Auxiliar 6 - Pilas y Colas

Profesores: Iván Sipiran
Nelson Baloian
Patricio Poblete

Auxiliares: Alonso Almendras, Albani Olivieri
Vicente Olivares, Ricardo Valdivia
Sebastián Acuña, Martín Paredes

P1. Chequeando secuencias de paréntesis (Ejercicio 5.1)

Decimos que una secuencia de paréntesis se encuentra balanceada si todo paréntesis que se abre tiene a su pareja correspondiente que lo cierra, sin que sobre ninguno. Por ejemplo, la secuencia "{[(){}]}" está balanceada mientras que ")()[]" y "[{()]}" no lo están.

Si únicamente trabajáramos con un tipo de paréntesis, por ejemplo los redondos "(" y ")", chequear que una secuencia de paréntesis se encuentra balanceada es sencillo: basta llevar un contador de paréntesis abiertos y se chequea que éste nunca sea negativo y que al final de la secuencia quede en cero. No obstante, el problema se complica cuando hay varios tipos de paréntesis permitidos, pues se debe verificar que cada paréntesis que cierra va en correspondencia con su tipo de paréntesis que abre.

Utilizando la clase Pila definida en el apunte, implemente una función `chequea_balance(s)`, que verifique que un string s compuesto de paréntesis se encuentra balanceado. Dentro de la función defina cuáles son los abre paréntesis y cierra paréntesis permitidos.

Implementación de Pila:

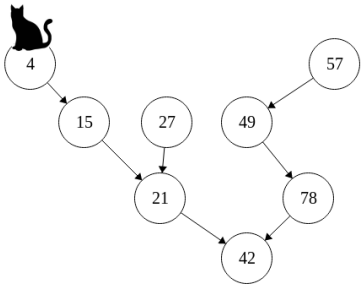
```
class Pila:
  def __init__(self):
    self.s=[]
  def push(self,x):
    self.s.append(x)
  def pop(self):
    assert len(self.s)>0
    return self.s.pop()
  def is_empty(self):
    return len(self.s)==0
```

Tests de prueba:

```
print(chequea_balance("(()())"))  # debe retornar True
print(chequea_balance("{([]{()})}")) # debe retornar True
print(chequea_balance("{([]()})}")) # debe retornar False
print(chequea_balance("{<{<>}>}")) # debe retornar True
print(chequea_balance("{<{<>>}>}")) # debe retornar False
```

P2. Gatito en un árbol

Al gatito de Alan le gusta trepar al árbol binario de su jardín. El problema es que todavía no sabe cómo bajarse de ahí, así que siempre tienen que buscarlo. Debido a lo frondoso del árbol, no tiene más remedio que buscarlo rama por rama (nodo por nodo). Como Alan quiere subir lo menos posible al realizar la búsqueda por temor a caer, cada vez que trepa una rama él quiere buscar todos los nodos en ese nivel (esto es una Búsqueda a lo ancho, BFS). Tu objetivo es ayudar a Alan a salvar su gatito del árbol.



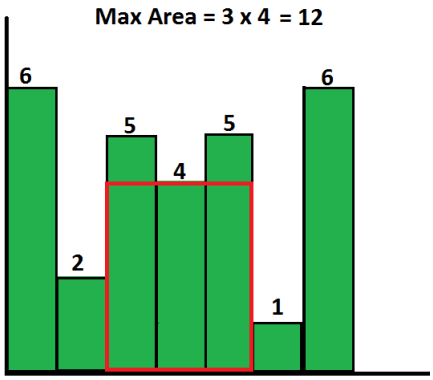
Los nodos del árbol se definen en la siguiente clase:

```
class NodoG:
    def __init__(self, ID_nodo, gatito, izq=None, der=None):
        self.gatito = gatito # True si está el gatito, False si no
        self.id = str(ID_nodo) + '' if gatito else str(ID_nodo)
        self.izq = izq
        self.der = der
```

Para bajar al gatito del árbol, utilice la clase Cola vista en clase, y cree un programa que muestre en pantalla los valores de los nodos que ha recorrido Alan hasta encontrar a su gatito, de izquierda a derecha en cada nivel. Por ejemplo, en el árbol de la figura, la salida sería `42 21 78 15 27 49 4`.

P3. Máxima Área contenida

Encuentre el área rectangular más grande posible en un histograma dado, donde el rectángulo más grande puede estar formado por varias barras contiguas. Para simplificar, suponga que todas las barras tienen el mismo ancho y el ancho es de 1 unidad. Por ejemplo, considere el siguiente histograma con 7 barras de altura 6, 2, 5, 4, 5, 1, 6. El rectángulo más grande posible es 12 (vea la figura a continuación, el rectángulo de área máxima se resalta en rojo)



a) Implemente una solución por fuerza bruta, es decir, pruebe iniciar desde cualquier barra y terminar en cualquier barra más a la derecha que la primera. Verifique que su solución es de orden $O(n^2)$, con n la cantidad de barras del histograma.

b) Implemente una solución más eficiente. Para ello haga uso de una pila que contenga la posición de la barra de manera creciente. Cuándo encuentre una barra que sea menor a la que está en la parte superior de la pila (*top*), quítela y calcule el área que obtenga utilizando la barra que está en *top* como la altura mínima. Por otra parte, si la barra es mayor a la que está en *top* sólo agréguela a la pila.