Repaso 2

Jonatan Goméz Perdomo, Ph.D.
jgomezpe@unal.edu.co

Arles Rodríguez, Ph.D.
aerodriguezp@unal.edu.co

Camilo Cubides, Ph.D.(c)
eccubidesg@unal.edu.co

Grupo de investigación en vida artificial – Research Group on Artificial Life – (Alife)
Departamento de Ingeniería de Sistemas e Industrial
Facultad de Ingeniería
Universidad Nacional de Colombia

Agenda

Enunciado

Al ver los precios y los anuncios del almacén *Cobra Mosmas*, un cliente le pide crear un programa de computador que le permita ingresar el precio individual de tres productos y el precio de la promoción en combo de los tres productos anunciada por el almacen y determine si es preferible comprarlos por separado o en el combo promoción.

(Pensemos por 3 minutos en definir claramente el problema)

Entendiendo el problema

Entradas : Los precios p_1, p_2, p_3 de tres productos y del combo p_c.

Salida : Un texto indicando si se debe comprar el combo o los tres productos por separado.

Relaciones : Si el precio del combo pc es menor o igual que la suma de los precios de los tres productos se debe imprimir `"Combo"` en otro caso se debe imprimir `"Por separado"`.

(Pensemos por 2 minutos en la especificación)

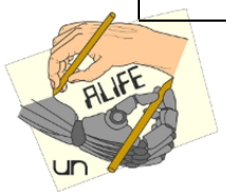
Especificando el problema

$$\mathit{comprar} : \mathbb{R} \times \mathbb{R} \times \mathbb{R} \times \mathbb{R} \mapsto \mathbb{ASCII}^*$$

$$(p_1, p_2, p_3, p_c) \rightarrow \begin{cases} \texttt{"Combo"}, & \text{si} \quad p_c \leq p_1 + p_2 + p_3; \\ \texttt{"Por separado"}, & \text{en otro caso.} \end{cases}$$

Codificación

```
def comprar(p1, p2, p3, pc):
    if pc <= p1 + p2 + p3:
        return 'Combo'
    else:
        return 'Por separado'

a = float(input('¿Precio del primer producto?: '))
b = float(input('¿Precio del segundo producto?: '))
c = float(input('¿Precio del tercer producto?: '))
d = float(input('¿Precio del combo?: '))

print("Comprar", comprar(a, b, c, d))
```



Agenda

Enunciado

Un campesino de la región le pide crear un programa de computador que le permita determinar cuál de dos opciones (madera o alambre) es la mejor opción (menor costo) para encerrar un terreno rectangular de $n * m$ metros cuadrados, sabiendo el costo de un metro lineal de alambre, el costo de un metro de madera y la cantidad de hilos de alambre o hileras de madera. El campesino sólo piensa en usar una de las dos opciones, no las piensa combinar.

(Pensemos por 3 minutos en definir claramente el problema)

Entendiendo el problema I

Entradas : Los dos lados del rectángulo n y m. El costo de un metro lineal de alambre a, el costo de un metro lineal de madera p, el número de hilos de alambre si se hace el cercado en alambre h y el número de hileras de madera w si se hace el cercado en madera.

Salida : Un texto indicando si se debe cercar en madera o si se debe cercar en alambre.

Entendiendo el problema II

Relaciones : El perímetro del rectángulo es $2n + 2m$. Una cerca en madera usará $(2n + 2m) * w$ metros lineales de madera. Una cerca de alambre usará $(2n + 2m) * h$ metros lineales de alambre. De esta manera el costo de usar alambre será $(2n + 2m) * h * a$ y el de usar madera será $(2n + 2m) * w * p$. Si el costo de usar madera es menor o igual que el de alambre se debe imprimir `"Madera"` en otro caso se debe imprimir `"Alambre"`.

(Pensemos por 2 minutos en la especificación)

Especificando el problema I

$$perimetro : \mathbb{R} \times \mathbb{R} \mapsto \mathbb{R}$$
$$(n, m) \rightarrow 2n + 2m$$

$$en\_madera : \mathbb{R} \times \mathbb{R} \times \mathbb{R} \times \mathbb{R} \mapsto \mathbb{R}$$
$$(n, m, w, p) \rightarrow perimetro(n, m) * w * p$$

$$en\_alambre : \mathbb{R} \times \mathbb{R} \times \mathbb{R} \times \mathbb{R} \mapsto \mathbb{R}$$
$$(n, m, h, a) \rightarrow perimetro(n, m) * h * a$$

Especificando el problema II

$$usar : \mathbb{R} \times \mathbb{R} \times \mathbb{R} \times \mathbb{R} \times \mathbb{R} \times \mathbb{R} \mapsto \mathbb{ASCII}^*$$

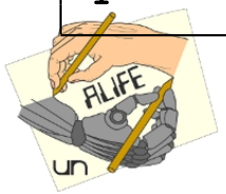
$$(n, m, h, a, w, p) \rightarrow \begin{cases} \texttt{"Madera"}, & \text{si } en\_madera(n, m, w, p) \leq \\ & \qquad en\_alambre(n, m, h, a); \\ \texttt{"Alambre"}, & \text{en otro caso.} \end{cases}$$

Codificación

```
def perimetro(n, m):
    return 2*n + 2*m

def en_madera(n, m, w, p):
    return perimetro(n,m) * w * p

def en_alambre(n, m, h, a):
    return perimetro(n,m) * h * a
```

Codificación II

```
def usar(n, m, h, a, w, p):
    if en_madera(n, m, w, p) <= en_alambre(n, m, h, a):
        return 'Madera'
    else:
        return 'Alambre'

n = float(input('¿Largo terreno?: '))
m = float(input('¿Ancho terreno?: '))
a = float(input('¿Costo metro alambre?: '))
h = int(input('¿Hilos de alambre?: '))
p = float(input('¿Costo metro madera?: '))
w = int(input('¿Hileras de madera?: '))

print("Usar", usar(n, m, a, h, p, w))
```



Agenda

Enunciado

Unos padres de familia desesperados por determinar el dinero que deben pedir prestado para pagar los útiles escolares de su hijo, le han pedido crear un programa de computador que a partir de una lista de los precios de cada útil escolar y de la cantidad de cada útil escolar en la lista, determine el precio total de la lista.

(Pensemos por 5 minutos en la solución)

Entendiendo el problema

Entradas : Una lista con los precios de los útiles (en la posición i de la lista está el precio del útil i), una lista con las cantidades de dichos útiles (en la posición i está la cantidad requerida del útil i).

Salida : El costo total de la lista.

Relaciones : El costo total es la suma sobre todos los útiles del precio del útil por la cantidad del mismo.

(Pensemos por 2 minutos en la especificación)

Especificando el problema

$$costo\_total : \mathbb{R}^* \times \mathbb{N}^* \mapsto \mathbb{R}$$

$$(precio, cantidad) \rightarrow \sum_{i=0}^{|precio|-1} precio[i] * cantidad[i]$$

Codificación

```
def costo_total(precio, cantidad):
    costo = 0
    for i in range(0, len(precio)):
        costo = costo + precio[i] * cantidad[i]
    return costo

precio = []
cantidad = []
while input('¿Ingresar otro útil?: ').upper() == 'S':
    precio.append(float(input('¿Precio útil?: ')))
    cantidad.append(float(input('¿Cantidad?: ')))
print("La lista cuesta", costo_total(precio, cantidad))
```

Agenda

Enunciado

En la última edición de la revista científica "ADN al día" se indica que las pruebas de relación entre individuos a partir de código genético se define de la siguiente manera: Si las dos cadenas se diferencian en a lo más p letras, existe una relación de padre-hijo, si se diferencian en a lo más $f > p$ letras, existe una relación de formar parte de la misma familia. De otra manera no existe relación. El laboratorio *Tein Cul Pan*, le pide desarrollar un programa que a partir de dos cadenas de ADN del mismo tamaño, determine si existe una relación padre-hijo, o de la misma familia o ninguna, siguiendo las reglas definidas por la revista científica "ADN al día".

(Pensemos por 5 minutos en la solución)

Entendiendo el problema

Entradas : Dos cadenas de caracteres (representando cadenas de ADN) a y b de la misma longitud. Los límites p y f para considerar una relación 'Padre-Hijo', 'Familiar' o 'Ninguna'.

Salida : Un texto indicando si las cadenas tienen una relación 'Padre-Hijo', 'Familiar' o 'Ninguna'.

Relaciones : Si las dos cadenas a, b se diferencian en a lo más p letras, existe una relación de 'Padre-Hijo', si se diferencian en a lo más $f > p$ letras, existe una relación de 'Familia'. En otro caso tienen 'Ninguna' relación.

(Pensemos por 2 minutos en la especificación)

Especificando el problema

$$dif : \mathbb{ASCCI}^* \times \mathbb{ASCCI}^* \mapsto \mathbb{N}$$

$$(a, b) \rightarrow \sum_{i=0}^{|a|-1} 1, \qquad \text{si } a_i \neq b_i$$

$$relacion : \mathbb{ASCCI}^* \times \mathbb{ASCCI}^* \times \mathbb{N} \times \mathbb{N} \mapsto \mathbb{ASCCI}^*$$

$$(a, b, p, f) \rightarrow \begin{cases} \texttt{"Padre-Hijo"}, & \text{si} \quad dif(a, b) \leq p; \\ \texttt{"Familiar"}, & \text{si} \quad p < dif(a, b) \leq f; \\ \texttt{"Ninguna"}, & \text{en otro caso.} \end{cases}$$

Codificación I

```
def dif(a, b):
    cuenta = 0
    for i in range(0, len(a)):
        if a[i] != b[i]:
            cuenta = cuenta + 1
    return cuenta

def relacion(a, b, p, f):
    d = dif(a, b)
    if d <= p:
        return 'Padre-Hijo'
    elif d <= f:
        return 'Familia'
    else:
        return 'Ninguna'
```

Codificación II

```
ind1 = input('¿Cadena ADN individuo 1?: ')
ind2 = input('¿Cadena ADN individuo 2?: ')
p = int(input('Diferencia máxima para ser Padre-Hijo?: '))
f = int(input('Diferencia máxima para ser Familia?: '))
print("Relación", relacion(ind1, ind2, p, f))
```

Agenda

Enunciado

Dada una lista de Universidades Colombianas, obtener el nombre del sitio web. Se asume que un nombre de universidad está entre los caracteres `www.` y `edu.co`. Por ejemplo de `www.unal.edu.co` se obtiene `unal`.

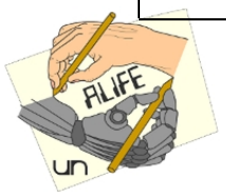
Input	Output
5 www.unal.edu.co www.udistrital.edu.co www.univalle.edu.co www.urosario.edu.co www.konradlorenz.edu.co	unal udistrital univalle urosario konradlorenz

(Pensemos por 5 minutos en la solución)

Una posible opción

Dividir la cadena separándola por el símbolo "." y quedarnos con la segunda componente:

```
def process(uni):
    return uni.split(".")[1]

def main():
    n = int(input("Número de Universidades: "))
    for i in range(n):
        uni = input("Universidad " + str(i+1) + ": ")
        print(process(uni))

main()
```



Agenda

Enunciado

Se tienen que procesar algunos comandos para realizar el procesamiento de notas de una Universidad. Se tiene una lista de estudiantes

Comando 1 : Agregar estudiante y nota

```
1&nombre_estudiante&nota
```

Comando 2 : Calcular promedio de los estudiantes en un momento dado

Comando 3 : Ordenar estudiantes agregados por nombre

Comando 4 : Consultar la nota de un estudiante

```
4&nombre_estudiante
```

Comando 5 : Visualizar lista de estudiantes

Comando 6 : Salir

(Pensemos por 5 minutos en la solución)

Análisis

Para poder resolver el problema se puede dividir dicho problema en problemas más pequeños que son:

- Definir la lista de estudiantes.
- Agregar un estudiante dada la información
- Calcular el promedio de notas de los estudiantes en un momento dado
- Ordenar estudiantes agregados por nombre
- Consultar la nota de un estudiante
- Visualizar lista
- Procesar los comandos
- Mostrar menú

(Pensemos por 5 minutos en la solución)

Agregar un estudiante

```
def agregar_estudiante(estudiantes, est):
    estudiantes.append(est)
```

Obtener promedio de notas

```
def promedio(estudiantes):
    prom = 0
    for estudiante in estudiantes:
        prom += float(estudiante[1])
    print("El promedio de los estudiantes es: ")
    print(prom/len(estudiantes))
```

Ordenar estudiantes

```
def ordenar(estudiantes):
    estudiantes.sort()
```

Consultar nota de un estudiante

```
def consultar(estudiantes, nombre):
    encontrado = False
    for estudiante in estudiantes:
        if estudiante[0] == nombre:
            encontrado = True
            print(estudiante[1])
    if not encontrado:
        print("Estudiante no encontrado")
```

Visualizar lista

```
def visualizar(estudiantes):
    print("Lista de estudiantes".center(30, "#"))
    if len(estudiantes) == 0:
        print("No hay estudiantes registrados.")
    for e in estudiantes:
        print("Nombre: " + e[0] + ", nota: " + str(e[1]))
```

Mostrar menú

```
def mostrar_menu():
    print("Seleccione una opción: ")
    print("Comando 1: Agregar estudiante y nota"
          " '1&nombre_estudiante&nota'")
    print("Comando 2: Calcular promedio de los"
          " estudiantes en un momento dado")
    print("Comando 3: Ordenar estudiantes agregados por nombre")
    print("Comando 4: Consultar la nota de un estudiante"
          " '4&nombre_estudiante'")
    print("Comando 5: Visualizar")
    print("Comando 6: Salir")
```

Procesar Comandos

```
def procesar_comandos():
    bandera = True
    estudiantes = []
    comando = [0]
    while bandera or comando[0] != "6":
        bandera = False
        mostrar_menu()
        comando = input().split("&")
        print(comando[0])
        if comando[0] == "1":
            agregar_estudiante(estudiantes,
                               (comando[1], float(comando[2])))
        elif comando[0] == "2":
            promedio(estudiantes)
        elif comando[0] == "3":
            ordenar(estudiantes)
        elif comando[0] == "4":
            consultar(estudiantes, comando[1])
        elif comando[0] == "5":
            visualizar(estudiantes)
```