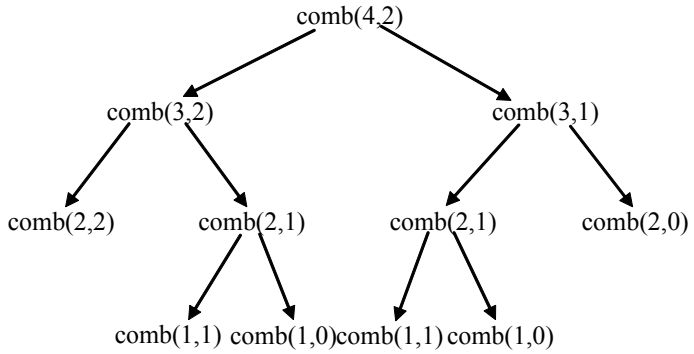
```
        else return comb(n-1,m)+comb(n-1,m-1);      }
```

3. 執行範例：組合公式之遞迴樹(recursion tree)



4. 時間複雜度：令 T(n,m) 代表計算 comb(n,m) 時，所需之時間，則

$$T(n,m)=\begin{cases}1 & ,m=0 \text{ or } m=n\\ T(n-1,m)+T(n-1,m-1)+1 & ,0<m<n\end{cases}$$

可以導出 $T(n,m)=2\times C_m^n-1=O(2^n)$ 。

5. 非遞迴方法：採用動態程式設計(Dynamic Programming)，先計算較簡單的答案，然後再組合出解答。

```
int comb(int n, int m)
{   int i,j;
    int a[MAX_N];
    for (i=0; i<=m; i++) a[i]=1;
    for (i=1; i<=n-m; i++)
    {       for (j=1; j<=m; j++) a[j]=a[j]+a[j-1];      }
    return a[m];
}
```

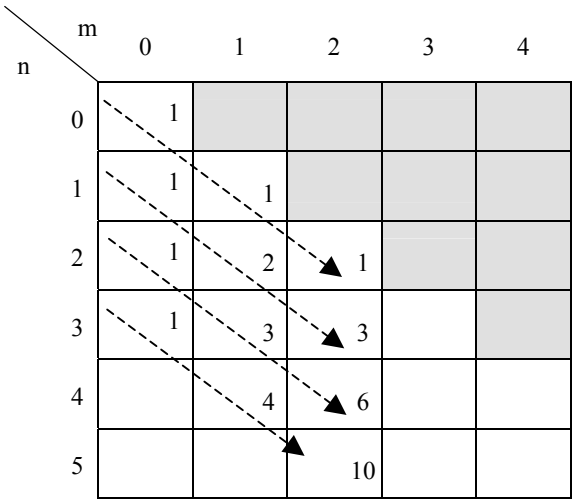
原理：

(1) 使用關係式 $C_m^n=C_{m-1}^{n-1}+C_m^n$ 來逐步計算 C_m^n 的值。

(2) 以對角線順序，由上而下一條一條對角線計算，直到 C_m^n 計算出來爲止。

(3) 時間複雜度：共有 n-m+1 條對角線需要計算，每一條對角線有

m+1 項，故總時間爲O(m(n-m))。



✧ **遞迴重要範例 5**：艾克瑪函數(Ackermann's Function)

1. 遞迴定義： $Ack(m,n)=\begin{cases} n+1 & ,m=0 \\ Ack(m-1,1) & ,m>0 \text{ and } n=0 \\ Ack(m-1,Ack(m,n-1)) & ,m>0 \text{ and } n>0 \end{cases}$

2. 遞迴方法

```
int ack(int m, int n)
{     if (m==0) return n+1;
      else if (n==0) return ack(m-1,1);
      else return ack(m-1,ack(m,n-1));        }
```

3. 試求 Ack(2,2)=?

   Ack(2,2)=Ack(1,Ack(2,1))=Ack(1,5)=Ack(0,Ack(1,4))=Ack(0,6)=7
   Ack(2,1)=Ack(1,Ack(2,0))=Ack(1,3)=Ack(0,Ack(1,2))=Ack(0,4)=5
   Ack(2,0)=Ack(1,1)=Ack(0,Ack(1,0))=Ack(0,2)=3
   Ack(1,0)=Ack(0,1)=2
   Ack(1,2)=Ack(0,Ack(1,1))=Ack(0,3)=4
   Ack(1,4)=Ack(0,Ack(1,3))=Ack(0,5)=6

4. 非遞迴方法：採用動態程式設計(Dynamic Programming)，先計算較

簡單的答案，然後再組合出解答。

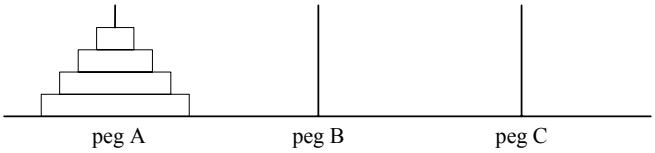
使用兩個 1-D arrays，index[i]=k 表示已計算出 Ack(i,k)，且 Ack(i,k) 的值爲 value[i]。

```
int ack(int m, int n)
{   int index[MaxM], value[MaxN];
    index[0]=0; value [0]=1;
    for (i=1;i<=m;i++) { index[i]=-1; value[i]=1; }
    while (index[m]!=n)
    {   index[0]=value[1];
        value[0]=index[0]+1;
        i=1;
        while (value[i]==index[i-1] && i<=m)
        {   index[i]=index[i]+1;
            value [i]=value[i-1];
            i++;                                    }
    }
    return value [m] ;
}
```

5. Ack(m,n) 的公式，不同的參數 m，有不同的公式如下。
   (1) Ack(0,n)=n+1
   (2) Ack(1,n)=n+2
   (3) Ack(2,n)=2n+3
   (4) Ack(3,n)= $8\times 2^n - 3$
   (5) Ack(4,n)= $2^{2^{\cdot^{\cdot^{2}}}} - 3$ (共有 n+3 個 2)

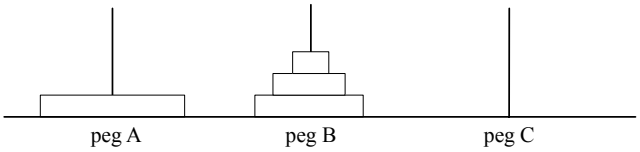
✧ **遞迴重要範例 6：**河內塔問題(Tower of Hanoi)

1. 問題定義：在三根柱子之一，套上大小不同中空的disks形成塔狀，請設法將全部的disks移到另一根柱子上，但須遵守下列規則：
   (1) disk 須一次一個地搬動。
   (2) 大的 disk 不可以放在小的 disk 上。

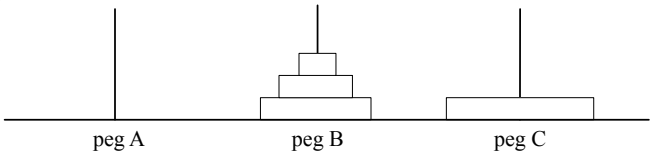


2. 分析：將高度爲 n 的塔搬動時，可以分解成三個動作進行。

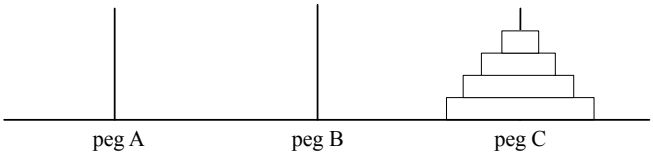
(1) 先將上面 n-1 個 disks 搬到另一根柱子上。



(2) 再將最大的 disk 搬至目的地。



(3) 將剩下的 n-1 個 disks 搬至目的地。



3. 遞迴方法

```
void move(int n, char source, char temp, char dest)
{   if (n) {    move(n-1, source, dest, temp);
                printf("move disk %d from peg %c to %c\n",n,source,dest);
                move(n-1, temp, source, dest);                          }
}
```

4. 碟子搬動次數： $T(n)=\begin{cases} 1 & ,n=1 \\ 2T(n-1)+1 & ,n>1 \end{cases}$

齊次式爲 $T(n)-3T(n-1)+2T(n-2)=0$

特徵方程式 $r^2-3r+2=0$

解得 $r=1,2$