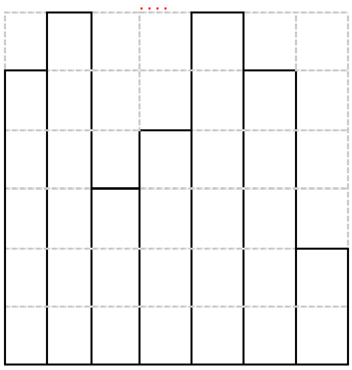
# hw-7

## 0511

### 1. 海报墙由n块宽度相同高度不同的木板组成，那么在此海报墙上能够张贴的最大海报面积是多少？设木板宽度为1，高度为h1,h2,⋯,hn，海报必须整体都粘贴在墙上，并且不能斜贴



$Solution:$

思路：单调栈

对于一段连续的木板，能张贴海报的最大面积由它们中最低的那块木板决定，因此可以记录以每块木板为底所能达到的最大面积，即记录左边和右边分别有几块连续的比它高的。对于这种类型的题目，使用单调栈会非常方便。

具体而言，建立木板结构体

```
struct board
{
    int pre;    //左侧有几块连续的高于它的
    int h;      //木板高度
    int i;      //木板位置
}
```

之后建立单调栈并从左到右扫描入栈

- 若当前 `a[i].h ⩾ s.top().h` ，直接入栈，否则 `s.pop()` 直到满足条件

- 每次 `s.pop()` 时，需要计算以出栈木板 `tmp` 为底能达到的最大面积，并对 `a[i].pre` 进行更新
  - `ans=max(ans, tmp.h * (i-tmp.i+tmp.pre))` (右端比它大的有 `i-tmp.i-1` 个，左端有 `tmp.pre` 个，加上它自身一共宽度为 `i-tmp.i+tmp.pre` )
  - `a[i].pre += tmp.pre+1` ( `tmp.pre` 记录着左端比 `tmp` 高的木板数，而 `tmp` 比 `a[i]` 高，因此需要增加 `tmp.pre+1` )
- 结束后，若 `s` 不为空，需要依次出栈计算面积

每块木板至多进栈一次，出栈一次，每次处理是常数时间，总的时间复杂度为$O(n)$

## 2. 设Fibonacci数列的定义为：F(1)=1, F(2)=1, F(n)=F(n-1)+F(n-2) (n>2)，证明每个大于2的整数n都可以写成至多logn个Fibonacci数之和，并设计算法对于给定的n寻找这样的表示方式

$\color{blue}{Solution:}$

首先设计用$Fibonacci$数表示n的方法:

- 若$n$为$Fibonacci$数，则显然成立
- 否则不妨设$\exists m>0, F(m)<n<F(m+1)$，令$n=F(m)+n'$，则由于$n=F(m)+n'<F(m+1)=F(m)+F(m-1)$，知$n'<F(m-1)$，对$n'$重复上述操作$n'=F(m')+n''$，由于$n'<F(m-1)$，因此一定有$m'<m-1$，即用来表示的$Fibonacci$数是互不相同的，一直重复上述操作直到$0<n'\le 2$，此时$n'$仍然为$Fibonacci$数，故结论成立

下面说明这样的$Fibonacci$数不超过$logn$个

事实上，在这种表示方法下，相邻的两个$Fibonacci$数$F(m')$和$F(m)$，一定有$m'\le m-2$，从而相邻两项之比为

$$\frac{F(m)}{F(m')}\ge\frac{F(m'+2)}{F(m')}=\frac{2F(m')+F(m'-1)}{F(m')}>2$$

相邻两项比值大于2，因此表示法中$Fibonacci$数的个数一定不超过$logn$

## 3. 设有复数x=a+bi和y=c+di，设计算法，只用3次乘法计算乘积xy

$\color{blue}{Solution:}$

$$xy=(a+bi)(c+di)=(ac-bd)+(ad+bc)i$$

- $A = ac$
- $B = bd$
- $C = (a+b)(c+d) = ac + bd + ad + bc$

则

- $ac - bd = A - B$
- $ad + bc = C - A - B$

## 4. 用C/C++如何在int范围内计算二项式系数$C_k^n$，能够计算的n和k的范围越大越好

*Solution* :

```
ans = 1
k = min(n-k, k)
for (int i=1; i<=k; i++)
{
    ans *= (n-i+1);
    ans = ans/i;
}
```

上面这种方法依然使用了乘法，为了更好的避免溢出，可以使用杨辉三角来计算$C_k^n$

$$C_k^n = C_{k-1}^{n-1} + C_k^{n-1}$$

这是一个$O(n^2)$的算法，可以最大程度避免溢出

## 5. 设P是一个n位十进制正整数，如果将P划分为k段，则可得到k个正整数，这k个正整数的乘积称为P的一个k乘积。1）求出1234的所有2乘积；2）对于给定的P和k，求出P的最大k乘积的值

*Solution* :

### (1)

$$1 \times 234 = 234$$
$$12 \times 34 = 408$$
$$123 \times 4 = 492$$

### (2)

使用动态规划求解。设$Num(p,i)$是整数$p$的最低$i$位构成的数。设$Prod(p,k)$是整数$p$的最大$k$乘积。则

$$Prod(p,k)=\begin{cases} p & ,k=1 \\ maxi\{\dfrac{p-Num(p,i)}{10^i}\cdot Prod(Num(p,i),k-1) & ,k>1 \end{cases}$$

总的状态数为$nk$，其中$n$为$p$的位数，总的时间复杂度为$O(kn^2)$

## 6. 如何快速计算1^2^3...^n的值，其中^表示按位异或

*Solution* :

找规律:

- 1=1
- 1^2=2
- 1^2^3=0
- 1^2^3^4=4
- 1^2^3^4^5=1
- ……

多写几项发现4个为一次循环，具体而言，可通过如下方法计算

```
int quickxor(int n)
{
    int ans;
    if (n % 4 == 0)
        ans = n;
    else if (n % 4 == 1)
        ans = 1;
    else if (n % 4 == 2)
        ans = n + 1;
    else if (n % 4 == 3)
        ans = 0;
    return ans;
}
```

其严格正确性可由数学归纳法证明:

- n=0,1,2,3时成立
- 假设$n \leq 4k$时均成立
- n=4k+1时，$quickxor(n) = quickxor(4k) \hat{} (4k+1) = 4k \hat{} (4k+1) = 1$ 成立
- n=4k+2时，$quickxor(n) = quickxor(4k+1) \hat{} (4k+2) = 1 \hat{} (4k+2) = 4k+3 = n+1$ 成立
- n=4k+3时，$quickxor(n) = quickxor(4k+2) \hat{} (4k+3) = (4k+3) \hat{} (4k+3) = 0$ 成立
- n=4k+4时，$quickxor(n) = quickxor(4k+3) \hat{} (4k+4) = 0 \hat{} (4k+4) = 4k+4 = n$ 成立

综上，归纳成立，算法正确，时间复杂度为$O(1)$