5/6

设栈的顺序存储空间为 S(0:49) ，栈底指针 bottom=49 ，栈顶指针 top=30 （指向栈顶元素）。则栈中的元素个数为（20 ）

栈是一种特殊的线性表，它所有的插入与删除都限定在表的同一端进行。入栈运算即在栈顶位置插入一个新元素，退栈运算即是取出栈顶元素赋予指定变量。元素依次存储在单元 30 ： 49 中，个数为 20 。

单链表实现的栈，栈顶指针为Top(仅仅是一个指针)，入栈一个P节点时，其操作步骤为

栈的表示在不同的书上一般有两种表达方式： 1.top指针指向栈顶元素，这样下一个出栈的元素就是top指向的结点元素；此时入栈操作为p->next=Top; Top=p; 2.top指向下一个可以入栈的位置，这样就相当于top是带头结点的链表中的头节点，然后入栈的话就要插入到top的下一个节点位置；此时入栈操作为p->next=Top->next;Top->next=p;

有六个元素6，5，4，3，2，1 的顺序进栈，问下列哪一个不是合法的出栈序列？

3 4 6 5 2 1

后进先出，34进的时候56都在了不能6先出

5/7

new 创建对象时，对象的内存和指向对象的指针分别分配在 堆区，栈区

中缀表达式 A+(B+C)\*D 的后缀表达式为 ABC+D\*+、

1. 先将中缀表达式加括号：(A + ((B + C) \* D))；

2. 再把运算符移到括号后面（前缀移到前面）：(A ((B C)+ D)\*)+；

3. 把括号去掉：ABC+D\*+。

对中缀表达式 3\*2^(4+2\*2-6\*3)-5 进行求值，求值过程中当扫描到6时，对象栈和算符栈分别为（），其中^为乘幂

A picture containing text, screenshot, font, number

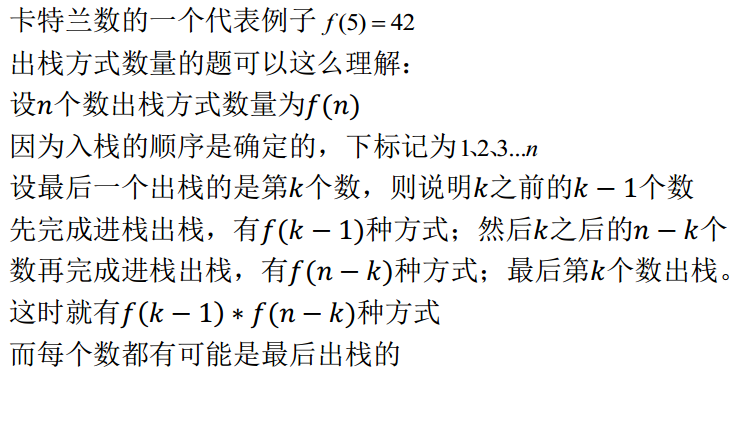
Description automatically generated

栈的大小是固定的，堆的大小受限于系统中有效的虚拟内存

栈的空间由系统决定何时释放，堆需要自己决定何时去释放

堆的使用容易产生碎片，但是用起来最方便

若一序列进栈顺序为e1,e2,e3,e4,e5,问存在多少种可能的出栈序列42

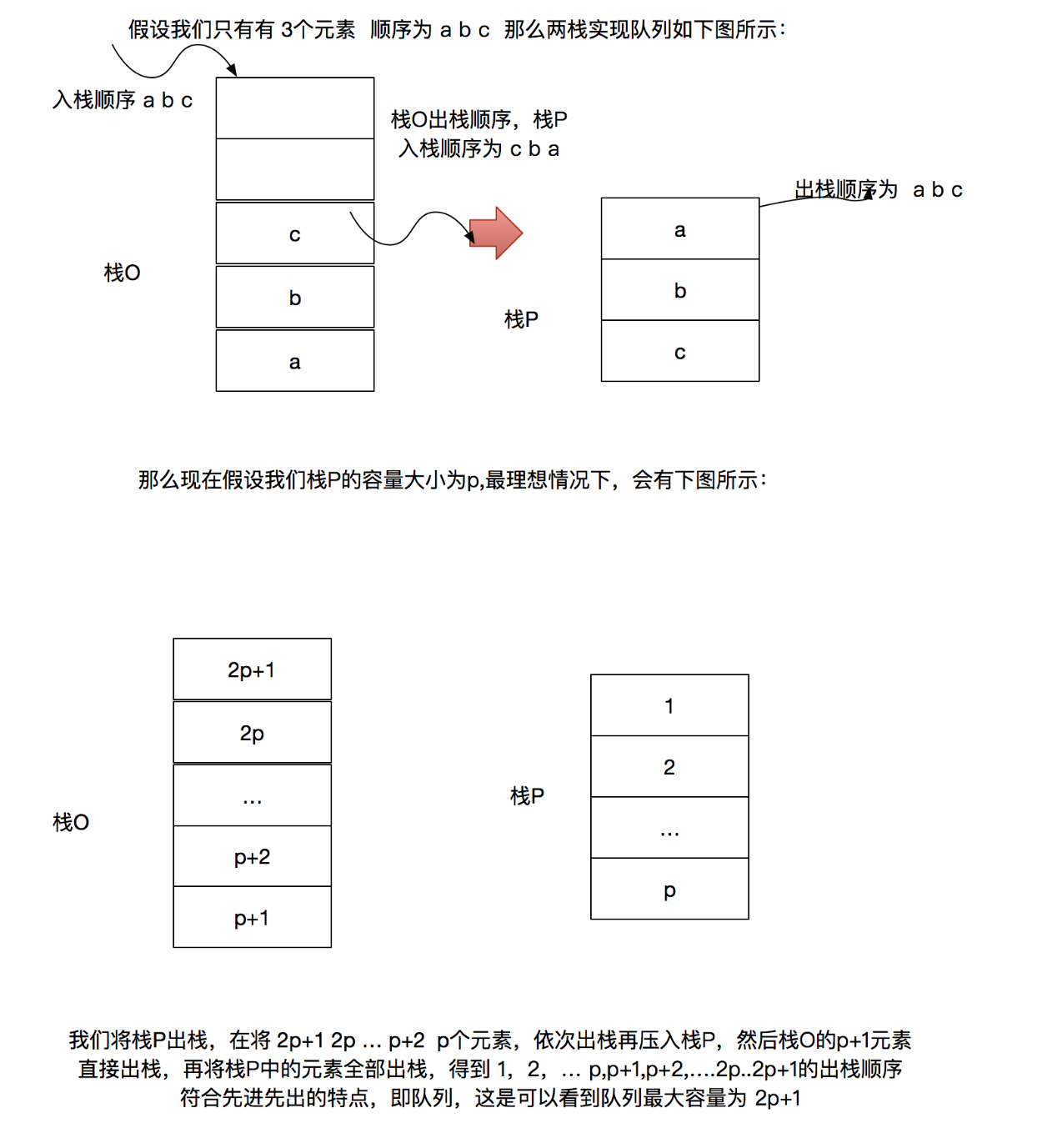


A picture containing text, font, white, screenshot

Description automatically generated

卡特兰数，相关问题还有：已知前序遍历的顺序是xxxx求这棵树有多少种形状

用俩个栈模拟实现一个队列，如果栈的容量分别是O和P(O>P)，那么模拟实现的队列最大容量是多少2P+1



可以这样理解么： 最后p和o全部出栈，不能有p+1在p前面的情况，必须保证先进先出，所以p全部出栈时是1、2...p，o比p大，我们暂且存p+1个就是从p+1...2p+1，出栈后放入p中，只能放2p+1...p+2，此时o中还剩个p+1，现在我们已经得到1..p，再将o中的p+1出栈，再将之前o压到p中的元素出栈，可以得到1...2p+1；如果o中再多存一个，存到2p+2的话，那最后o中就会剩p+2和p+1，出栈的时候p+2在p+1前面，不符合队列的先进先出了，所以最多只能到2p+1

一个空栈，如果有顺序输入序列：a1，a2，a3...an（个数大于3），而且输出第一个元素为 a(n-1), 那么所有元素都出栈后

a(n-2) 一定比 a(n-3) 先出 已知输出第一个元素为 a(n-1)，此时a1~a(n-1)全部入栈，an未入栈。

所以a1~a(n-2)出栈顺序确定，an出栈顺序在a1~a(n-2)间任意位置都可以。

5/8

两个栈共享一片连续内存空间时,为提高内存利用率,减少溢出机会,应把两个栈的栈底分别设在这片内存空间的两端

一个栈的入栈序列为1,2,3,...,n ，其出栈序列是 p 1 ,p 2 ,p 3 ,...p n 。若p 2 = 3，则 p 3 可能取值的个数是n-1

第二个出栈的是3,根据栈先进后出的规律，可知第一个可能是1也有可能是2。假设第一个出栈的是1,则留在栈里边的是2,等3出栈后2接着出栈。对于第一个出栈的是2情况也可以推出，3后面可能是1,其他的数字可以进栈接着出栈得到。因此，唯一能排除的数字就是3,其他数字都有可能！故为n-1

已知栈s允许在两端出栈，但只允许在一端入栈;队列Q只允许在一 端入队列，在另一端出队列。设栈s和队列Q的初始状态为空, e1,e2,e3, e4,e5,e6 依次通过栈S,一个元素出栈后即进队列Q，则不可能得到的出队列的顺序是 e5,e1,e6,e3,e2,e4

若一个栈以向量 V... 存储，初始栈顶指针 top 为 n+1，则下面x入栈的正确操作是

top=top-1; V[top]=x

1.如果初始化top=-1，则栈底地址到栈顶是由小到大的，这样top=-1的时候才表示栈为空

2.如果初始化top=n+1，（n是表示栈的容量），则栈底地址到栈顶是由大到小的，这样top=n+1的时候才表示栈为空

题目显然符合2情况，所以top - -才符合栈顶指针向上移动