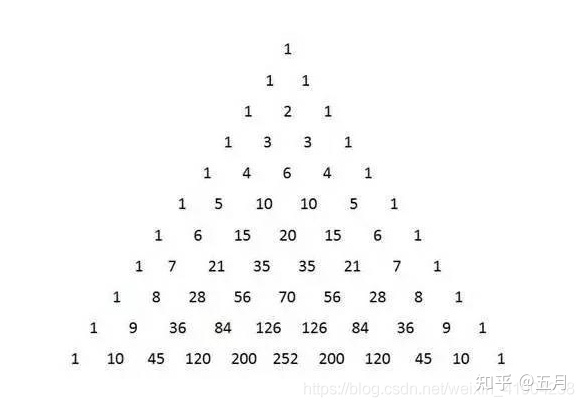
**杨辉三角的实现方法（数据结构之队列）**

学过C语言编程的，势必都要接触到杨辉三角，它本身并不深奥，就是一种二项式系数的几何排列，然后以三角形的形式给我们展现出来，说白了，就是两个未知数之和的幂次方后的各项系数。

杨辉三角的几何排列如下：



1

1 1

1 2 1

1 3 3 1

1 4 6 4 1

稍微解释一下，第一行就是（a+b）的0次方的各项系数，第二行就是（a+b）的1次方的各项系数，第三行就是（a+b）的二次方的各项系数，依次类推，正是因为有了杨辉三角，所以我们在计算幂次方各项系数的时候可以完全按照杨辉三角排列来匹配各项的系数，大大提高了我们的计算效率。

而且也可以轻易看出杨辉三角的基本特征：每一行两边的系数都是1，其他各项的系数都是肩上（上一行）两个系数之和。

方法一：利用数组和循环构建杨辉三角；

方法二：利用数据结构中的队列构建杨辉三角；

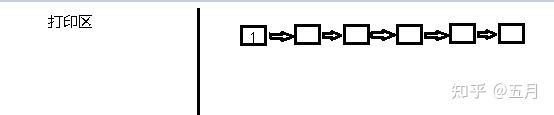
下面就是利用数据结构中的队列来构建杨辉三角：

在这里我就讲讲如何利用队列来构建杨辉三角之前，首先要知道队列具体是个什么东西。

学过链表的都知道，链表可以在任何位置进行插入和删除，那么当链表被进行限制，只能在同一端进行插入和删除，那么这个被限制的链表，就成了栈；同样的，当链表被限制只能在两端中的一端进行插入，另一端进行删除，那么这个被限制的链表就是队列，插入的那一端就是队尾，删除的那一端就是队首。

假设我们此时需要打印出四行的杨辉三角，以下是算法的设计思路：

因为杨辉三角都是从第一行开始构建的，因为第一行只有一个系数1，所以队列先插入系数1，此时还没有开始打印，因此打印区是空的。



然后是开始插入第二行的系数数据，从第二行开始，遵循四个步骤：

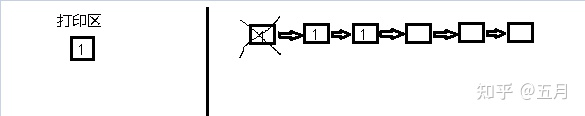
1、第一个系数肯定是1，因此队列插入1；

2、进行出队，把出队数据打印出来，并将出队元素与此时队首元素进行相加并保存，保存的数据就是中间各项系数了；

3、每一行最后一个系数肯定是1，因此队列插入一个1；

4、将此时处于队首的元素出队，并打印出来；

第二行的第一个系数毫无疑问肯定是1，所以队列再插入一个1，中间没有任何需要肩上系数相加的系数，所以直接跳过第二步，第三步再插入一个1，说明这一行的系数插入完成了。接着将处于队首的系数进行一个出队操作，，将出队的数据打印出来，打印区就打印出了杨辉三角的第一行系数。

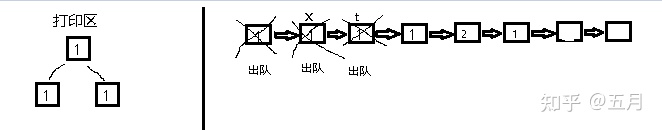


第三行，第一步插入1；

第二步，出队的元素是1，打印出来，并用一个变量x保存起来，即x = 1，此时队首也是1，用t保存起来，那么两者相加就是1+1 = 2，将相加结果用temp保存起来，并插入队列中。

第三步，第三行最后一个系数是1，因此队列插入1；

第四步，将此时处于队首的元素，此时是1，打印出来。



看，打印区已经打印除了杨辉三角的前两行，第三行还留在队列中没有打印出来，因为第四行的系数计算需要第三行的数据，现在还不能出队打印。

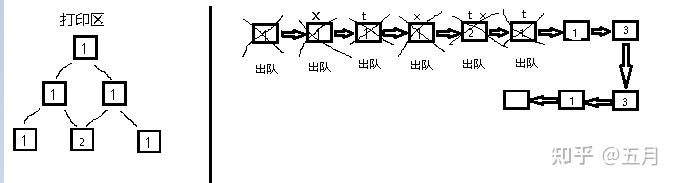
第四行，第一步，插入1；

第二步，出队并打印一个元素，此时是1，用变量x保存，即x = 1，再加上此时队首是2，用变量t保存，即t = 2，两者相加用temp保存起来，即temp = 1+2 = 3，入队。由于第四行中间需要相加的元素有两个，因此第二步需要执行两次。

即出队并打印一个元素，此时就是2了，用变量x保存起来，即x = 2，此时的队首就不再是2了，因为2已经被出队了，此时队首是1，用变量t保存，即t = 1，两者相加的结果用temp保存起来并出队，即temp = 2 + 1 = 3；

第三步，出队一个1，杨辉三角第四行元素再队列中存储完毕；

第四步，将此时处于队首的元素出队打印，也就是1；



现在，就已经打印出了杨辉三角的前三行了，到现在可能有些朋友发现一个问题了，就是无论想要打印几行的杨辉三角，最后一行肯定是被留在队列中的，那么总体的最后一步，就是需要把队列中剩下的元素全部依次出队打印出来，那么四行的杨辉三角就被完全打印出来了。

以下是编译结果：

