OBSERVACIONES DE LA PRÁCTICA

Simon Peña Alarcon, 202512907, s.penaa2@uniandes.edu.co

Manuel Santiago Figueroa, 202511697, m.figueroa@uniandes.edu.co

Juan Sebastian Chacón Ochoa , 202513196, J.chacono@uniandes.edu.co

Preguntas de análisis

1) ¿Qué relación encuentra entre el número de elementos en el árbol y la altura del árbol?
   Que la altura del árbol depende directamente de cómo se inserten las llaves. Si las llaves se insertan en un orden aleatorio, el árbol tiende a estar balanceado. Si las llaves se insertan en orden creciente o decreciente, el árbol se desbalancea.

En la funcion:

```
def height(my_bst):
    return height_tree(my_bst["root"])
```

Lo que hace es medir la altura recursivamente cont ando el camino más largo desde la raíz. Por tanto, a mayor número de elementos, mayor altura, aunque la relación exacta depende del orden de inserción. En resumen, el número de elementos influye en la altura: si el árbol está balanceado, la altura crece logarítmicamente, si está desbalanceado crece linealmente.

2) Si tuviera que responder esa misma consulta y la información estuviera en tablas de hash y no en un BST, ¿cree que el tiempo de respuesta sería mayor o menor? ¿Por qué?
   En tabla de hash las búsquedas, las inserciones y eliminaciones tienen un tiempo constante, pero en un árbol binario de búsqueda, esas mimas operaciones son logarítmicas si el árbol está balanceado o O(n) si no lo está. Por esto mismo consultar los datos en tabla de hash es mas rápido porque no necesita recorrer nodos y comparar llaves ordenadamente. Sin embargo, el BST hace consultas por rango si las llaves están ordenadas, lo que no hace una tabla de hash eficientemente.

3) ¿Qué operación del TAD se utiliza para retornar una lista con la información encontrada en un rango de fechas?
   La operación del árbol que nos permite retornar una lista con la información dentro de un rango de fechas es values(). Esta funciónobtiene primero todas las llaves del arbol con la función key_set() y luego usa values_range() para filtrar las llaves que están entre las fechas indicadas, así agregando los valores correspondientes a una lista enlazada. Values() regresa los elementos del árbol si sus llaves que son las fechas están en un rango especificado.

```python
def values(my_bst, key_initial, key_final):
    if my_bst["root"] is None:
        return lt.new_list()
    else:
        all_keys = key_set(my_bst)
        return values_range(my_bst, key_initial, key_final, all_keys)
```