紅黑樹

題目一：驗證紅黑樹是否合法

題目描述

給你一棵二元樹，其節點結構如下（語言僅供參考）：

```
class TreeNode:
    def __init__(self, val, color, left=None, right=None):
        self.val = val          # int
        self.color = color      # 'R' or 'B'
        self.left = left
        self.right = right
```

請實作函式：

```
class Solution:
    def isValidRedBlackTree(self, root: Optional[TreeNode]) -> bool:
        pass
```

判斷該樹是否為一棵**合法的紅黑樹**。紅黑樹需同時滿足以下性質：

1. 每個節點要嘛是紅色，要嘛是黑色。
2. 根節點必須是黑色。
3. 所有葉節點（NIL / 空節點）視為黑色。
4. 如果一個節點是紅色，則它的兩個子節點必須都是黑色。
5. 對任一節點，從該節點到其所有後代 NIL 節點的路徑上，黑色節點數量必須都相同（黑高一致）。

若滿足以上條件，回傳 `true`，否則回傳 `false`。

範例 1：

```
輸入：
   root = [10(B), 5(R), 15(R)]

結構：
       10(B)
       /   \
    5(R)  15(R)

輸出：
false
```

解釋：

兩個紅色節點 (5, 15) 皆為紅色且皆為根的子節點，但紅色節點可以有黑色父節點，這裡沒有違反「紅節點不可有紅父節點」，然而若其下層結構導致黑高不一致或其他性質不符，仍需判定為 false（實作時需完整檢查所有條件；此範例可自行設計更完整樹結構）。

範例 2：

```
輸入：
   root = [10(B), 5(R), 15(B), 2(B), 7(B)]

結構：
        10(B)
       /     \
    5(R)    15(B)
    /  \
  2(B) 7(B)

輸出：
true
```

提示

- 你可以撰寫一個遞迴函式同時計算「是否有效」與「每條路徑黑節點數」，若發現不一致即可提早返回。
- NIL 子節點視為黑色，計算黑高時需納入。

限制條件

- 節點數量 `1 <= N <= 10^5`
- 節點值 `10^9 <= val <= 10^9`
- 顏色只會是 `'R'` 或 `'B'`

題目二：設計支援插入與搜尋的紅黑樹

題目描述

請你設計一個支援以下操作的資料結構 `RedBlackTree`：

- `insert(x)`：將整數 `x` 插入紅黑樹中。如果 `x` 已存在，可以選擇忽略或自行定義規則（例如允許重複或維護計數），在題目中假設不會測重複值情況。
- `search(x)`：回傳布林值，表示整數 `x` 是否存在於樹中。

在每次插入後，你必須**維持紅黑樹的所有性質**（透過旋轉與重新著色）。

實作介面（語言範例）：

```
class RedBlackTree:

    def __init__(self):
        pass

    def insert(self, val: int) -> None:
        pass

    def search(self, val: int) -> bool:
        pass
```

線上測試系統會給你一連串操作：

```
operations = ["RedBlackTree", "insert", "insert", "search", "insert", "search"]
values     = [[],             [10],     [5],      [10],     [20],     [15] ]
```

你需要回傳對應的輸出：

```
[null, null, null, true, null, false]
```

範例 1：

```
輸入：
operations = ["RedBlackTree", "insert", "insert", "insert", "search", "search"]
values     = [[],             [10],     [5],      [15],     [5],      [20]]

輸出：
[null, null, null, null, true, false]
```

解釋：

- 建立一棵空的紅黑樹
- 插入 10、5、15，經過一連串旋轉與著色後，樹仍為合法紅黑樹
- `search(5)` 回傳 `true`
- `search(20)` 回傳 `false`

提示

- 你必須實作：
  - 左旋與右旋操作
  - 插入後的修正流程（Case 1/2/3：叔叔紅或黑）
- 測試資料不會檢查你輸出樹結構，只會檢查 `search` 的正確性，**但你仍必須維持紅黑樹性質**，否則在隱藏測資（大量插入後再搜尋）可能超時或錯誤。

限制條件

- 操作次數 `1 <= Q <= 10^5`

- 插入的整數 `10^9 <= x <= 10^9`
- 不保證操作順序中 `insert` 與 `search` 的比例
