4 动态规划

Dynamic Programming

引例：费氏数列

- 费氏数列是由13世纪的意大利数学家、来自Pisa的 Leonado Fibnacci发现。
- 费氏数列是由0，1开始，之后的每一项等于前两项之和：

  0，1，1，2，3，5，8，13，21，34，55，89，144……。
- 这个数列有如下一些特性：
  - 前2个数相加等于第3个数
  - 前1个数除以后一个数越往后越无限接近于0.618 (黄金分割)
  - 相邻的两个比率必是一个小于0.618一个大于0.618
  - 后1个数除以前一个数越往后越无限接近于1.618
  - …

$$f(n)=\begin{cases}1 & ,\ if\ n=1,2\\ f(n-1)+f(n-2) & ,\ if\ n\geq 3\end{cases}$$

递归形式的算法：

```
procedure fib(n)
    if n=1 or n=2 then return 1
    else return fib(n-1)+fib(n-2)
```

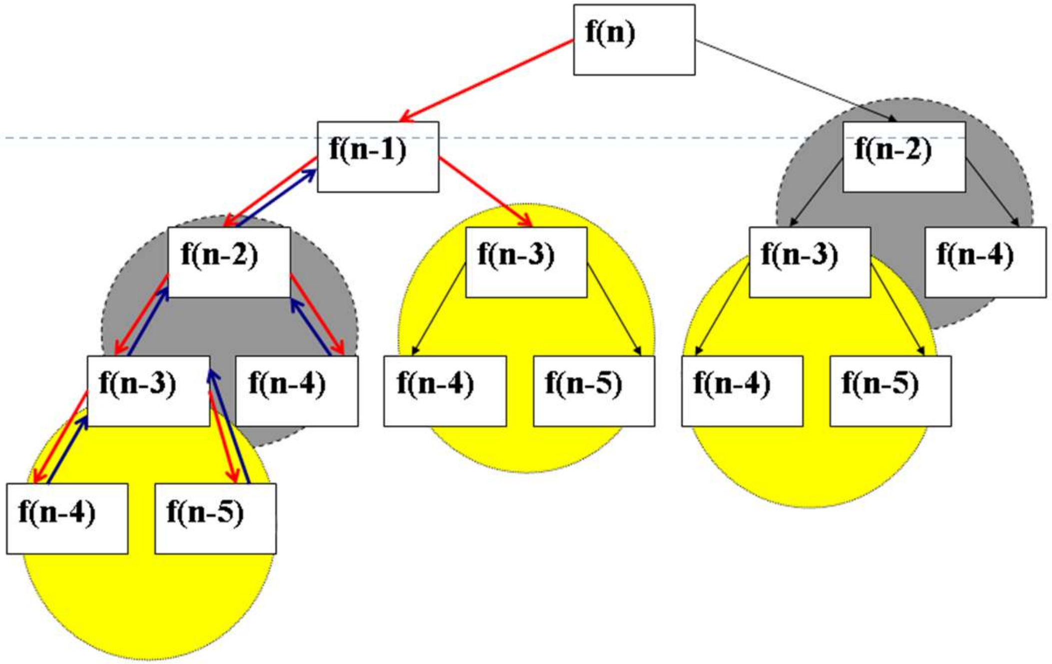
优点：→ 简洁，容易书写以及调试。

缺点：→ 效率低下 。

为何效率低下？

- **使用直观的方式分析**

存在大量重复计算



- 使用时间复杂性的方式分析

$$T(n)=\begin{cases}1 & if\ \ n=1,2\\ T(n-1)+T(n-2) & if\ \ n\geq 3\end{cases}$$

$$T(n)\approx\frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n\approx 0.447(1.618)^n$$

即时间复杂度为输入规模的指数形式。当n=100时，用递归求解的时间 T(100)≈3.53×10^{20}, 若每秒计算10^8次，需111,935年!

解决方法

- 借助于变量存储中间计算结果，消除重复计算。代码片断如下：

```
f1 ← 1
f2 ← 1
for  i ← 3  to n
   result ← f1+f2
   f1 ← f2
   f2 ← result
end for
return result
```

$$T(n) = \Theta(n)$$

动态规划的基本思想

- 动态规划的实质是分治和消除冗余，是一种将问题实例分解为更小的、相似的子问题，并存储子问题的解以避免计算重复的子问题，来解决最优化问题的算法策略。
- 基本步骤：
  - 找出最优解的性质，并刻划其结构特征。
  - 递归地定义最优值。
  - 以自底向上的方式计算出最优值。
  - 根据计算最优值时得到的信息，构造最优解。

矩阵链相乘

- 给定n个连乘的矩阵$M_1 \cdot M_2 \ldots M_{n-1} \cdot M_n$，问：所需要的**最小乘法次数(最优值)**是多少次？对应此最小乘法次数，矩阵是按照什么**结合方式相乘(最优解)**的？

$(A)_{p\times q}\cdot(B)_{q\times r}$ 所需要的乘法次数为： $p\times q\times r$

$$(M_1)_{2\times10}\cdot(M_2)_{10\times2}\cdot(M_3)_{2\times10}$$

$$(M_1\cdot M_2)\cdot M_3 \qquad 2\times10\times2+2\times2\times10=80$$

$$M_1\cdot(M_2\cdot M_3) \qquad 10\times2\times10+2\times10\times10=400$$

观察结论：多个矩阵连乘时，相乘的结合方式不同，所需要的乘法次数大不相同。

$M_1 \cdot M_2 \cdot M_3 \cdots M_n$ 按照何种结合方式相乘，所需要的乘法次数最少？

穷举(蛮力)法：

1.找出所有可能的相乘结合方式;
2.计算每种相乘结合方式所需要的乘法次数;
3.求min;

$f(n)$ 表示n 个矩阵连乘所有可能的结合方式,下面设法求出其解析解。

$$\underbrace{(M_1 \cdot M_2 \cdot M_3 \cdots M_k)}_{f(k)} \cdot \underbrace{(M_{k+1} \cdots M_n)}_{f(n-k)}$$

结论：
穷举法时间复杂度太高。

$$f(n) = \sum_{k=1}^{n-1} f(k) \cdot f(n-k)$$

$$f(1) = 1, f(2) = 1, f(3) = 2 \longrightarrow f(n) = \frac{1}{n} C_{2n-2}^{n-1} \xrightarrow{n! \approx \sqrt{2\pi n}(n/2)^n}$$

$$f(n) = \frac{(2n-2)!}{n((n-1)!)^2} \approx \frac{4^n}{4\sqrt{\pi} n^{1.5}} \longrightarrow f(n) = \Omega(\frac{4^n}{n^{1.5}})$$

使用动态规划法：

$$M_1 \cdot M_2 \cdot M_3 \cdots M_n$$

$$r_1, r_2, r_3, \cdots r_n, r_{n+1}$$

$$(M_i)_{r_i \times r_{i+1}} \quad 1 \le i \le n$$

$$M_{i,j} = M_i \cdot M_{i+1} \cdots M_{j-1} \cdot M_j$$

$C[i, j]$：计算 $M_{i,j}$ 所需的最小乘法次数。

$i = 1, j = n$ 时，原问题得解。

$$\underbrace{M_{i,j}}_{C[i,j]} = \underbrace{(M_i \cdot M_{i+1} \cdots M_{k-1})}_{C[i,k-1]} \cdot \underbrace{(M_k \cdot M_{k+1} \cdots M_{j-1} \cdot M_j)}_{C[k,j]}$$

$$\longleftarrow k \longrightarrow$$

$$C[i, j] = C[i, k-1] + C[k, j] + r_i \cdot r_k \cdot r_{j+1}$$

$$\Downarrow$$

$$C[i, j] = \min_{i < k \le j} \{C[i, k-1] + C[k, j] + r_i \cdot r_k \cdot r_{j+1}\}$$

输入： r[1..n+1]，表示n个矩阵规模的n+1个整数.

输出： n个矩阵连乘的最小乘法次数.

```
for i←1 to n {填充对角线d_0}
  C[i,i] ←0
end for
for d←1 to n-1 {填充对角线d_1到d_{n-1}}
  for i←1 to n-d {填充对角线d_i的每个项目}
      j←i+d  {该对角线上j,i满足的关系}
      C[i,j] ←∞
      for k←i +1 to j
        C[i,j] ←min{ C[i,j], C[i,k-1]+ C[k,j]+ r_i×r_k×r_{j+1}}
      end for
  end for
end for
return C[1,n]
```

$$T(n)=\sum_{d=1}^{n-1}\sum_{i=1}^{n-d}\sum_{k=i+1}^{j}c=\sum_{d=1}^{n-1}\sum_{i=1}^{n-d}\sum_{k=i+1}^{i+d}c=\sum_{d=1}^{n-1}\sum_{i=1}^{n-d}\sum_{k=1}^{d}c=\Theta(n^3)$$

$$T(n)=\sum_{d=1}^{n-1}\sum_{i=1}^{n-d}\sum_{k=i+1}^{j}c=\sum_{d=1}^{n-1}\sum_{i=1}^{n-d}\sum_{k=i+1}^{i+d}c=\sum_{d=1}^{n-1}\sum_{i=1}^{n-d}\sum_{k=1}^{d}c$$

$$=\sum_{d=1}^{n-1}\sum_{i=1}^{n-d}cd=c\sum_{d=1}^{n-1}\sum_{i=1}^{n-d}d$$

$$=c\left(\sum_{i=1}^{n-1}1+\sum_{i=1}^{n-2}2+\sum_{i=1}^{n-3}3+\ldots+\sum_{i=1}^{n-(n-1)}(n-1)\right)$$

$$=c\left((n-1)\cdot 1+(n-2)\cdot 2+(n-3)\cdot 3+\ldots+(n-(n-1))\cdot(n-1)\right)$$

$$=c\left(n\cdot 1+n\cdot 2+n\cdot 3+\ldots+n\cdot(n-1)-1\cdot 1-2\cdot 2-(n-1)\cdot(n-1)\right)$$

$$=c\left(n\left(1+2+3+\ldots+(n-1)\right)-\sum_{k=1}^{n-1}k^2\right)$$

$$=c\left(n\cdot\frac{n(n-1)}{2}-\frac{1}{6}(n-1)n(2n-1)\right)$$

$$=\frac{1}{6}\left(cn^3-cn\right)\qquad=\Theta\left(n^3\right)$$

蛮力方法:

$$T(n)=\Omega\left(\frac{4^n}{n^{1.5}}\right)$$

一个实例

$(M_1)_{5\times10}\cdot(M_2)_{10\times4}\cdot(M_3)_{4\times6}\cdot(M_4)_{6\times10}\cdot(M_5)_{10\times2}$ $r_1=5, r_2=10, r_3=4, r_4=6, r_5=10, r_6=2$

$d=0$ $d=1$ $d=2$ $d=3$ $d=4$

C[1,1]=0 (M_1)	C[1,2]=200 ($M_1 M_2$)	C[1,3]=320	C[1,4]=620	C[1,5]=348
	C[2,2]=0 (M_2)	C[2,3]=240 ($M_2 M_3$)	C[2,4]=640 (M_2) (M_3M_4)	C[2,5]=248
		C[3,3] =0 (M_3)	C[3,4]=240 (M_3M_4)	C[3,5]=168
			C[4,4] =0 (M_4)	C[4,5]=120 ($M_4 M_5$)
				C[5,5] =0 (M_5)

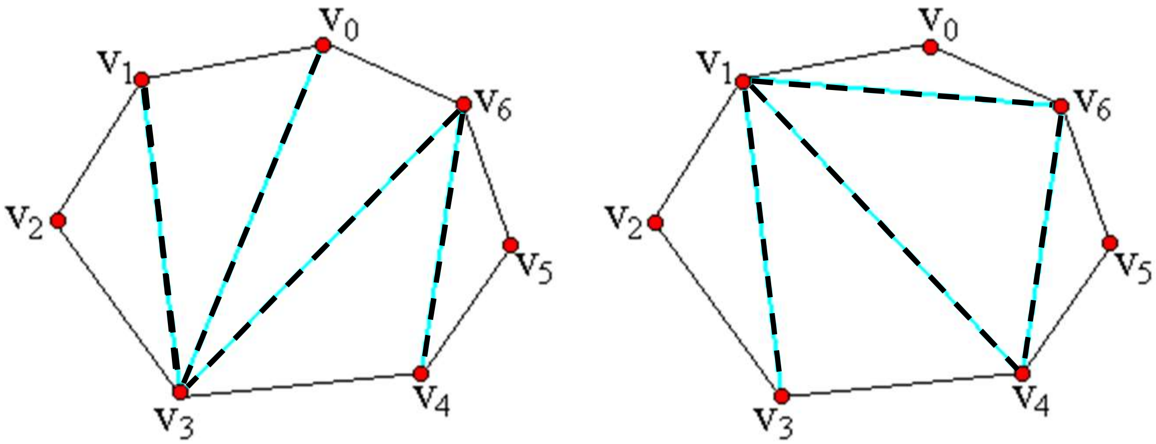
$$C[2,4]=\min_{2<k\le4}\{C[2,k-1]+C[k,4]+r_2\cdot r_k\cdot r_{4+1}\}$$

$$\left.\begin{array}{l} k=3 \ \rightarrow\ \mathrm{C}[2,4]=C[2,2]+C[3,4]+r_2\cdot r_3\cdot r_{4+1}=0+240+10\cdot4\cdot10=640\rightarrow(M_2)\cdot(M_3\cdot M_4) \\ k=4 \ \rightarrow\ \mathrm{C}[2,4]=C[2,3]+C[4,4]+r_2\cdot r_4\cdot r_{4+1}=240+0+10\cdot6\cdot10=840\rightarrow(M_2\cdot M_3)\cdot(M_4) \end{array}\right\}\min$$

平面凸多边形最优三角划分

- **平面多边形**
  - 由在同一平面上，且不在同一直线上的多条线段首尾顺次连结且不相交所组成的图形称为平面多边形。
- **平面凸多边形**
  - 弦：连接平面多边形的任意两个不同顶点的线段。
  - 平面凸多边形：如果一个平面多边形的任意一条弦，要么在该多边形的内部，要么恰好为该多变形的边，那么，称该平面多边形为凸的。否则，称该平面多变形为凹的。
- **三角划分**
  - 将平面凸多边形分割成互不相交的三角形。

平面凸多边形的表示：用平面凸多边形顶点的逆时针序列表示凸多边形，即$P=\{v_0, v_1,\ldots,v_n\}$表示具有n＋1条边的平面凸多边形。



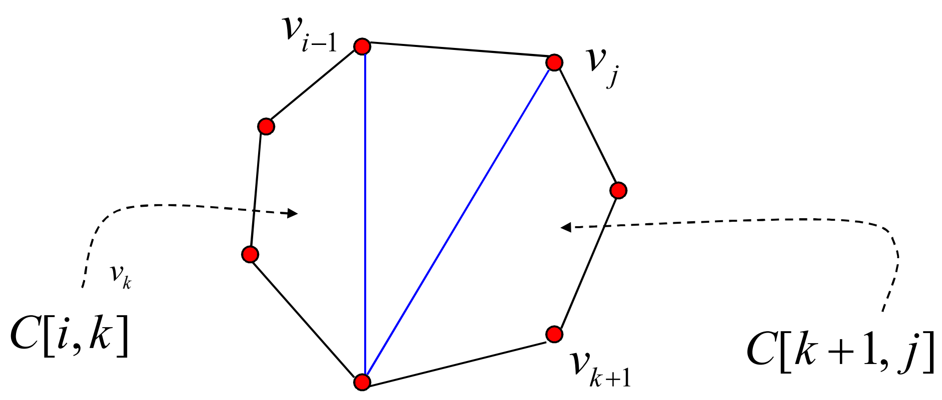
1. 给定一个平面凸多边形P，其三角划分不是唯一的。
2. 给定平面凸多边形P，对于P的每种三角划分，可以定义一个权函数(例如：三角划分中所有三角形的边长之和)。
3. 最优三角划分：使得权函数取最小值的三角划分。

问题：给定平面凸多边形P={v_0, v_1,…,v_n}，求：P的最优三角划分。
(不妨假设：权函数定义为三角划分中所有三角形的边长之和)
求解目标：权函数的最小值，以及对应该最小值的三角划分方式。

类比

矩阵连乘：最小乘法次数，以及对应该最小乘法次数的矩阵结合方式。

若P ={v_0, v_1,…,v_n}是一个凸多边形，那么{v_{i-1},v_i,…,v_j}所构成的必定也是一个凸多边形。定义C[i,j] 为子凸多边形{v_{i-1},v_i,…,v_j}的最优三角划分所对应的权函数值，即其最优值。

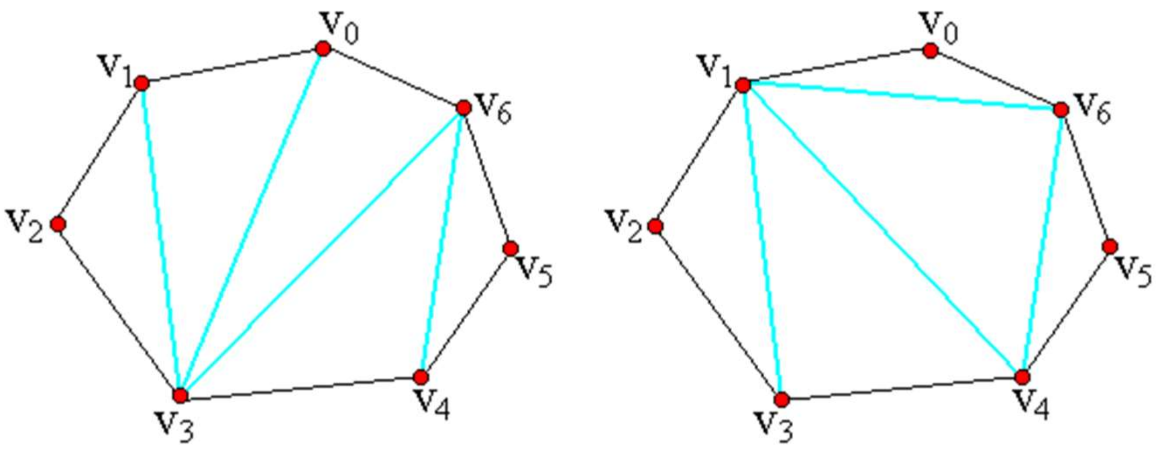


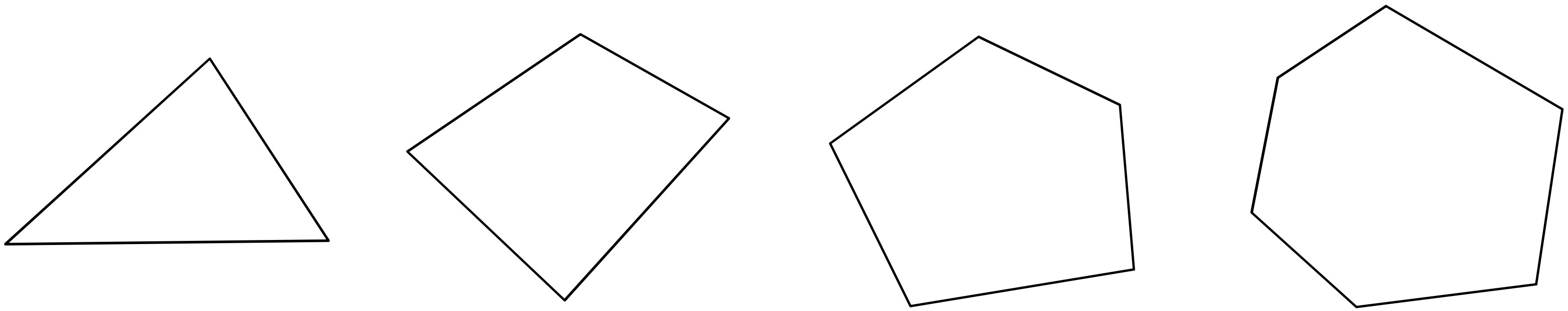
退化的多边形{v_{i-1}, v_i}的权值

$$C[i][j]=\begin{cases} 0 & i=j \\ \min\limits_{i\le k<j}\{C[i][k]+C[k+1][j]+w(v_{i-1}v_kv_j)\} & i<j \end{cases}$$

平面凸多边形三角划分（拓展）

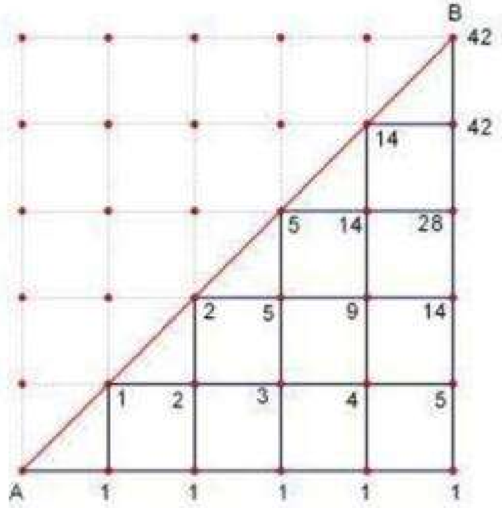
- **不同三角剖分的种类数**



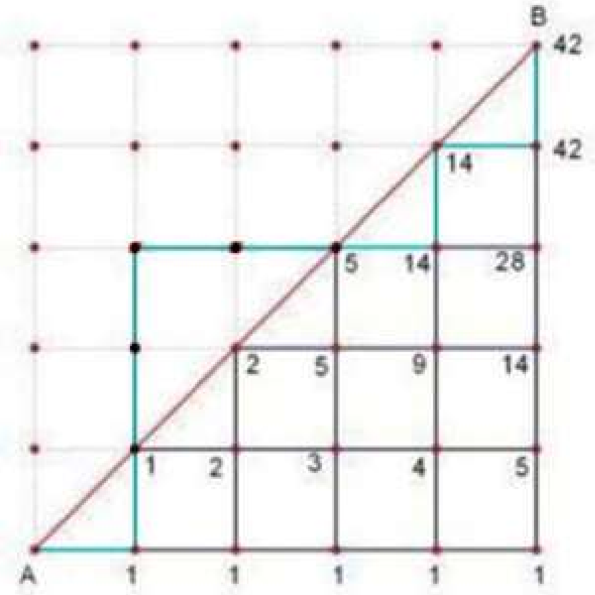


Catalan Number（卡特兰数）

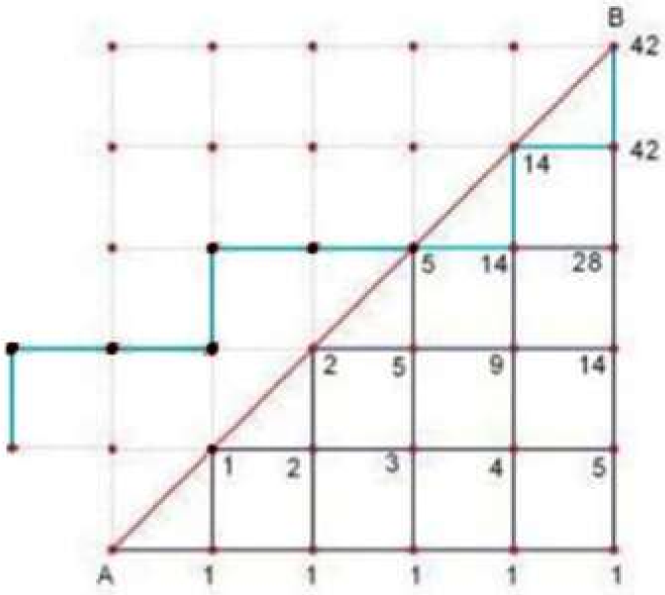
C_0	C_1	C_2	C_3	C_4	C_5	C_6
1	1	2	5	14	42	132



$$C_n = \frac{1}{n+1}\binom{2n}{n}$$



$$\binom{2n}{n}$$

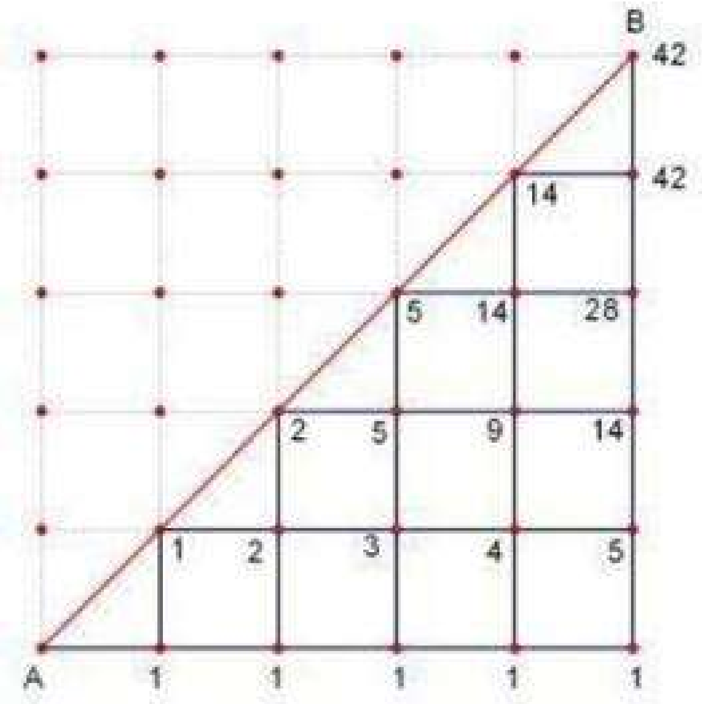


$$\binom{2n}{n+1}$$

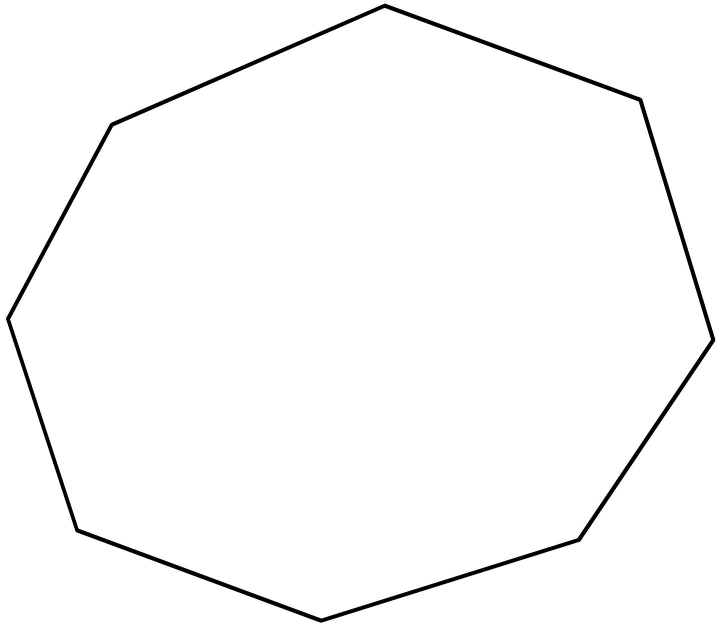
Catalan Number（卡特兰数）

C_0	C_1	C_2	C_3	C_4	C_5	C_6
1	1	2	5	14	42	132

$$C_{n+1} = C_0 C_n + C_1 C_{n-1} + \cdots + C_n C_0 = \sum_{k=0}^{n} C_k C_{n-k}$$



平面凸多边形三角划分（拓展）



$F(n)=F(n-1)+F(n-2)+\ldots+F(4)F(n-3)+F(5)F(n-4)+\ldots+F(n-2)+F(n-1)$

设$P(n-2)=F(n)$

$P(n-2)=P(n-3)+P(n-4)+\ldots+P(2)P(n-5)+P(3)P(n-6)+\ldots+P(n-4)+P(n-3)$

$P(n)=P(n-1)+P(n-2)+\ldots+P(2)P(n-3)+P(3)P(n-4)+\ldots+P(n-2)+P(n-1)$

$P(n)=C_n$

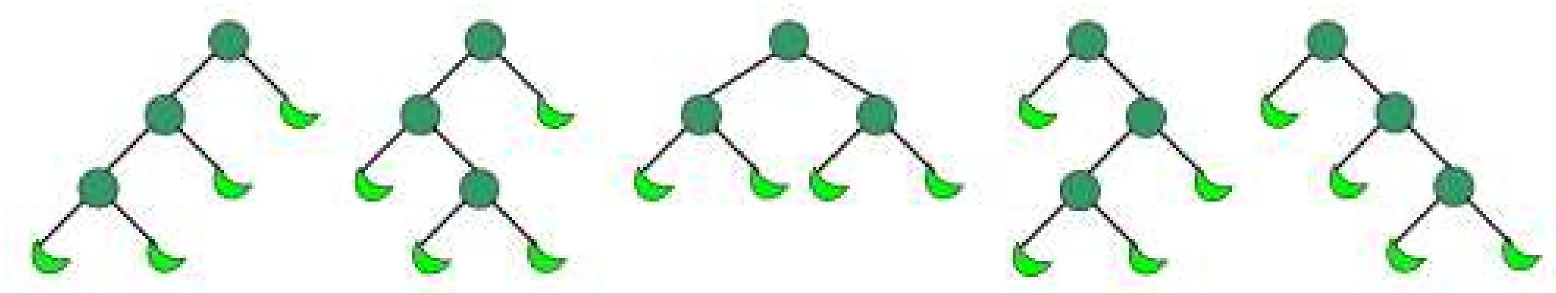
$F(n)=P(n-2)=C_{n-2}$

边数	3	4	5	6	7	8
剖分数	1	2	5	14	42	132

课堂练习

拥有 n个内部节点的二叉树的数量。

例如 3个内部节点的所有二叉树形态



当有6个内部节点时，所有二叉树形态共有多少种？

$$F(n)=F(0)F(n-1)+F(1)F(n-2)+\ldots+F(4)F(n-5)+\ldots+F(n-2)F(1)+F(n-1)F(0)$$

$$F(n)=C_n$$

课堂练习

出栈次序问题

一个栈(无穷大)的进栈序列为1,2,3,..n,有多少个不同的出栈序列?

当n=7时，不同的出栈序列共有多少种？

$$F(n)=F(0)F(n-1)+F(1)F(n-2)+\ldots+F(4)F(n-5)+\ldots+F(n-2)F(1)+F(n-1)F(0)$$

$$F(n)=C_n$$

最长公共子序列问题

- 给定两个定义在字符集∑上的字符串A和B，长度分别为n和m，现在要求它们的**最长**公共子序列的长度值**(最优值)，以及对应的子序列(最优解)**。
- 子序列

$A = a_1 a_2 \cdots a_n$ 的一个子序列是形如下式的一个字符串：$a_{i_1} a_{i_2} \cdots a_{i_k}$,其中

$1 \le i_1 < i_2 < \cdots < i_k \le n$

例如：$A = zxy$ 子序列可以是 : “”, z, x, y, zx, zy, xy, zxy。

但是xz, yz, xyz不是它的子序列。

$$C_n^0 + C_n^1 + \ldots + C_n^n = 2^n$$

- 穷举法(Brute-Force)：
  - 找出A字符串所有可能的子序列(2^n)；
  - 对于A的每一个子序列，判断其是否是B的一个子序列，需要的时间为$\Theta(m)$；
  - 求max;总的时间为$\Theta(m2^n)$.

$A = a_1 a_2 \cdots a_n \qquad B = b_1 b_2 \cdots b_m$

$$\left.\begin{matrix} a_1 a_2 \cdots a_{i-1} a_i \\ b_1 b_2 \cdots b_{j-1} b_j \end{matrix}\right\} \xrightarrow{\text{最长公共子序列的长度值}} C[i, j]$$

$$C[i,j] = \begin{cases} 0 & \text{if} \quad i{=}{=}0 \text{ or } j{=}{=}0 \\ C[i-1, j-1]+1 & \text{if} \quad i>0 \&\& j>0 \&\& a_i == b_j \\ \max\{C[i, j-1], C[i-1, j]\} & \text{if} \quad i>0 \&\& j>0 \&\& a_i \neq b_j \end{cases}$$

$$\left.\begin{matrix} \underline{a_1 a_2 \cdots a_{i-1} a_i} \\ \underline{b_1 b_2 \cdots b_{j-1}} b_j \end{matrix}\right. \longrightarrow C[i, j-1]$$

$$\left.\begin{matrix} \underline{a_1 a_2 \cdots a_{i-1}} a_i \\ \underline{b_1 b_2 \cdots b_{j-1} b_j} \end{matrix}\right. \longrightarrow C[i-1, j]$$

max

```
输入：两个字符串A, B，长度分别为n, m.
输出：X和Y的最长公共子序列长度.
for i ← 0 to n
  C[i,0] ← 0
end for
for j ← 0 to m
  C[0,j] ← 0
end for
for i ← 1 to n
  for j ← 1 to m
     if a_i=b_j then C[i, j] ← C[i−1, j−1]+1
     else C[i, j]←max{C[i, j−1],C[i −1, j]}
     end if
   end for
end for
return C[n,m]
```

$$T(n) = \Theta(nm)$$

$$A = xyxxz \qquad B = zxzyyz$$

一个实例:

使用一个(n+1)×(m+1)的表格来进行计算。
逐行填满表格，问题得解。

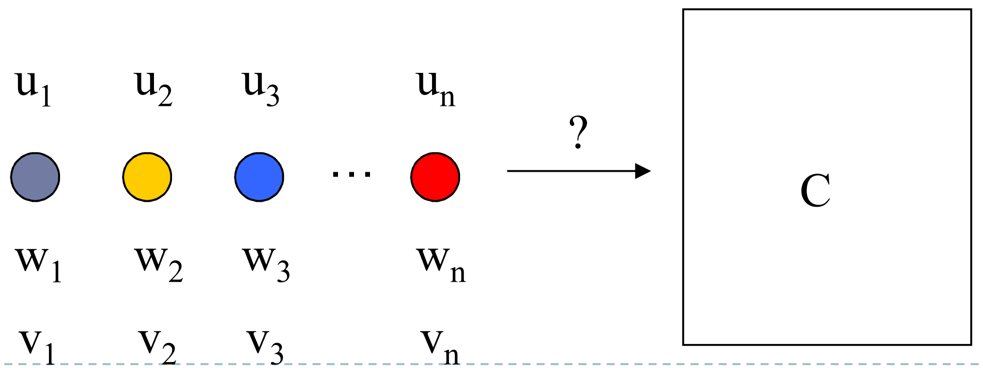
	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	1	1	1	1	1
2	0	0	1	1	2	2	2
3	0	0	1	1	2	2	2
4	0	0	1	1	2	2	2
5	0	1	1	1	2	2	3

因为$a_3 \ne b_4$, 所以$C[3, 4] = \max\{C[3, 3], C[2, 4]\} = \max\{1,2\} = 2$

- 因为$a_5 == b_6$, 所以$C[5, 6] = C[4,5] + 1 = 2+1=3$

0-1背包问题

- 给定n个物品$\{u_1,u_2,\ldots,u_n\}$和一个背包，物品i 的重量为w_i，价值为v_i，已知背包的承重量为C。问：在不撑破背包的条件下，选择哪些物品装入背包，得到的总价值最大?
- 之所以称为0-1 背包问题，是因为一个物品要么装入、要么不装入，这两种状态分别用1 和0 表示。



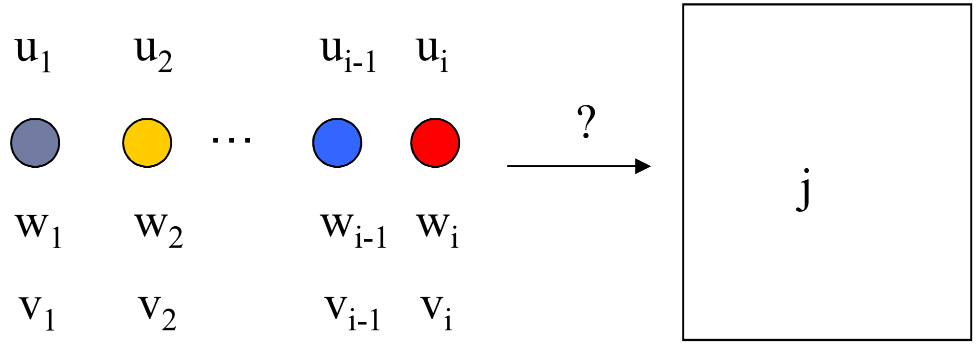
0-1背包问题的形式化描述：

给定$C>0$, $w_i>0$, $v_i>0$, $1\leq i\leq n$, 找出一个n 元的0-1 向量 $(x_1,x_2,\ldots,x_n)$， $x_i\in\{0,1\}$, $1\leq i\leq n$, 求如下优化问题：

$$\max \sum_{i=1}^{n} v_i x_i$$

$$s.t.\quad \sum_{i=1}^{n} w_i x_i \leq C,\quad x_i \in \{0,1\}, 1\leq i\leq n$$

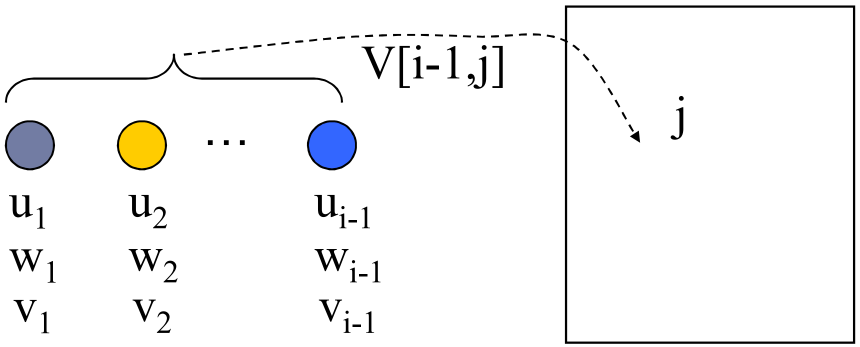
设V[i,j]表示从前i个物品$\{u_1,u_2,\ldots,u_i\}$中取出一部分装入承重量为j的背包所能取得的最大价值。那么，当i=n,j=C时， V[n,C] 就是原问题的解。

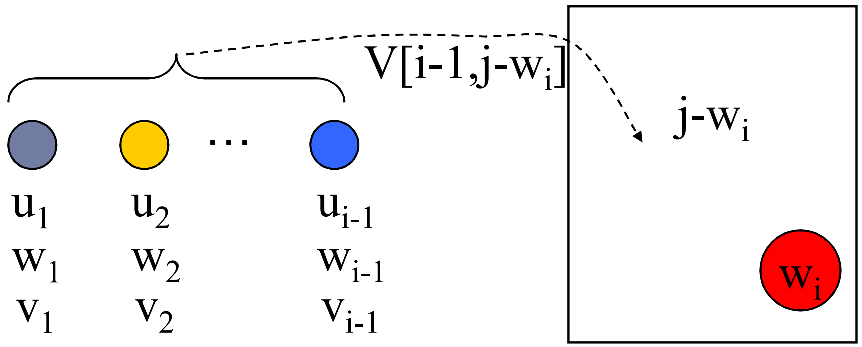


$$V[i,j]=\begin{cases} 0 & \text{if } i=0 \text{ or } j=0 \\ V[i-1,j] & \text{if } j<w_i \\ \max\{V[i-1,j],V[i-1,j-w_i]+v_i\} & \text{if } i>0 \text{ and } j\ge w_i \end{cases}$$

Case 1:

Case 2:





输入：物品集合{$u_1,u_2,\ldots,u_n$}，重量分别为$w_1, w_2,\ldots,w_n$，价值分别为$v_1,v_2,\ldots,v_n$, 承重量为C的背包

输出：背包所能装物品的最大价值

```
for i←0 to n
  V[i,0] ←0
end for
for j←0 to C
  V[0,j] ←0
end for
for i←1 to n //前i个物品
  for j←1 to C //承重量C与物品重量wi均为整数，故j为整数
    V[i,j] ← V[i-1,j]
    if wi≤ j then V[i,j] ←max{ V[i,j], V[i-1,j-wi]+vi}
    end if
  end for
end for
return V[n,C]
```

$$T(n) = \Theta(nC)$$

一个实例

背包的承重量为C=9；给定4个物品，重量(w)分别为2，3，4，5；价值(v)依次为3，4，5，7。问：背包中最多能装的物品的总价值是对少？

	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	0	3	3	3	3	3	3	3	3
2	0	0	3	4	4	7	7	7	7	7
3	0	0	3	4	4	7	8	9	9	12
4	0	0	3	4	5	7	8	10	11	12

因为$w_3=4<=j=7$;

所以$V[3, 7] = \max\{V[3-1,7], V[3-1,7-w_3]+v_3\}$

$= \max\{V[2,7], V[2,3]+5\} = \max\{7,4+5\} = 9$

课堂练习（一）

给出一个求解二项式系数C_n^k的高效算法。你设计的算法的时间复杂度是什么？

$$C_n^k = \begin{cases} 1 & , k==0||k==n \\ C_{n-1}^{k-1}+C_{n-1}^{k} & , 0<k<n \end{cases}$$

输入：整数n，k
输出：二项式系数C_n^k

1.
2.
3.

课堂练习

《算法设计技巧与分析》习题7.9

7.9 考虑用算法 MATCHAIN 把下面 5 个矩阵相乘

$$M_1: 4\times 5, M_2: 5\times 3, M_3: 3\times 6, M_4: 6\times 4, M_5: 4\times 5$$

假设为了要得到乘法 $M_1\times M_2\times M_3\times M_4\times M_5$ 的中间结果如图 7.6 所示，这里的 $C[i,j]$是执行乘法 $M_i\times\cdots\times M_j(1\leqslant i\leqslant j\leqslant 5)$所需要的数量乘法的最少次数。图中还显示了括号表达式所显示的乘法 $M_i\times\cdots\times M_j$ 执行的最优顺序。找出 $C[1,5]$和执行乘法 $M_1\times\cdots\times M_5$ 的最优括号化表达式。

$C[1,1]=0$ M_1	$C[1,2]=60$ M_1M_2	$C[1,3]=132$ $(M_1M_2)M_3$	$C[1,4]=180$ $(M_1M_2)(M_3M_4)$	
	$C[2,2]=0$ M_2	$C[2,3]=90$ M_2M_3	$C[2,4]=132$ $M_2(M_3M_4)$	$C[2,5]=207$ $M_2((M_3M_4)M_5)$
		$C[3,3]=0$ M_3	$C[3,4]=72$ M_3M_4	$C[3,5]=132$ $(M_3M_4)M_5$
			$C[4,4]=0$ M_4	$C[4,5]=120$ M_4M_5
				$C[5,5]=0$ M_5

课堂练习

- 在3×3的方格表的9个小方格中按照规则分别填入1~9中的一个数字。
- 规则：同一行的方格中左边的数字比右边的大，同一列的方格中上边的数字比下边的大。
- 那么，符合规则的填数方法共有多少种？
