Fundamentos de Algoritmos – Examen Final

Master in Management + Analytics, Escuela de Negocios, UTDT

Primer semestre 2020

- El examen es individual y se aprueba con 60 puntos o más, sobre un total de 100.
- Está permitido usar Python y Spyder para hacer pruebas, consultar materiales de la materia y documentación online.
- No está permitido comunicarse por ningún medio con otros estudiantes ni con otras personas durante el examen, excepto con los docentes de la materia.
- Puede consultarse a los docentes solo por aclaraciones específicas del enunciado, a través del chat de Zoom.
- La resolución de los ejercicios debe realizarse en un único archivo de texto plano llamado "`examen.txt`". Ese archivo debe subirse a la "Entrega del examen" en el campus virtual.

Problema 1. (25 puntos) [3B5N7]
Contamos con un tipo de datos `Cliente` ya definido, con las siguientes operaciones:

- `Cliente(código, consumo)`: Crea un nuevo elemento de tipo `Cliente`, con los valores de código (un entero) y consumo (un float) especificados.
- `c.código()`: Devuelve el código del cliente `c` (un valor de tipo `int`).
- `c.consumo()`: Devuelve el consumo del cliente `c` (un valor de tipo `float`).

Se pide definir una función `consumo_promedio`, que toma como argumentos una lista de clientes y una lista de códigos, y devuelve el consumo promedio de los clientes cuyo código está en la lista de códigos. Si ningún cliente cumple esa condición, debe devolver cero.

En este problema no está permitido usar listas por comprensión, ni tampoco la función `mean`. Debe resolverse usando ciclos (`while` o `for`).

Ejemplos:

```
clientes = []
clientes.append(Cliente(1, 60.0))
clientes.append(Cliente(2, 70.0))
clientes.append(Cliente(3, 80.0))
clientes.append(Cliente(4, 90.0))
print(consumo_promedio(clientes, [1]))         # imprime 60.0
print(consumo_promedio(clientes, [3,2]))       # imprime 75.0
print(consumo_promedio(clientes, [3,4,6]))     # imprime 85.0
print(consumo_promedio(clientes, [6,7,8]))     # imprime 0.0
print(consumo_promedio(clientes, [1,2,3,4]))   # imprime 75.0
```

Problema 2. (20 puntos) [324AM]
Escribir una función en Python que, dada una lista de floats, devuelva una lista de enteros formada por el cuadrado de los elementos negativos.

En este problema no está permitido usar ciclos (`while`, `for`, etc.). Debe resolverse usando **listas por comprensión**, completando solamente la línea punteada del siguiente código:

```
1  def procesar(lista):
2      return [ ........ ]
3  }
```

Ejemplos:
`procesar([4.0, -2.0, -10.0, 12.0])` debe devolver `[4, 100]`
`procesar([])` debe devolver `[]`
`procesar([-2.0, -2.0, -2.0])` debe devolver `[4, 4, 4]`
`procesar([4.0, 1.0])` debe devolver `[]`

Problema 3. (25 puntos) [D9L69]
Considerar el tipo `CondicionMeteorologica` con las operaciones que se muestran a continuación:

- `CondicionMeteorologica(temperatura, presion, humedad)`:
  Crea un nuevo elemento de tipo CondicionMeteorologica con los datos especificados.
- `cm.temperatura()`: Devuelve la temperatura de la condición meteorológica `cm`.
- `cm.presion()`: Devuelve la presión de la condición meteorológica `cm`.
- `cm.humedad()`: Devuelve la humedad de la condición meteorológica `cm`.

A continuación puede observarse un ejemplo de uso y el comportamiento esperado:

```
1 # Nueva CM con temperatura de 13°, presion de 1016 hPa y humedad del 70 %.
2 cm = CondicionMeteorologica(13, 1016, 70)
3 print(cm.temperatura())              # imprime '13'
4 print(cm.presion())                  # imprime '1016'
5 print(cm.humedad())                  # imprime '70'
```

Se cuenta con una implementación cuyo comportamiento ha sido demostrado correcto verificándolo formalmente.

(continúa)

(continuación)

Considerar las siguientes dos funciones para calcular el promedio de temperatura de una lista no vacía de condiciones meteorológicas:

```
def temp_promedio_v1(lista):
  resultado = 0
  i = 0
  while i < len(lista) - 1:
    resultado = resultado + lista[i].temperatura()
    i = i + 1
  return resultado/len(lista)

def temp_promedio_v2(lista):
  resultado = 0
  i = len(lista) - 1
  while i > 0:
    resultado = resultado + lista[i].temperatura()
    i = i - 1
  return resultado/len(lista)
```

Para poner a prueba las funciones se escribieron los siguientes tests: [1]

```
def test_funcionan_igual(self):
  cm1 = CondicionMeteorologica(1, 1024, 50)
  cm2 = CondicionMeteorologica(2, 1024, 50)
  self.assertEqual(temp_promedio_v1([cm1,cm2,cm1]), temp_promedio_v2([cm1,cm2,cm1]))

def test_v1(self):
  cm1 = CondicionMeteorologica(4, 1024, 50)
  cm2 = CondicionMeteorologica(0, 1024, 50)
  self.assertEqual(temp_promedio_v1([cm1,cm2]), 2)

def test_v2(self):
  cm1 = CondicionMeteorologica(0, 1024, 50)
  cm2 = CondicionMeteorologica(8, 1024, 50)
  self.assertEqual(temp_promedio_v2([cm1,cm2]), 4)
```

a) Para cada test, hacer un seguimiento cuidadoso e indicar si falla o no. Justificar las respuestas.

b) Escribir dos tests, uno para cada versión, que revele que cada función tiene un error. Justificar la elección de cada test.

---

[1] Suponer que estas funciones se encuentran dentro de una clase de tests de unidad correctamente implementada.

Problema 4. (30 puntos) [446CL]
En una biblioteca pública se tienen N anaqueles, cada uno de los cuales puede albergar un máximo de M libros. Leer el siguiente código y responder las preguntas que se presentan a continuación:

```
1  def f_aux(libros):
2      i = 0
3      r = 0
4      while i < len(libros):
5          if anio(libros[i]) > 1899 and anio(libros[i]) < 2000:
6              if idioma(libros[i]) != "Espaniol":
7                  r = r + 1
8          i = i + 1
9      return r
10
11 def f(anaqueles):
12     r = 0
13     l = 0
14     for anaq in anaqueles:
15         libros = listar_libros(anaq)
16         k = f_aux(libros)
17         r = r + k
18         l = l + len(libros)
19     return r/l
```

(a) Sabiendo que la función `listar_libros(a)` tiene complejidad temporal $O(M)$ en el peor caso, y las funciones `anio(l)` e `idioma(l)` $O(1)$ en el peor caso, determinar cuál es la complejidad temporal de `f` en el peor caso en función de N (la cantidad de anaqueles) y M (la cantidad máxima de libros en un anaquel). Justificar.

(b) Renombrar las funciones `f` y `f_aux` por nombres más descriptivos que den cuenta de lo que hacen (máximo 6 palabras cada nombre, sin contar los monosílabos). Justificar la elección de los nuevos nombres.

(c) Responder las siguientes preguntas y justificar:

(I) ¿Cuál es el valor de la variable `l` después de salir del ciclo `for`?

(II) Suponiendo $N > 0$, ¿qué devuelve la función `f` si los N anaqueles están vacíos?