3.1 栈

栈的基本概念

只允许在一端进行插入或删除操作的线性表　　先进后出

结构
- 栈顶（Top）：线性表允许进行插入删除的那一端
- 栈底（Bottom）：固定的，不允许进行插入和删除的另一端

卡特兰数　　n个不同元素进栈,出栈元素不同排列的个数为　　$\frac{1}{n+1}C_{2n}^{n}$

栈的基本操作
- InitStack (&S):初始化一个空栈S
- StackEmpty (S):判断一个栈是否为空，若栈S为空则返回true.否则返回falseo Push (&S, x):进栈,若栈S未满,则将x加入使之成为新栈顶
- Pop (&S， &x):出栈,若栈S非空,则弹出栈顶元素，并用x返回
- GetTop(S，&x)：读栈顶元素，若栈S非空，则用x返回栈顶元素
- DestroyStack (&S):销毁栈，并释放栈S占用的存储空间("&"表示引用调用)

在解答算法题时，若题干未做出限制，则可直接使用这些基本的操作函数

栈的顺序存储结构

顺序栈的实现

采用顺序存储的栈称为顺序栈，它利用一组地址连续的存储单元存放自栈底到栈顶的数据元素，同时附设一个指针(top)指示当前栈顶元素的位置

基本结构与操作
- 栈顶指针:S.top,初始时设置S.top=-1
- 栈顶元素:S.data[S.top]
- 进栈操作:栈不满时,栈顶指针先加1,再送值到栈顶元素
- 出栈操作:栈非空时,先取栈顶元素值,再将栈顶指针减1
- 栈空条件:S.top==-1
- 栈满条件:S.top==MaxSize-1;栈长:S.top+1

顺序栈的基本操作

初始化
```
void InitStack(SqStack &S){
   S.top=-1;                    //初始化栈顶指针
}
```

判栈空
```
bool StackEmpty(SqStack S){
   if(S.top==-1)                //栈空
      return true;
   else                         //不空
      return false;
}
```

进栈
```
bool Push(SqStack &S,ElemType x){
  if(S.top==MaxSize-1)          //栈满，报错
    return false;
  S.data[++S.top]=x;            //指针先加 1，再入栈
  return true;
}
```

出栈
```
bool Pop(SqStack &S,ElemType &x){
   if(S.top==-1)                //栈空，报错
      return false;
   x=S.data[S.top--];           //先出栈，指针再减 1
   return true;
}
```

读栈顶元素
```
bool GetTop(SqStack S,ElemType &x){
   if(S.top==-1)                //栈空，报错
      return false;
   x=S.data[S.top];             //x 记录栈顶元素
   return true;
}
```

共享栈

让两个顺序栈共享一个一维数组空间，将两个栈的栈底分别设置在共享空间的两端，两个栈顶向共享空间的中间延伸

基本原则
- 两个栈的栈顶指针都指向栈顶元素,top0=-1时0号栈为空,top1=MaxSize时1号栈为空
- 当两个栈顶指针相邻（topl-top0=1）时,判断为栈满
- 当0号栈进栈时top0先加1 再赋值,1号栈进栈时 top1先减1再赋值;出栈时则刚好相反

存取数据的时间复杂度均为O(1)

栈的链式存储结构

采用单链表实现，并规定所有操作都是在单链表的表头进行

优点
- 便于多个栈共享存储空间和提高其效率
- 且不存在栈满上溢的情况