# 12.6.2018

记忆化搜索，就是通过dp[][]去记录已经到达过的状态，不需要每次到达该节点都继续向下搜索一遍。

# 12.20.2018

**模拟进栈出栈的问题，这里因为是有序的，所以每个数字后面比这个数字小的几个数字应该是逆序**

例如 14532 符合进栈出栈序列 对于4之后比4小的有2、3必须是逆序32排列

判断代码：

int pd()

{

int maxx;

for(int i=0;i<n;i++) //遍历每一个数之后有没有比自身小的

{

int flag=0;

for(int j=i+1;j<n;j++)

{

if(ans[j]<ans[i])

//有小于自身的数的话 要保证之后比自身数小的数都要比ans[j]小 保证逆序

{

if(flag==0)

{

maxx = ans[j];

flag = 1;

}

if(maxx<ans[j])

return 0;

}

}

}

return 1;

}

# 12.21.2018

递推：一步一步利用前一项或前几项推； 已知推未知

递归：自函数 自己调用自己

### 杨辉三角：

1 2 3 4 5 6 7

1

1 1

1 2 1

1 3 3 1

1 4 6 4 1

1 5 10 10 5 1

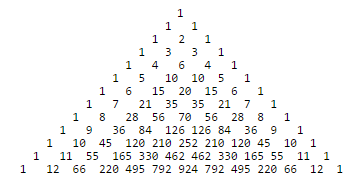
1 6 15 20 15 6 1

a(x,y) = a(x-1,y-1)+a[x-1,y)

1 (y == 1||x==y)

杨辉三角性质 （计算组合数）

杨辉三角上的每一个数字都等于它的左上方和右上方的和（除了边界）



第n行，第m个就是，就是C(n, m) （从0开始）

### 汉诺塔：伪代码

void hanoi(int n,char A,char B,char C) //把A上的n块 借助B 放到C

{

if(n==1)

{

move(A,C);

}

else{ //分解问题

hanoi(n-1,A,C,B); //先把A上的n-1块借助C放在B上

move(A,C);

hanoi(n-1,B,A,C);

}

}

斐波那契数列：



Pell数列：



# 12.23.2018

尺取法：

例：给长度为n的数组和一个整数m，求总和不小于m的连续子序列的最小长度

n = 10，m = 15

5 1 3 5 10 7 4 9 2 8

那么我们先用sum存当前这个子序列的和，从左边第一个数来存，直到这个子序列的和大于等于m为止，再记录下当前长度。

其实相当于当不满足条件就入队，然后得到队列长度，再将队首元素出队，再进行下一次的

入队，直到满足条件再次出队，并且将这一次的长度与历史最短长度进行取舍，最后扫到最

后的元素却无法再满足入队条件的时候就结束，此时用O(n)的时间就可以得到答案。

5 1 3 5 10 7 4 9 2 8

5 1 3 5 10 7 4 9 2 8

5 1 3 5 10 7 4 9 2 8

5 1 3 5 10 7 4 9 2 8

5 1 3 5 10 7 4 9 2 8

5 1 3 5 10 7 4 9 2 8

5 1 3 5 10 7 4 9 2 8

5 1 3 5 10 7 4 9 2 8

for(int i=0;i<n;i++)

{

q.push(a[i]);

now+=a[i];

if(now>m)

{

while(now>=m)

{

flag = 1;

int t = q.size();

ans = min(ans,t);

now-=q.front();

q.pop();

}

}

}

# 12.28.2018

容斥原理（就是包含排斥原理）：设A1，A2为有限集合，其元素个数分别为|A1|,|A2|,则|A1|

# 1.14.2019

map<**int**,map<**int**,**bool**> >mp; 二维map

可以直接

mp[x][y] = true;

默认为false

#include <bits/stdc++.h>

**using** **namespace** std;

map<**int**,map<**int**,**bool**> >mp;

**int** n,m;

**const** **int** maxn = 100005;

**int** pd(**int** nowx,**int** nowy)

{

**if**(nowx>=1&&nowx<=n&&nowy>=1&&nowy<=m){

**return** 1;

    }

**else**{

**printf**("AWaDa!\n");

**return** 0;

    }

}

**int** main()

{

**scanf**("%d%d",&n,&m);

**int** k,len;

**scanf**("%d%d",&k,&len);

**int** x,y;

**scanf**("%d%d",&x,&y);

    mp[x][y] = **true**;

**long** **long** **int** ans = n\*m-1;

**char** order[100005];

**scanf**("%s",order);

**int** nowx = x;

**int** nowy = y;

**for**(**int** i=1;i<=len;i++)

    {

**if**(order[i-1]=='U')

        {

**int** t = pd(nowx-1,nowy);

**if**(t)

            {

                nowx--;

            }

        }

**else** **if**(order[i-1]=='D')

        {

**int** t = pd(nowx+1,nowy);

**if**(t)

            {

                nowx++;

            }

        }

**else** **if**(order[i-1]=='L'){

**int** t = pd(nowx,nowy-1);

**if**(t){

                nowy--;

            }

        }

**else** **if**(order[i-1]=='R'){

**int** t = pd(nowx,nowy+1);

**if**(t){

                nowy++;

            }

        }

**if**(i%k==0)

        {

**if**(mp[nowx][nowy]==**true**)

           {

**printf**("AKTang!\n");

           }

**else**

           {

               ans--;

               mp[nowx][nowy] = **true**;

           }

        }

    }

**printf**("%lld\n",ans);

**return** 0;

}

2.25 哈希表

关键码------~~映射~~--------->值

安或按（an） f（an） 页码4 **哈希冲突**

关键字 哈希函数 哈希值

Key f(x) f(key)

哈希冲突：

Key1≠key2 但是 f(key1) = f(key2)