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

IIC2115 – Programación como Herramienta para la Ingeniería (I/2021)

Actividad Práctica 2

Objetivos

- Evaluar la utilización adecuada de algoritmos y técnicas de programación, en conjunto con estructuras de datos, para resolver problemas.

Entrega

- **Lenguaje a utilizar:** Python 3.6 o superior
- **Lugar:** repositorio privado en GitHub. Recuerde incluir todo en una carpeta de nombre **A2**.
- **Entrega: lunes 26 de abril a las 18:30 hrs.**
- **Formato de entrega (ES IMPORTANTE EL NOMBRE DEL ARCHIVO):**
  - Archivo python notebook (**A2.ipynb**) con la solución de los problemas y ejemplos de ejecución. Utilice múltiples celdas de texto y código para facilitar la revisión de su laboratorio.
  - Archivo python (**A2.py**) con la solución de los problemas. Este archivo solo debe incluir la definición de las funciones, utilizando exactamente los mismos nombres y parámetros que se indican en el enunciado, y la importación de los módulos necesarios. Puede crear y utilizar nuevas funciones si lo cree necesario. No debe incluir en este archivo ejemplos de ejecución ni la ejecución de dichas funciones. **No deje instancias del método `print()` en el archivo .py**. Si desea mantener ejecuciones o prints en el archivo **.py** estos **DEBEN** estar dentro de un bloque main.
  - Todos los archivos deben estar ubicados en la carpeta **A2**. No se debe subir ningún otro archivo a la carpeta.

- **NO SE ADMITEN ENTREGAS FUERA DE PLAZO**
- **Entregas con errores de sintaxis y/o que generen excepciones serán calificadas con nota 1.0.**

Introducción

Con el fin de evaluar el uso de estructuras de datos y técnicas de programación, en esta actividad práctica deberá desarrollar 4 ejercicios. Para cada uno, no solo es fundamental utilizar alguna(s) de las técnicas y/o estructuras cubiertas en el material del curso, sino que además hacerlo de manera adecuada.

Corrección

Todos los problemas tienen el mismo puntaje (**1,5 ptos.**) Para la corrección, se revisarán dos ítems por problema. El primero serán las estructuras y técnicas utilizadas para resolverlos. Estos deben ser adecuados al problema y cumplir con los requisitos de complejidad indicados en el enunciado. De este modo, es importante que comenten correctamente su código para que sea más sencilla la corrección. Recuerde que debe utilizar las estructuras de datos y algoritmos adecuados a cada problema. Además, se evaluará el orden de su trabajo. Este ítem tendrá un valor de **1,0 pto.** por problema.

El segundo ítem será la correctitud de los resultados, que será verificada mediante un revisor automático. Por cada problema se probarán distintos tamaños y valores de entrada, y para cada uno de estos se evaluará su correctitud. En particular, por cada tamaño de input probado, el resultado debe ser correcto para todas las instancias para obtener el puntaje. Este ítem tendrá un valor de **0,5 ptos.** por problema.

Ejercicio 1: Inversión de *strings*

Dado un *string* de largo arbitrario, escriba un programa que retorne el *string* con sus carácteres en orden invertido. Empaque el algoritmo en una función que reciba como argumento un *string* y retorne un *string*. La solución debe tener una complejidad no mayor a $\mathcal{O}(n)$, donde n es el largo del *string*, y **no debe** utilizar funciones propias de Python para invertir el *string* de manera directa. Un ejemplo de ejecución del algoritmo es el siguiente:

Código

```
string_original = "programación_como_herramienta_para_la_ingeniería"
string_inverso = invertir_string(string_original)
print(string_inverso)
```

Salida

```
aíreinegni_al_arap_atneimarreh_omoc_nóicamargorp
```

Ejercicio 2: Producto de máximo valor absoluto

Dada una lista de números enteros de tamaño arbitrario, escriba un programa que encuentre el producto de máximo valor absoluto entre dos elementos de esta. Empaque el algoritmo en una función que reciba como argumento una lista de enteros y retorne un número natural (representado en python como un int), que indique el producto de máximo valor absoluto. La solución debe tener una complejidad no mayor a $\mathcal{O}(n)$, donde n es el largo de la lista. Un ejemplo de ejecución del algoritmo es el siguiente:

Código

```
lista = [-10, -3, 4, 6, -2]
max_prod = maximo_producto_absoluto(lista)
print(max_prod)
```

Salida

```
60
```

Ejercicio 3: Raíz cuadrada eficiente

Dado un número natural, encuentre de manera eficiente la raíz cuadrada de este número, y en caso que el número no sea un cuadrado perfecto, retorne el piso de la raíz. Empaque el algoritmo en una función que reciba como argumento un número natural y retorne un número natural (representados en python como

int), que indique la raíz cuadrada, o el piso de esta. La solución debe tener una complejidad no mayor a $\mathcal{O}(log\ n)$, donde n es el valor del número al que se le busca calcular la raíz, y **no debe** utilizar la función de raíz cuadrada de Python. Un ejemplo de ejecución del algoritmo es el siguiente:

Código

```
numero = 17
raiz = calcular_raiz(numero)
print(raiz)
```

Salida

```
4
```

Ejercicio 4: Caminos en un grafo

Dada un grafo dirigido, verifique si es posible llegar a todos los nodos de este, partiendo de cualquier otro nodo. Empaque el algoritmo en una función que reciba como argumento una lista de listas, que representa la matriz de adyacencia del grafo, y retorne un valor booleano, indicando si es posible llegar a todos los nodos del grafo, partiendo de cualquier otro nodo. La solución debe tener una complejidad no mayor a $\mathcal{O}(N + A)$, donde N es el número de nodos del grafo y A el número de arcos. Un ejemplo de ejecución del algoritmo es el siguiente:

Código

```
grafo = [[0,0,0,0,1],[1,0,1,0,0],[0,1,0,0,1],[0,1,1,0,0],[0,0,0,1,0]]
se_puede = caminos_desde_todos(grafo)
print(se_puede)
```

Salida

```
True
```

Política de Integridad Académica

"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.