Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2115 – Programación como Herramienta para la Ingeniería (II/2017)

Actividad 7

Objetivos

- Aplicar técnicas de recursión y backtracking para resolver problemas complejos.

Entrega

- **Lenguaje a utilizar:** Python
- **Lugar:** Github
- **Hora:** 16:55
- **Desarrollo en parejas**

Introducción

Instrucciones

En esta actividad se desarrollarán cuatro problemas distintos, los cuales requieren distintos métodos y algoritmos de resolución. Los problemas están puestos en orden creciente de dificultad.

Problema 1: Listas falladas - Dificultad: Bastián (Lacayo)

Bastián acaba crear una nueva lista llamada `BastianList`, que no soporta los métodos `len` o `sum`. Por eso te pide que crees dos funciones recursivas (`largo` y `suma`) que calculan el largo y suma, respectivamente, de esta nueva lista. A continuación se muestra el código que debes copiar y completar:

```
class BastianList(list):
    def __init__(self, *args):
        for elem in args:
            self.append(elem)
    def __len__(self):
        raise ValueError("Metodo no permitido")

    def __iter__(self):
        raise ValueError("Metodo no permitido")

def largo(lista):
    # Completar
    pass
```

```
def suma(lista):
    # Completar
    pass

lista = BastianList(1,2,34,4,5,6,7,8)
print(suma(lista))
print(largo(lista))
```

Problema 2: Multiplicación de números grandes - Dificultad: Laika[1] (Perro de Bastián)

Actualmente, dispones de un computador con un particular problema, **no puede multiplicar números de más de 2 dígitos que sean diferentes a potencias de 10**. Por ejemplo, no es capaz de calcular $213 \cdot 213$, $101 \cdot 2$, $3 \cdot 367$, pero sí $2 \cdot 23$, $99 \cdot 25$ o $23 \cdot 1000$ (el segundo número es potencia de 10).

Es por ello que se Laika decidió darte un buen desafio: implementar un algoritmo que permita multiplicar números grandes usando esta desventaja. Para esto te indica que utilices el *algoritmo de Karatsuba* con una base B = 10 para descomponer de forma recursiva el número y lograr su multiplicación. Para comprobar que cumples esta misión, Laika te **prohibe ocupar * para multiplicar** y en cambio te entregó la siguiente función que multiplica emulando dicho computador.

```
def mul(a, b):
    if (b % 10 != 0 and b > 100) or (a % 10 != 0 and a > 100):
        raise ValueError("No puedo procesar esto")
    return a*b

print(mul(123123123,1231233312))
```

Problema 3: Vuelto - Dificultad: Antonio (Ayudante)

Actualmente Antonio (un supuesto amigo de Bastián) se encuentra trabajando como cajero en su parque de diversiones y tiene un gran dilema: le cuesta **calcular la cantidad mínimas de monedas** que debe dar cuando entrega vuelto a los clientes. Es por esto que le solicitó a su supuesto amigo que te pidiera una función de backtracking que resuelva su problema.

Para resolver lo anterior, tu función recibirá un diccionario y un entero. El diccionario tendrá de *key* las diferentes monedas existentes en la caja del parque, de *value* la cantidad que se dispone de esa moneda, mientras que el entero corresponde al vuelto que se quiere dar. Como resultado, deberás entregar un diccionario donde las *keys* son las diferentes monedas y el *value* es la cantidad que se debe entregar de esa moneda. Debes tener precuación de no dar más o menos vuelto de lo solicitado y que la cantidad de monedas entregadas sea mínima[2]. Suponiendo que tu función se llama `dar_vuelto`, un posible código de esta actividad será:

```
monedas = {1000: 1, 2000: 2, 500: 3, 100: 8, 50: 2, 10:5}
vueltos = [3610, 2610, 3610, 4610, 5610]
for vuelto in vueltos:
    print(dar_vuelto(monedas, vuelto))
```

[1] Laika demoró 5 minutos en hacer este ejercicio, Bastián 10.

[2] Porque a nadie le gusta andar con un millón de monedas de 10, ¿cierto?

Problema 4: Ordenamiento - Dificultad: Hans (Difícil)

Un hospital en HansLandia tiene un curioso sistema para decidir las prioridades en su servicio de urgencia: no importa cuán grave está el sujeto en cuestión, sino solo su edad; es decir, las personas más jovenes serán atendidas primero. Lo curioso del hospital no termina en este sistema, pues su lista de espera está en forma de una lista ligada, donde la información contenida en cada nodo corresponde al nombre y la edad de las personas.

Se te pide entonces que ordenes esta lista ligada para que, dada una lista de espera cualquiera, se retorne una lista ligada de pacientes cuyas edades estén en orden creciente. La lista ligada que recibirás de input corresponde a la estructura que aparece en el material de clases. **Importante**: está prohibido cambiar la estructura de datos a ordenar.

Política de Integridad Académica

Los alumnos de la Escuela de Ingeniería deben mantener un comportamiento acorde al Código de Honor de la Universidad:

"Como miembro de la comunidad de la Pontificia Universidad Católica de Chile me comprometo a respetar los principios y normativas que la rigen. Asimismo, prometo actuar con rectitud y honestidad en las relaciones con los demás integrantes de la comunidad y en la realización de todo trabajo, particularmente en aquellas actividades vinculadas a la docencia, el aprendizaje y la creación, difusión y transferencia del conocimiento. Además, velaré por la integridad de las personas y cuidaré los bienes de la Universidad."

En particular, se espera que mantengan altos estándares de honestidad académica. Cualquier acto deshonesto o fraude académico está prohibido; los alumnos que incurran en este tipo de acciones se exponen a un procedimiento sumario. Ejemplos de actos deshonestos son la copia, el uso de material o equipos no permitidos en las evaluaciones, el plagio, o la falsificación de identidad, entre otros. Específicamente, para los cursos del Departamento de Ciencia de la Computación, rige obligatoriamente la siguiente política de integridad académica en relación a copia y plagio: Todo trabajo presentado por un alumno (grupo) para los efectos de la evaluación de un curso debe ser hecho individualmente por el alumno (grupo), sin apoyo en material de terceros. Si un alumno (grupo) copia un trabajo, se le calificará con nota 1.0 en dicha evaluación y dependiendo de la gravedad de sus acciones podrá tener un 1.0 en todo ese ítem de evaluaciones o un 1.1 en el curso. Además, los antecedentes serán enviados a la Dirección de Docencia de la Escuela de Ingeniería para evaluar posteriores sanciones en conjunto con la Universidad, las que pueden incluir un procedimiento sumario. Por "copia" o "plagio" se entiende incluir en el trabajo presentado como propio, partes desarrolladas por otra persona. Está permitido usar material disponible públicamente, por ejemplo, libros o contenidos tomados de Internet, siempre y cuando se incluya la cita correspondiente.