**Taller Final - Diego Velásquez Varela- Simón Gómez Arango/202010004101-202010090010**

1. (10) Las listas enlazadas son por definición estructuras de datos homogéneas (es decir, todos sus nodos almacenan elementos del mismo tipo y estructura). ¿Eso es totalmente cierto?, es decir, ¿Sería posible tener una lista que almacene elementos diferentes en cada uno de sus nodos (tanto en tipo como en estructura)? Justifique su respuesta

La respuesta a este punto debe darse en dos partes.

**¿Es totalmente cierto?** – Sí, puesto que la definición de la lista enlazada nos indica que son estructuras de datos homogéneas, finitas y ordenadas. Su ventaja radica en que, a diferencia de los vectores convencionales, el orden de los elementos enlazados puede ser diferente al orden de almacenamiento en memoria o disco, lo que hace posible que el orden del recorrido de la lista sea diferente al de almacenamiento.

Ahora bien, **¿Sería posible tener una lista que almacene elementos diferentes en cada uno de sus nodos (tanto en tipo como en estructura)? –** podríamos crear una estructura parecida a las listas enlazadas que tenga conexiones numérico-categóricas, implementando la lista enlazada en forma de clases guardando los dos tipos de atributos. En esta variación serían los nodos los que están conectados entre sí, lo que acarrea implicaciones en las operaciones que se harían con la estructura (como lo son el ordenamiento o la búsqueda)

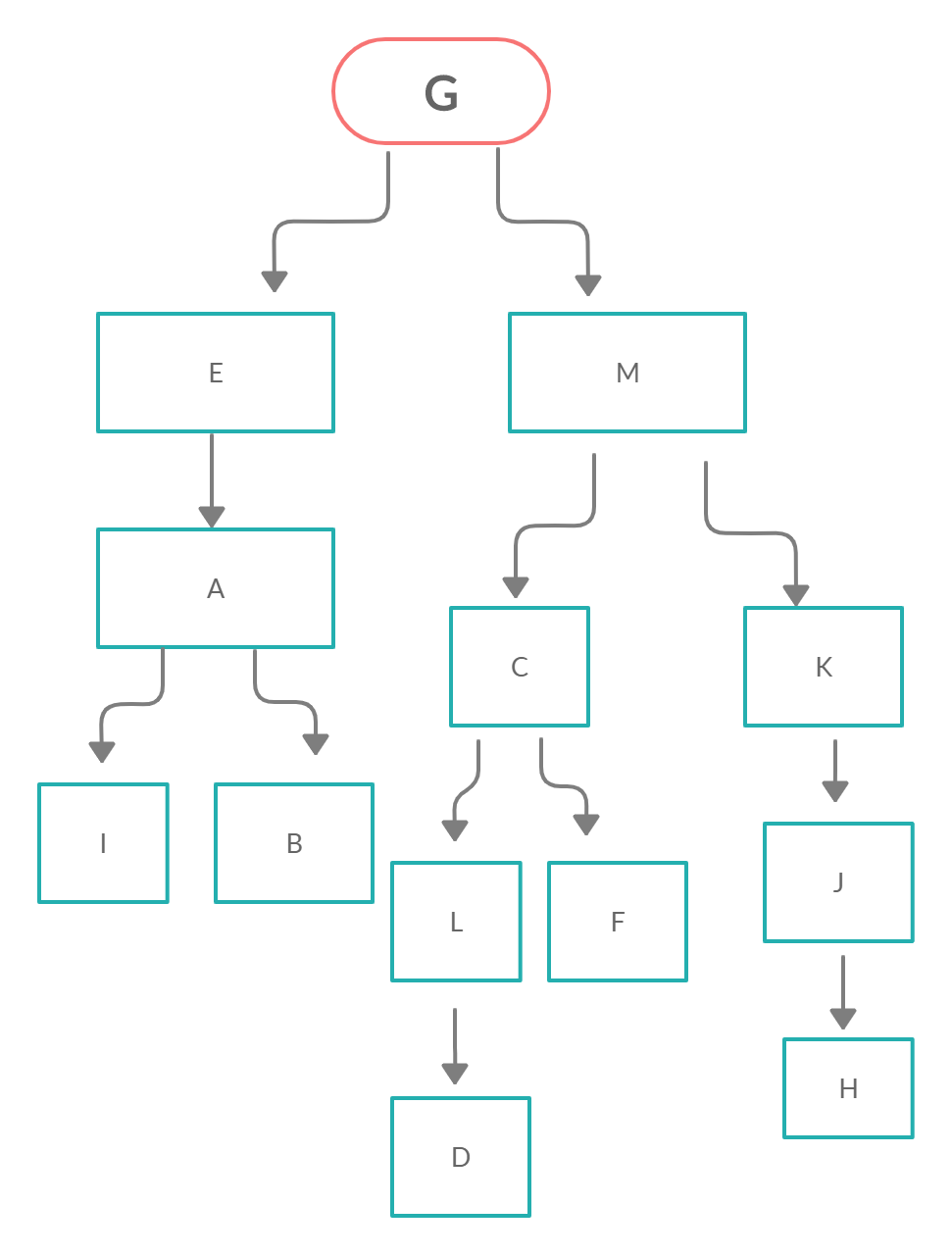
1. (10) Dada una pila, se desea conocer el promedio de los elementos que ella almacena. Como restricción la pila puede ser recorrida una sola vez.

**Adjunto a la carpeta se encuentra punto2.py**

1. (15) El recorrido en preorden de un determinado árbol binario es: GEAIBMCLDFKJH y en inorden IABEGLDCFMKHJ.

Resolver:

3.1 Dibujar el árbol binario. (al crear el archivo en GitHub deberá mostrar la figura que quedaría puede ser en Word o Excel)



3.2 Dar el recorrido en postorden.

- IBAEDLFCHJKMG

3.3 Diseñar una función para dar el recorrido en postorden dado el recorrido en preorden e inorden y escribir un programa para comprobar el resultado del apartado anterior

**Adjunto a la carpeta se encuentra punto3.py**

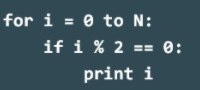
1. (10) En McDonald’s se pueden comprar Nuggets de pollo en paquetes de 6, 9 o 20 unidades.

Escribir un algoritmo en Python que reciba un parámetro entero y decida si es posible o no comprar ese numero de nuggets.

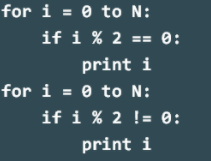
Ejemplos

* es posible comprar 15 nuggets (un paquete de 6 y uno de 9)
* es posible comprar 18 nuggets (3 paquetes de 6 o 2 paquetes de 9)
* **Es imposible** comprar 14 nuggets

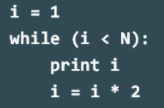
R/= Lo seguiremos intentando, hasta ahora no reconocemos bien la lógica del ejercicio. Estará adjunto en punto4.py si lo logramos ☹

1. (5) Indicar con notación “Big -O”, la complejidad algorítmica (respecto al tiempo) de cada fragmento de código a continuación
   1. 

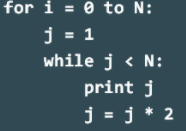
O(n)

* 1. 

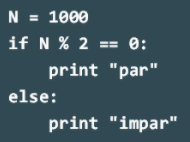
O(n)

* 1. 

O (log n)

* 1. 

O (n log n)

* 1. 

O (1)

1. (10) Escribe un programa que calcule la altura máxima de un árbol binario. Si está vacío, la altura se considera 0, y si solo hay la raíz se considera 1.

**Adjunto a la carpeta se encuentra punto6.py**

1. (10)En <https://github.com/gsepulv1/ST0245-08/blob/master/taller_Final/estudiantes.py> se encentra un algoritmo que almacena información de estudiantes.
   1. que estructura se utilizó para almacenar los datos de los estudiantes?

R= Lista simplemente enlazada

* 1. que complejidad tiene este algoritmo

R= O(n)

* 1. Realice las optimizaciones de código a este algoritmo

R= Revisar punto7.py, realizamos una lista con los estudiantes, y en for recorrimos la lista y desde allí mismo la impresión, ahorrándonos el ciclo del print

* 1. Que complejidad tiene después de la mejora realizada en 7.3

El código sigue siendo de O(n) pero se simplifica mucho más el funcionamiento.

1. (30) Implemente las modificaciones necesarias al algoritmo anterior y almacene e imprima la información de sus compañeros de curso empleando las presentaciones que se encuentran en el grupo de teams

**Adjuntado en punto8.py**