$$T(n-1)-2T(n-2)\le 1 \cdots\cdots\cdots\cdots (2)$$
$$T(n-2)-2T(n-3)\le 1 \cdots\cdots\cdots\cdots (3)$$
$$\cdots\cdots$$
$$T(2)-2T(1)\le 1 \cdots\cdots\cdots\cdots (n-1)$$

計算 $(1)+(2)\times 2+(3)\times 2^2+\ldots+(n-1)\times 2^{n-2}$ 可得

$$T(n)\le 1+2+2^2+\ldots+2^{n-2}+2^{n-1}=2^n-1$$

故 $T(n)=O(2^n)$ 。

(2) 若使用 $T(n)=T(n-1)+T(n-2)+1\ge 2T(n-2)+1$ 推導，

則可得 $T(n)=\Omega((\sqrt{2})^n)\approx\Omega(1.414^n)$ 的時間上限。

6. 討論：費氏級數的公式爲 $F(n)=\frac{1}{\sqrt{5}}(\phi^n-\hat{\phi}^n)$ ，可以求解下面遞迴關係式而得到，推導過程詳見5-2「遞迴關係式(Recurrence Relations)的解法」之內容。

$$F(n)=\begin{cases} n & ,n=0,1 \\ F(n-1)+F(n-2) & ,n>1 \end{cases}$$

7. 討論：遞迴呼叫次數與時間複雜度。

令 T(n) 表示計算 F_n 所需要的呼叫次數，則

$$T(n)=\begin{cases} 1 & ,n=0,1 \\ T(n-1)+T(n-2)+1 & ,n\ge 2 \end{cases}$$

解關係式可得

$$T(n)=\frac{1+\sqrt{5}}{\sqrt{5}}\times\phi^n+\frac{-1+\sqrt{5}}{\sqrt{5}}\times\hat{\phi}^n-1=O(\phi^n)$$

詳見5-2「遞迴關係式(Recurrence Relations)的解法」之內容。

✧ **遞迴重要範例 4**：組合公式(Combination)

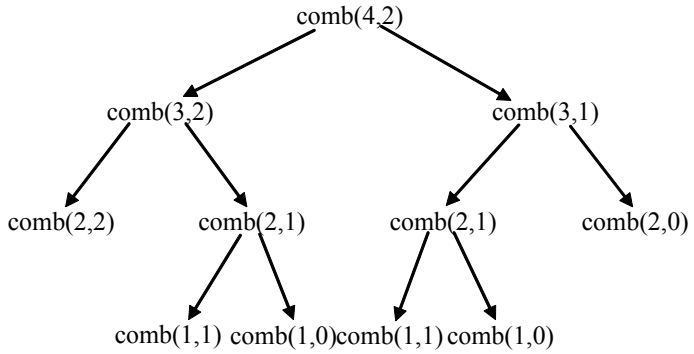
1. 遞迴定義：$C_m^n=\begin{cases} 1 & ,m=0 \text{ or } m=n \\ C_m^{n-1}+C_{m-1}^{n-1} & ,0<m<n \end{cases}$

2. 遞迴方法

```
int comb(int n, int m)
{    if (m==0 || n==m) return 1;
```

```
        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];
}
```

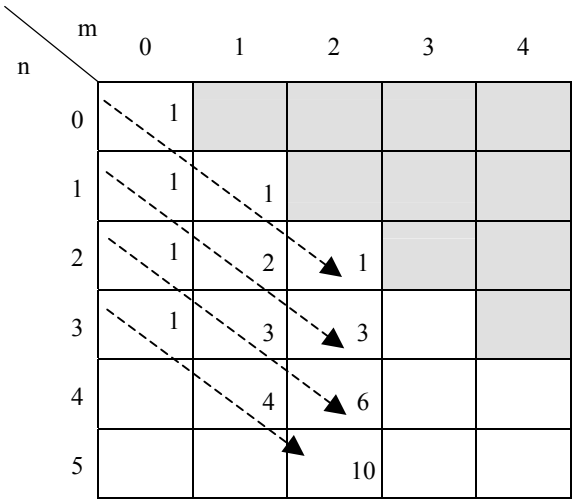
原理：

(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