # 6
print(a)         # [0, 3, 1, 2, 2, 1, 7, 4, 5]

a = [9, 8, 7, 6, 5, 4, 3, 2, 1]
particion(a, 4)  # 4
print(a)         # [4, 3, 2, 1, 9, 8, 7, 6, 5]
```

2. Evaluación

Se implementará un script para la evaluación de cada solución propuesta, el cual aplicará una variedad de casos de prueba, y generará un resultado final. Se evaluarán las firmas de las funciones. Si no se pueden importar las funciones, se penalizará con -2 puntos. Las funciones que no se puedan correr sin realizar ninguna modificación tendrán la mitad de los puntos obtenidos. Aunque no se realizará una evaluación del estilo de codificación, se efectuarán correcciones en este aspecto, lo cual les será de utilidad en evaluaciones futuras, donde dicho estilo podría ser objeto de evaluación.

3. Condiciones de Entrega

El programa correspondiente al laboratorio `LabSem7.py` y la declaración de autenticidad debidamente firmada, deben ser incluidos en un archivo comprimido, en formato `tar.xz` (o `zip`), denominado `LabSem7_X.tar.xz` (o `LabSem4_X.zip`), donde `X` representa el número de carné del estudiante. La entrega del archivo debe realizarse a través de Gmail, enviándolo al correo electrónico **16-10072@usb.ve** antes de las 11:59 PM del día domingo 03 de marzo de 2024. Las entregas que se realicen después de la hora límite tendrán una penalización de un (1) punto, más un punto adicional por cada hora de retraso. Por lo tanto, después de 4 horas de retraso, la evaluación será anulada.