

```

1  /*
2  给定二叉搜索树的根结点 root，返回 L 和 R（含）之间的所有结点的值的和。
3
4  二叉搜索树保证具有唯一的值。
5
6  示例 1:
7
8  输入: root = [10,5,15,3,7,null,18], L = 7, R = 15
9  输出: 32
10
11 示例 2:
12
13 输入: root = [10,5,15,3,7,13,18,1,null,6], L = 6, R = 10
14 输出: 23
15
16
17
18 提示:
19
20     树中的结点数量最多为 10000 个。
21     最终的答案保证小于 2^31。
22
23 来源：力扣（LeetCode）
24 链接：https://leetcode-cn.com/problems/range-sum-of-bst
25 著作权归领扣网络所有。商业转载请联系官方授权，非商业转载请注明出处。
26 */

```

分析:

- 二叉搜索树本身满足一些性质:当前节点左侧的值不大于当前节点的值，当前节点右侧的值不小于当前节点的值，又题设中"二叉搜索树保证具有唯一的值。",所以性质中都取不到等号.
- 根据以上性质:可以使用递归方法先序遍历的方法遍历二叉搜索树中在[L,R]之间的元素，并累加返回即可.
- 同理使用栈结构辅助即可使用迭代方法遍历二叉搜索树.

方法一:C_递归遍历

```

1
2  /**
3   * Definition for a binary tree node.
4   * struct TreeNode {
5   *     int val;
6   *     struct TreeNode *left;
7   *     struct TreeNode *right;
8   * };
9   */
10
11  int ret_val = 0;
12
13
14  void __preOrderTravel(struct TreeNode* node , int L , int R)
15  {
16      if(node==NULL)

```

```

17     {
18         return ;
19     }
20     if(node->val < L)
21     {
22         __preOrderTravel(node->right,L,R);
23     }
24     else if(node->val > R)
25     {
26         __preOrderTravel(node->left,L,R);
27     }
28     else
29     {
30         ret_val += node->val;
31         __preOrderTravel(node->left,L,R);
32         __preOrderTravel(node->right,L,R);
33     }
34
35     return ;
36 }
37
38
39 int rangeSumBST(struct TreeNode* root, int L, int R)
40 {
41     ret_val = 0;
42     __preOrderTravel(root,L,R);
43     return ret_val;
44 }
45 }
46
47 /*
48 执行结果:
49 通过
50 显示详情
51 执行用时 :128 ms, 在所有 C 提交中击败了54.15% 的用户
52 内存消耗 :43.8 MB, 在所有 C 提交中击败了100.00%的用户
53 */

```

方法二:C++_借助栈迭代遍历

```

1
2  /**
3   * Definition for a binary tree node.
4   * struct TreeNode {
5   *     int val;
6   *     TreeNode *left;
7   *     TreeNode *right;
8   *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
9   * };
10  */
11  class Solution
12  {
13  public:
14      int rangeSumBST(TreeNode* root, int L, int R)
15      {
16          stack<TreeNode*> stn          ;

```

```

17         int ret_val = 0 ;
18         TreeNode* temp = NULL ;
19         if(root!=NULL)
20         {
21             stn.push(root);
22             while(!stn.empty())
23             {
24                 temp = stn.top();
25                 stn.pop();
26                 if(temp == NULL)
27                 {
28                     continue;
29                 }
30
31                 if(temp->val < L )
32                 {
33                     stn.push(temp->right);
34                 }
35                 else if(temp->val > R)
36                 {
37                     stn.push(temp->left);
38                 }
39                 else
40                 {
41                     ret_val += temp->val;
42                     stn.push(temp->right);
43                     stn.push(temp->left);
44                 }
45             }
46         }
47         return ret_val;
48     }
49 };
50
51
52 /*
53 执行结果:
54 通过
55 显示详情
56 执行用时 :276 ms, 在所有 C++ 提交中击败了28.21% 的用户
57 内存消耗 :41.1 MB, 在所有 C++ 提交中击败了91.02%的用户
58 */
59
60
61
62 class Solution
63 {
64 public:
65     int rangeSumBST(TreeNode* root, int L, int R)
66     {
67         stack<TreeNode*> stn ;
68         int ret_val = 0 ;
69         TreeNode* temp = NULL ;
70         if(root!=NULL)
71         {
72             stn.push(root);
73             while(!stn.empty())
74             {

```

```

75         temp = stn.top();
76         stn.pop();
77
78         if(temp->val < L )
79         {
80             if(temp->right)
81             {
82                 stn.push(temp->right);
83             }
84         }
85         else if(temp->val > R)
86         {
87             if(temp->left)
88             {
89                 stn.push(temp->left);
90             }
91         }
92         else
93         {
94             ret_val += temp->val;
95             if(temp->right)
96             {
97                 stn.push(temp->right);
98             }
99             if(temp->left)
100             {
101                 stn.push(temp->left);
102             }
103         }
104     }
105 }
106 return ret_val;
107 }
108 };
109
110
111 /*
112 执行结果:
113 通过
114 显示详情
115 执行用时 :276 ms, 在所有 C++ 提交中击败了28.21% 的用户
116 内存消耗 :41.1 MB, 在所有 C++ 提交中击败了87.11%的用户
117 */

```