```
/*
给定 S 和 T 两个字符串，当它们分别被输入到空白的文本编辑器后，判断二者是否相等，并返回结果。 # 代表退格字符。
示例 1：
输入：S = "ab#c", T = "ad#c"
输出：true
解释：S 和 T 都会变成 “ac”。
示例 2：
输入：S = "ab##", T = "c#d#"
输出：true
解释：S 和 T 都会变成 “”。
示例 3：
输入：S = "a##c", T = "#a#c"
输出：true
解释：S 和 T 都会变成 “c”。
示例 4：
输入：S = "a#c", T = "b"
输出：false
解释：S 会变成 “c”，但 T 仍然是 “b”。

提示：

    1 <= S.length <= 200
    1 <= T.length <= 200
    S 和 T 只含有小写字母以及字符 '#'。
来源：力扣（LeetCode）
链接：https://leetcode-cn.com/problems/backspace-string-compare
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。
*/
```

分析:

- 退格键特殊处理，最直接的方法是利用栈这个数据结构，从头到尾遍历这个两个数组，不是退格键就压栈，是退格键就弹栈一个数据，最后比较两个栈是否完全相等.(方法一:C++)
- 可以直接用数组来模拟栈(方法二:C).

方法一:C++_辅助栈

```
class Solution
{
    public:
        bool backspaceCompare(string S, string T)
        {
            stack<char> scs;
            stack<char> sct;
            int     i           =   0       ;
            int     size        =   0       ;
```

```
bool    ret_val     =   true    ;

size = S.size();
for(i=0;i<size;i++)
{
    if(S.at(i)=='#')
    {
        if(!scs.empty())
        {
            scs.pop();
        }
    }
    else
    {
        scs.push(S.at(i));
    }
}

size = T.size();
for(i = 0;i<size;i++)
{
    if(T.at(i)=='#')
    {
        if(!sct.empty())
        {
            sct.pop();
        }
    }
    else
    {
        sct.push(T.at(i));
    }
}

do{
    if(scs.size() != sct.size())
    {
        ret_val = false;
        break;
    }

    while(!scs.empty())
    {
        if(scs.top()!=sct.top())
        {
            ret_val = false;
            break;
        }
        else
        {
            scs.pop();
            sct.pop();
        }
```

```
                }
            }while(0);
            return ret_val;
    }
};
/*
执行结果：
通过
显示详情
执行用时 :0 ms, 在所有 C++ 提交中击败了100.00% 的用户
内存消耗 :8.5 MB, 在所有 C++ 提交中击败了87.25%的用户
*/
```

方法二:C_数组栈

```
/*数组栈基础数据结构定义*/
typedef struct tag_nanmian_stack
{
    char arr[200];
    int  length;
}nanman_stack_t;

/*初始化栈结构体*/
void stack_init(nanman_stack_t* nanman_stack_s)
{
    nanman_stack_s->length = 0;
}

/*向栈内压入一个字符*/
void stack_push(nanman_stack_t* nanman_stack_s,char ch)
{
    nanman_stack_s->arr[nanman_stack_s->length] = ch;
    (nanman_stack_s->length)++;
}

/*出栈一个字符*/
void stack_pop(nanman_stack_t* nanman_stack_s)
{
    if(nanman_stack_s->length>0)
    {
        (nanman_stack_s->length)--;
    }
}

/*判断栈的内容是否一样*/
bool stack_isequal( nanman_stack_t* nanman_stack_s1 ,
                    nanman_stack_t* nanman_stack_s2
                  )
{
    bool ret_val    =   true    ;
    int  i          =   0       ;
    do{
```

```
        if(nanman_stack_s1->length != nanman_stack_s2->length)
        {
            ret_val = false;
            break;
        }

        for(i=0;i<nanman_stack_s1->length;i++)
        {
            if(nanman_stack_s1->arr[i]!=nanman_stack_s2->arr[i])
            {
                ret_val = false;
                break;
            }
        }
    }while(0);

    return ret_val;
}

bool backspaceCompare(char* S, char* T)
{
    nanman_stack_t nanman_stack_S;
    nanman_stack_t nanman_stack_T;
    int i = 0;

    stack_init(&nanman_stack_S);
    stack_init(&nanman_stack_T);
    while(S[i])
    {
        if(S[i]=='#')
        {
            stack_pop(&nanman_stack_S);
        }
        else
        {
            stack_push(&nanman_stack_S,S[i]);
        }
        i++;
    }

    i = 0;
    while(T[i])
    {
        if(T[i]=='#')
        {
            stack_pop(&nanman_stack_T);
        }
        else
        {
           stack_push(&nanman_stack_T,T[i]);
        }
        i++;
    }
```

```
    return stack_isequal(&nanman_stack_S,&nanman_stack_T);
}
/*
执行结果：
通过
显示详情
执行用时 :4 ms, 在所有 C 提交中击败了81.54%的用户
内存消耗 :6.7 MB, 在所有 C 提交中击败了89.47%的用户
*/
```

AlimyBreak
2019.08.07