**Taller Final**

**Estructura de Datos y Algoritmos**

**Cristian Camilo Zapata García**

**Julián Andrés Mazo Zapata**

**Gloria Stella Sepúlveda Cossío**

**Universidad EAFIT**

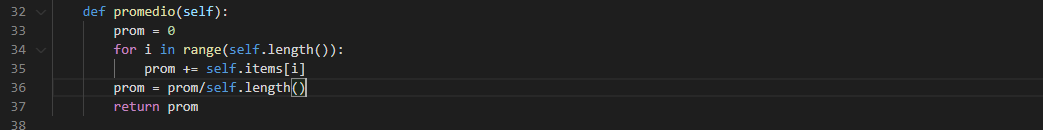
**2020**

1. **R/:**

La lista en Python son variables que almacenas Arrays, internamente cada posición puede ser de un tipo de datos distintos así que si es posible crear listas enlazadas cuyos nodos sean de distinto tipo, ya que, según lo que se ha visto implementando listas enlazadas en Python, a la hora de crear un nodo lo importante no es el tipo de dato que recibe, sino, el hecho de tener un dato y un apuntador; por esta razón, si es algo que es posible hacerse, pero que sigue siendo no recomendado hacer, por cuestiones de orden e implementación de métodos que use la lista.

1. **R/:**

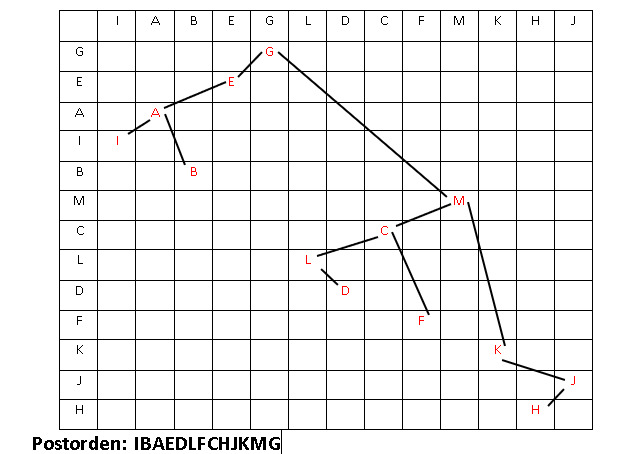
Hay muchas maneras posibles de sacar el promedio, este fue hecho basado en la primera implementación de pila que se hizo, donde la pila es una lista, otra manera como se puede hacer si la pila es una lista simplemente enlazada es usando en el método ‘while pila != self.novacia()’ llamando al método ‘novacia()’ para verificar que la pila siga con elementos, mientras se va llamando el método ‘desapilar()’ y se va sumando 1 por 1 al promedio, para después dividirlo.

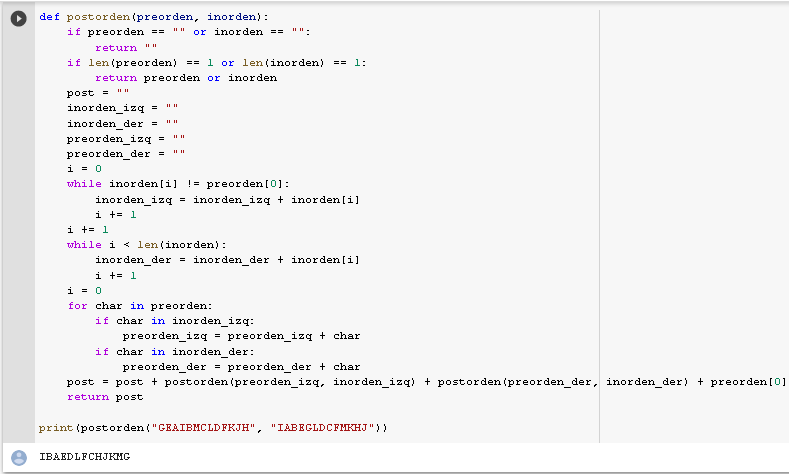


Se deja el link al colab totalmente funcional con una pila ya declarada y mostrando dos ejemplos.

<https://colab.research.google.com/drive/18_QmReIiyZwvgrkxw8LJY5CxDUPrTsi_?usp=sharing>

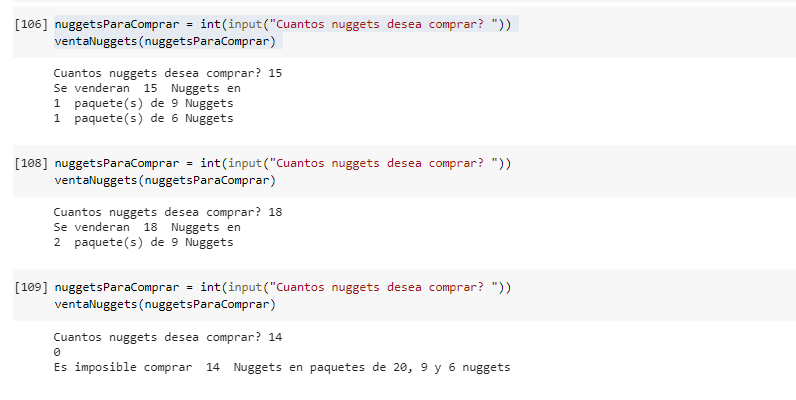
1. **R/:**



**4. R/:**

Este es el código totalmente funcional para la venta de Nuggets de pollo en McDonald's; el código puede ser optimizado si se divide en métodos, pero, así como está funciona en Colab





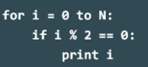
Se adjunta un link al Colab totalmente funcional, con comentarios:

<https://colab.research.google.com/drive/1VK6ptMFE6dy-mKfAgI5EYUB-5rVrcPqQ?usp=sharing>

1. **R/:**

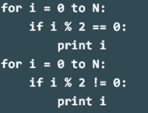
El cálculo de la complejidad da como resultado

5.1

.

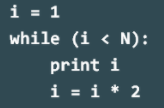
O(n)

5.2.



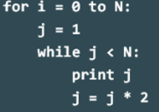
O(n+n) = O(n)

5.3.



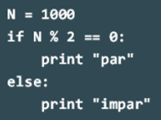
O(log(n))

5.4.



O(nlog(n))

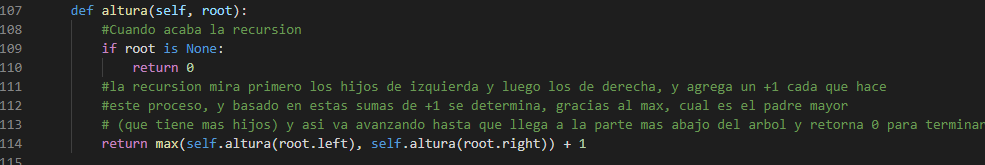
5.5.



O(1)

1. **R/:**

El método que calcula la altura máxima, basado en el código de árbol visto en clase, implementado de forma recursiva es el siguiente:



En Colab se encuentra el código para este método y para crear un árbol simplificado, dejando solo lo necesario para este ejercicio (método de insertar, visualizar in-order, borrar y, por último, mostrar la altura), se adjunta el link completamente funcional al Colab

<https://colab.research.google.com/drive/1ygXL8pzqGV5Albi3BWo5tmmqk2etrPbw?usp=sharing>

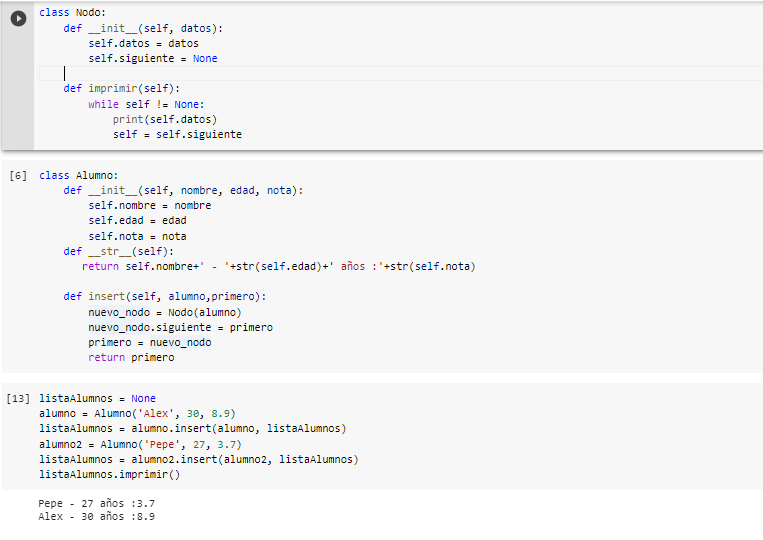
1. **R/=**

**7.1.** La estructura que se utilizó es muy similar a la de una lista simplemente enlazada, donde, para guardar los datos, está la clase alumno que guarda nombre, edad y nota y para guardar dichos datos está la clase Nodo que almacena el dato actual, y el dato del estudiante siguiente, es simplemente enlazada porque no guarda información del estudiante anterior sino únicamente del que sigue

**7.2.** El algoritmo presenta una complejidad de O(n)

**7.3**. En este código como optimización se decidió agregar un método ‘insert’ para no tener que repetir código escribiendo ‘nodo = Nodo(alumno)’ y tener que declararlo y asignar valores nuevos a los nodos cada vez; además se agregó un método imprimir a la clase código para que cuando se desee imprimir solo halla que llamarlo. Cabe mencionar que, obviamente se le pueden añadir mejoras, como un método que permita borrar un estudiante previamente ingresado, o un menú que pregunte que se desea hacer y que pida al usuario digitar los datos de los estudiantes que desea ingresar, pero como el objetivo del ejercicio es optimizar código, se evitó añadir cosas que puedan hacer la ejecución del programa más lenta.

Debajo de la imagen está el link al Colab totalmente funcional

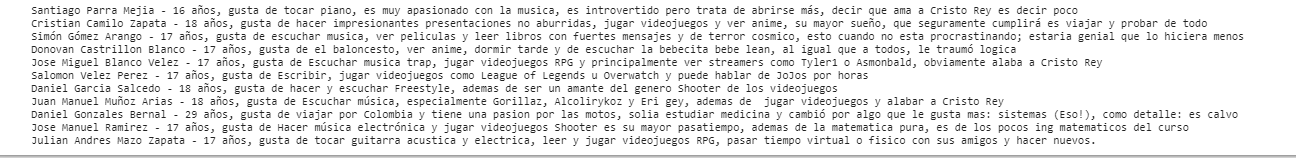


<https://colab.research.google.com/drive/12FTVvrrMQfC2zTi6ILZNJ8c3Gf5KhNFS?usp=sharing>

**7.4.** Después de las optimizaciones de código el algoritmo aun presenta una complejidad de O(n)

1. **R/=**

Se cambió el dato “Nota” por “Gustos” para guardar ahí los gustos o curiosidades de la persona que hizo la presentación, se agregaron a la lista en el orden que subieron sus presentaciones, y el resultado del print final fue:



Se adjunta el link al Colab totalmente funcional donde se agregan los estudiantes a la lista:

<https://colab.research.google.com/drive/1VB0I7hghIKwuWBk4a9SbPP5bsInTgsTv?usp=sharing>