## 金刚坐飞机问题

乘客们飞机号（1，2，3，…，N）依次排队上飞机。突然来了一只金刚，他有飞机票，但是他插队第一个登上飞机，随意选择一个作为坐下，其他乘客两种反应：

1. 其他人也随意找位置坐下，并且不给其他人让座。
2. 如果自己位置没有被占，就赶紧坐下，如果位置被占，就随机选择位置坐下，并不给其他人让座。

在这两种情况下，第i个乘客坐到自己位置的概率分别是多少？

### 情况一：

每个人都随意坐，所以做到自己位置的概率就是1/N

### 情况二：

假设金刚原本位置为k，不放设置为1，设f(n)表示金刚坐到位置n。

1. 金刚坐到自己位置n=1，乘客i做到自己位置概率=1
2. 金刚坐的位置n > i,乘客i坐到自己位置概率为1
3. 金刚坐到位置i，乘客i做到自己位置概率为0
4. 金刚位置1<n<i时，如下：

乘客i做到自己位置的概率：

f(n) =

其中j表示第n个乘客做的位置。

f(n) = 1/(N-n+1)\*(f(1)+f(n+1)+f(n+1)+…+f(N)); ————（1）

当乘客n做到1（即就是金刚原本位置）的时候，乘客i坐到自己位置概率为1，所以式（1）= 1/(N-n+1)\*( 1+f(n+1)+f(n+1)+…+f(N) );

f(n) \* (N-n+1) = 1 + f(n+1) + f(n+2) + ... +f(N) ————（2）

当金刚做到n+1位置时，也满足上式：

f(n+1) \* (N-n) = 1 + f(n+2) + f(n+3) + … + f(N) —————（3）

式（2）-（1） ——>

f(n)\*(N-n+1) = f(n+1)\*(N-n+1)🡪

f(n) = f(n+1) 🡪

f(n) = f(i-1) 🡪

即就是金刚做到i-1位置的概率 =

## 判断点D是否在三角形ABC之内

### 解法一：

根据面积判断：如果D在ABC之内，则SABD+SACD+SBCD=SABC

海伦公式S = 其中p = 周长的一半。

### 解法二：

D如果在ABC内，则向量AB x AD >= 0 && BC x BD >= 0 && CA x CD >= 0

## 找零钱

卡特兰数 =

形如Cn = C1Cn-1+ C2Cn-2+ C3Cn-3+…+ Cn-1C1

应用：

1. 排队买票，n个人持有100整，n个人持有50整，假设售票厅初始时没有钱，有多少种排序使得售票厅可以给每一个人找钱。
2. 不同的出栈顺序个数
3. 矩阵连乘加括号的方式数目
4. 凸多边形分割三角形

## 最长不重复子串

题目：给定一个字符串（只包含小写字母），求出最长的不重复子串。比如：abcbdac，最长子串为cbda

|  |
| --- |
| class Solution **{**  public**:**  int lengthOfLongestSubstring**(**string s**)** **{**  **if(**s**.**empty**()){**  **return** 0**;**  **}**  int m**[**255**]** **=** **{**0**};**  int last **=** 0**;**  int len **=** s**.**size**();**  int dp**[**len**+**1**]** **=** **{**0**};**  int ret **=** 1**;**  **for(**int i **=** 1 **;** i **<=** len **;++**i**){**  **if(**m**[**s**[**i**-**1**]]** **==** 0**){**  dp**[**i**]** **=** dp**[**i**-**1**]** **+** 1**;**  **}**  **else{**  last **=** max**(**last**,**m**[**s**[**i**-**1**]]);**  dp**[**i**]** **=** i**-**last**;**  **}**  **if(**ret **<** dp**[**i**]){**  ret **=** dp**[**i**];**  **}**  m**[**s**[**i**-**1**]]** **=** i**;**  **}**  **return** ret**;**  **}**  **};** |

## 单调栈

问题：

* 找出数组中每一个数右边第一个比他大的数，要求时间复杂度为O(n).
* 柱状图找出面积最大的矩形

For example,  
Given heights = [2,1,5,6,2,3],  
return 10

|  |
| --- |
| class Solution **{**  public**:**  int largestRectangleArea**(**vector**<**int**>&** heights**)** **{**  **if(**heights**.**empty**()){**  **return** 0**;**  **}**  stack**<**int**>** st**;**  int len **=** heights**.**size**();**  int right**[**len**]** **=** **{**0**};**  **for(**int i **=** 0 **;** i **<** len **;++**i**){**  right**[**i**]** **=** i**;**  **while(!**st**.**empty**()** **&&** heights**[**st**.**top**()]** **>** heights**[**i**]){**  right**[**st**.**top**()]** **=** i**-**1**;**  st**.**pop**();**  **}**  st**.**push**(**i**);**  **}**  int tmp **=** st**.**top**();**  **while(!**st**.**empty**()){**  right**[**st**.**top**()]** **=** tmp**;**  st**.**pop**();**  **}**  int left**[**len**]** **=** **{**0**};**  **for(**int i **=** len**-**1 **;** i **>=** 0 **;--**i**){**  left**[**i**]** **=** i**;**  **while(!**st**.**empty**()** **&&** heights**[**st**.**top**()]** **>** heights**[**i**]){**  left**[**st**.**top**()]** **=** i**+**1**;**  st**.**pop**();**  **}**  st**.**push**(**i**);**  **}**  tmp **=** st**.**top**();**  **while(!**st**.**empty**()){**  left**[**st**.**top**()]** **=** tmp**;**  st**.**pop**();**  **}**  int ret **=** 0**;**  **for(**int i **=** 0 **;** i **<** len **;++**i**){**  **if(**heights**[**i**]\*(**right**[**i**]-**left**[**i**]+**1**)** **>** ret**){**  ret **=** heights**[**i**]\*(**right**[**i**]-**left**[**i**]+**1**);**  **}**  **}**  **return** ret**;**  **}**  **};** |

## 抽中黑球的概率

## 蚂蚁爬杆

## 区间重合判断

## 数独知多少

## 二叉树节点最大距离

## 重建二叉树

## 队列最大值

### 滑动窗口

### 队列最大值