给定一个二叉搜索树，编写一个函数 `kthSmallest` 来查找其中第 **k** 个最小的元素。

说明：

你可以假设 k 总是有效的，1 ≤ k ≤ 二叉搜索树元素个数。

示例 1:

```
输入：root = [3,1,4,null,2], k = 1
   3
  / \
 1   4
  \
   2
输出：1
```

示例 2:

```
输入：root = [5,3,6,2,4,null,null,1], k = 3
       5
      / \
     3   6
    / \
   2   4
  /
 1
输出：3
```

进阶：

如果二叉搜索树经常被修改（插入/删除操作）并且你需要频繁地查找第 k 小的值，你将如何优化 `kthSmallest` 函数?

思路，个人用了最蠢的方法去取数据，花的时间为f(n),先用中度遍历把树转化为有序数组，再返回数据。

代码：

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
 * };
```

```
 */
class Solution {
public:
    void dfs(TreeNode* root,vector<int> &nums){
        if(root==NULL){
            return;
        }
        if(root->left!=NULL){
            dfs(root->left,nums);
        }
        nums.push_back(root->val);
        if(root->right!=NULL){
            dfs(root->right,nums);
        }
    }
    int kthSmallest(TreeNode* root, int k) {
        vector<int> nums;
        dfs(root,nums);
        return nums[k-1];
    }
};
```

更新方法二，从最小的遍历，查找k个节点即可得到答案

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
 * };
 */
class Solution {
public:
    bool dfs(TreeNode* root,int &num,int &k,int &result){
```

```
        if(root==NULL){
            return false;
        }
        if(root->left!=NULL){
            if(dfs(root->left,num,k,result)){
                return true;
            }
        }
        num+=1;
        if(num==k){
            result=root->val;
            return true;
        }
        if(root->right!=NULL){
            if(dfs(root->right,num,k,result)){
                return true;
            }
        }
        return false;
    }
    int kthSmallest(TreeNode* root, int k) {
        int num=0;
        int result;
        dfs(root,num,k,result);
        return result;
    }
};
```