

Auxiliar 5 - Listas Enlazadas y Árboles

Profesores: Jérémy Barbay
Patricio Poblete, Nelson Baloian
Auxiliares: Felipe Lizama, Franco Sanguinetti,
Matías Ramírez, Sven Reisenegger.

P1. Lista enlazada circular

Si se tiene una lista enlazada y se une el último nodo con el primero, entonces se tiene una lista enlazada circular.

- Defina una clase `NodoLista`, que contenga un campo de información capaz de almacenar un *entero* y una referencia al nodo siguiente. El constructor de esta clase debe recibir un número entero y un objeto `NodoLista` que apunte al nodo siguiente.
- Defina una clase `ListaCircular` que represente una lista enlazada de números enteros, que tenga como atributos una referencia al primer elemento y la cantidad de elementos en la lista.
- Escriba un método `largo(self)` que retorne un entero con el largo de la lista (es decir, cuantos elementos contiene).
- Defina el método `obtener(self, i)` que retorne el *i*-ésimo elemento de la lista. Como la lista es circular, siempre se puede encontrar un *i*-ésimo elemento (a menos que esté vacía).
- Cree el método `existe(self, num)` que retorne la posición del elemento en la lista y si no está retorne -1.
- Cree el método `eliminar(self, i)` que elimine el *i*-ésimo elemento de la lista y retorne el siguiente nodo al eliminado.

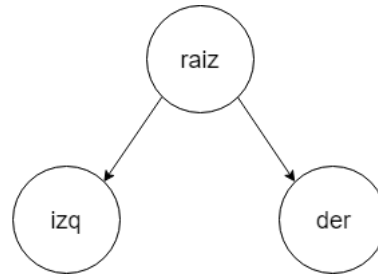
P2. Josephus

Los niños de la antigüedad no conocían los videojuegos y cuando estaban aburridos se veían obligados a jugar juegos con números. Un niño llamado Josephus inventó el siguiente juego: Se sentaban todos en un círculo y uno de los niños decía un número entero k , luego, a partir de la posición del niño, se empezaba a contar desde 0 en voz alta y el niño que decía k quedaba eliminado.

Luego, partiendo del siguiente al que se eliminó se volvía a contar hasta k , hasta que solo le quedaba un niño en el círculo, que era el ganador. Usando la lista circular programada en el punto anterior, ayude a los niños a resolver quién ganaría sin la necesidad de que jueguen.

Programa una función `josephus(lista_circular, k)` que recibe una lista circular y el entero k y retorne el último elemento sobreviviente del juego. (Asumiendo que los niños se pueden representar como una lista circular de enteros)

P3. Árboles binarios



Los árboles binarios son una estructura de datos recursiva en que cada nodo se compone de 3 partes: una llave o valor, un hijo derecho y un hijo izquierdo. Cada hijo es a su vez un árbol binario, por lo que la estructura puede crecer infinitamente. El “caso base” es un árbol vacío representado por `None`.

- Implemente la clase `ArbolBinario` que almacena un valor entero en su raíz y recursivamente a otros árboles en sus hijos. Cree un constructor:

```
– def __init__(self, x, izq=None, der=None)
```

Donde `x` corresponde al entero a almacenar como llave, `izq` corresponde al árbol que hay hacia la izquierda y `der` al árbol que hay hacia la derecha. Utilizando la estructura anterior implemente soluciones a los siguientes problemas:

- Obtener el número de nodos.
- Obtener el máximo valor.
- Obtener la suma de todos los valores.
- Obtener el número de “pisos” que tiene.
- Un Árbol Binario de Búsqueda (ABB) es un árbol binario que cumple la siguiente propiedad: todos los nodos a la izquierda de la raíz son menores a la raíz y todos los nodos a la derecha son mayores, esta propiedad se cumple recursivamente para sus hijos. Implemente un método que chequee que un árbol binario sea un ABB.