7 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
  - …

南京大学 NANJING UNIVERSITY

引例：费氏数列

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

递归形式的算法：

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

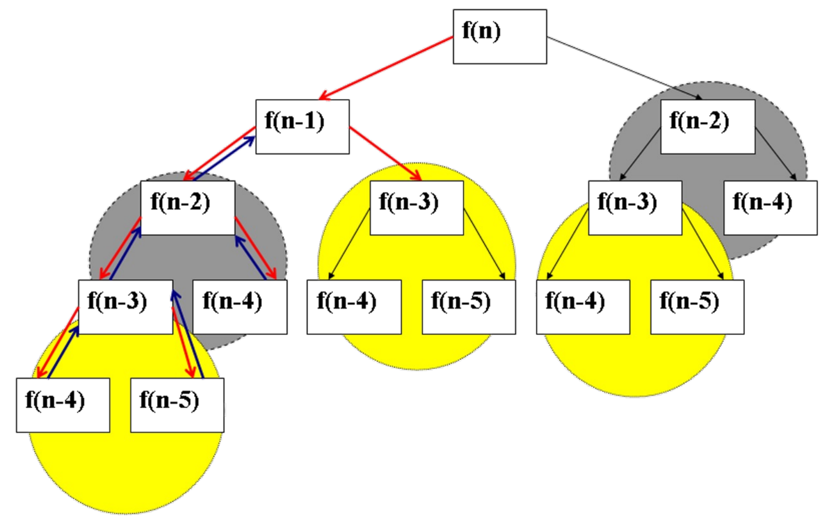
优点：$\longrightarrow$ 简洁，容易书写以及调试。

缺点：$\longrightarrow$ 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

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

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

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：

$$
\begin{array}{l}
f_1 \leftarrow 1 \\
f_2 \leftarrow 1 \\
\text{for } \ i \leftarrow 3 \ \text{ to } n \\
\quad \text{result} \leftarrow f_1 + f_2 \\
\quad f_1 \leftarrow f_2 \\
\quad f_2 \leftarrow \text{result} \\
\text{end for} \\
\text{return result}
\end{array}
$$

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

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： $p\times r\times q = p\times q\times r$

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

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

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

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。

穷举法

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：

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

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。

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

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

$$f(1)=1, f(2)=1, f(3)=2$$

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

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

结论：

穷举法时间复杂度太高。

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。

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

$$\leftarrow k \rightarrow$$

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

$k = i + 1$

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

$k = j$

一个实例

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

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

C[1,1]=0 (M_1)	C[1,2]=200 ($M_1 M_2$)			
	C[2,2]=0 (M_2)	C[2,3]=240 ($M_2 M_3$)		
		C[3,3] =0 (M_3)	C[3,4]=240 (M_3M_4)	
			C[4,4] =0 (M_4)	C[4,5]=120 ($M_4 M_5$)
				C[5,5] =0 (M_5)

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d0}
  C[i,i] ←0
end for
for d←1 to n-1 {填充对角线d1到dn-1}
  for i←1 to n-d {填充对角线di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]+ ri×rk×rj+1}
      end for
  end for
end for
return C[1,n]
```

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

NANJING UNIVERSITY

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

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

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

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

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

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

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

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

蛮力方法:

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

一个实例

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

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

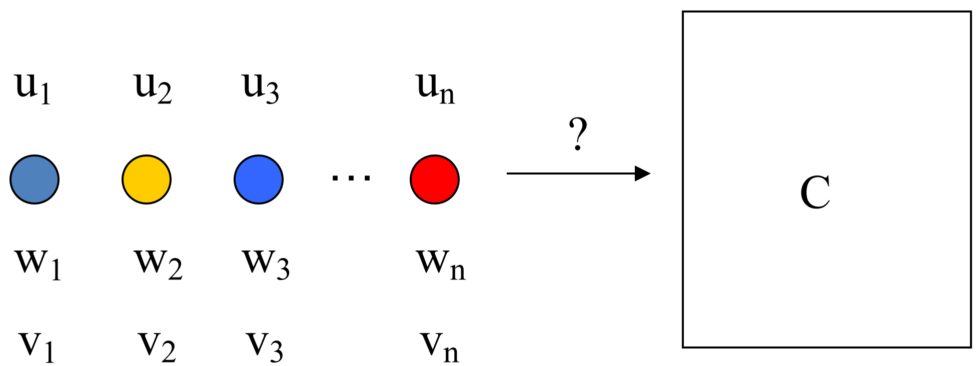
C[1,1]=0 (M_1)	C[1,2]=200 $(M_1 M_2)$	C[1,3]=320	C[1,4]=620	C[1,5]=348
	C[2,2]=0 (M_2)	C[2,3]=240 $(M_2 M_3)$	C[2,4]=640 (M_2) (M_3M_4)	C[2,5]=248
		C[3,3] =0 (M_3)	C[3,4]=240 (M_3M_4)	C[3,5]=168
			C[4,4] =0 (M_4)	C[4,5]=120 $(M_4 M_5)$
				C[5,5] =0 (M_5)

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

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

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：

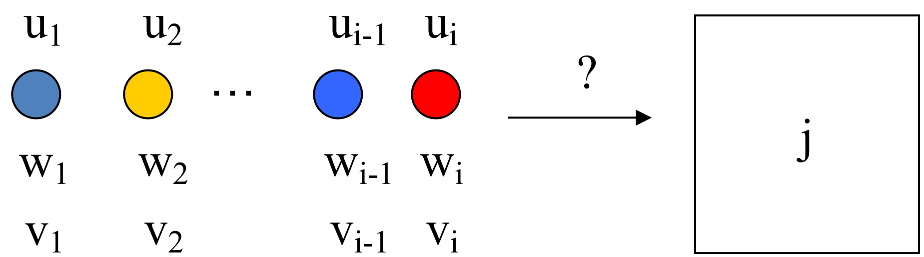
给定$C>0$, $w_i>0$, $v_i>0$, $1\le i\le n$, 找出一个n 元的0-1 向量

$(x_1,x_2,\ldots,x_n)$，$x_i\in\{0,1\}$, $1\le i\le n$, 求如下优化问题：

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

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

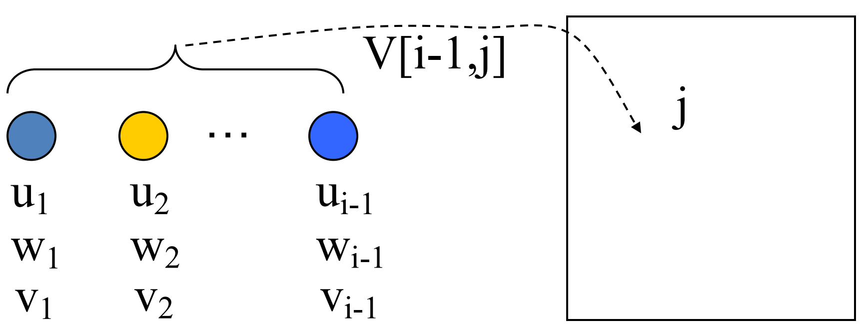
设V[i,j]表示从前i个物品$\{u_1,u_2,\ldots,u_i\}$中取出一部分装入承重量为j的背包所能取得的最大价值。那么，当i=n,j=C时， V[n,C] 就是原问题的解。

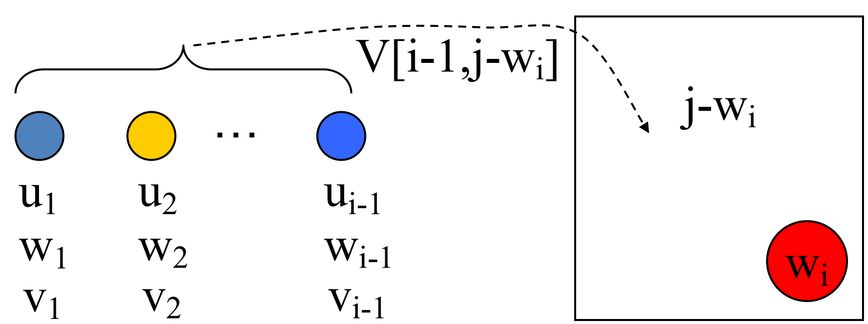


$$
V[i,j]=\begin{cases} 0 & \text{if } i=0 \text{ or } j=0 \\ V[i-1,j] & \text{if } j<w_i \\ \max\{V[i-1,j],V[i-1,j-w_i]+v_i\} & \text{if } i>0 \text{ and } j\ge w_i \end{cases}
$$

Case 1:

Case 2:





一个实例

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

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

南京大学 NANJING UNIVERSITY

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]
```

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

NANJING UNIVERSITY

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**$\Theta(m\ 2^n)$**.

南京大學
NANJING UNIVERSITY

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

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

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

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

一个实例

$A = xyxxz \quad B = zxzyyz$

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。

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

	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	2	2	2	3

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

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

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]
```

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

NANJING UNIVERSITY