2.动态规划

多段决策，每个阶段都需要做出决策，本阶段的决策影响下一阶段的发展，做出全局最优的决策

动态规划的基本要素

- 最优子结构性质
  - 一个最优化策略的子策略总是最优的，原问题的最优解包含子问题最优解
  - 反证法：假设子问题有另一最优解，与原问题矛盾
- 子问题的重叠性质

动态规划的实质

- 分治思想：将问题分解成更小的、相似的子问题
- 解决冗余：存储子问题的解避免计算重复的子问题

动态规划解决步骤

1. 分析最优解性质，并刻画其结构特征
2. 递归地定义最优解
3. 自底向上计算出最优值
4. 逐步构造整个问题的最优解（备忘录，画表法）

SORTBOT

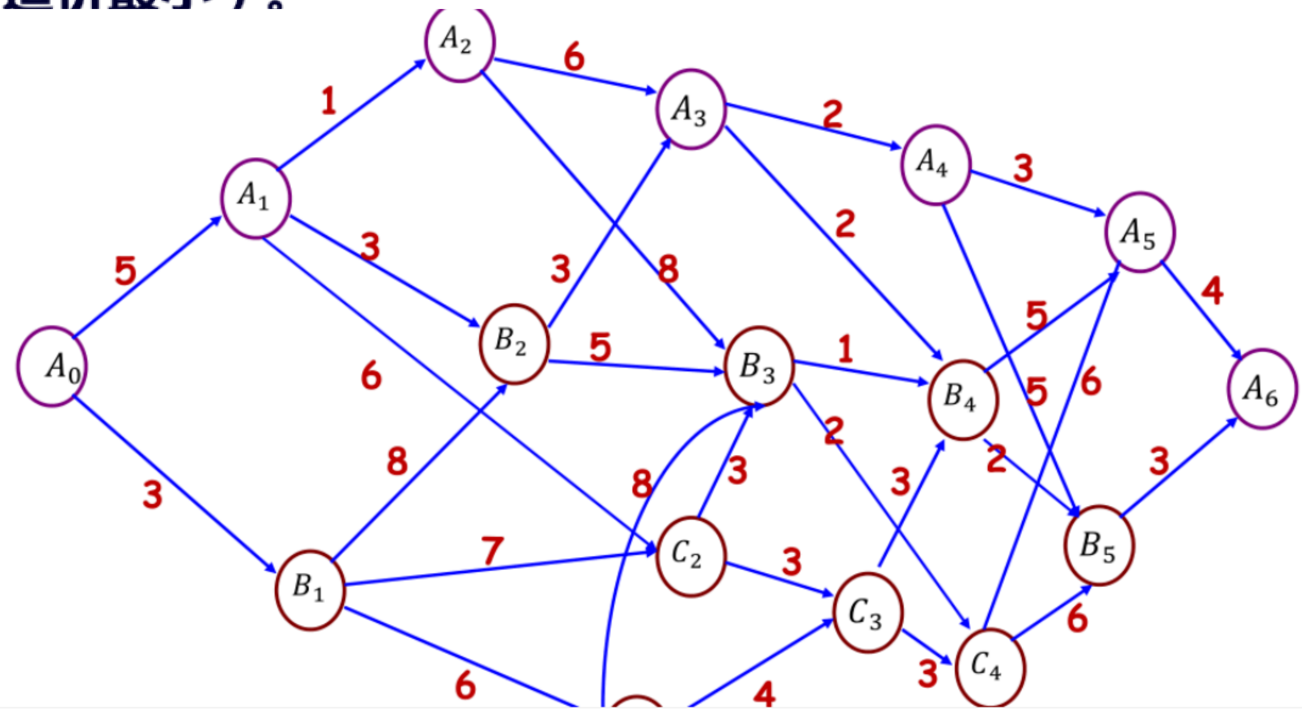
- Subproblem 子问题
- Original problem 原问题
- Relation 原问题与子问题的关系
- Topology 拓扑结构
- Base Case 边界问题
- Original solution 原问题的解
- Time complexity 时间复杂度

最短路径

如图：从A_0点要铺设一条管道到A_6点，中间必须经过5个中间站。第一站可以在、 两地中任选一个，类似地，第二、三、四、五站可供选择的地点分别是：

$\{A_2, B_2, C_2, D_2\}$，$\{A_3, B_3, C_3\}$ ，$\{A_4, B_4, C_4\}$ ，$\{A_5, B_5\}$。

连接两地间管道的距离（或造价）用连线上的数字表示，要求选一条从A_0到A_6的铺管线路，使总距离最短（或总造价最小）。



	0	1	2	3	4	5	6
A	18A	13B	13A	7B	7A	4A	A_6
B		16C	10A	6B	5B	3A	
C			9B	8B	9B		
D			12C				

∴最短距离=18

最短路径=$A_0 \to A_1 \to B_2 \to A_3 \to B_4 \to B_5 \to A_6$

最长公共子序列LCS

设 $X=\{x_1, x_2, \ldots, x_m\}$, $Y=\{y_1, y_2, \ldots y_n\}$ 的最长公共子序列为 $Z=\{z_1, z_2, \ldots, z_k\}$

① $x_m = y_n$, $z_k = x_m = y_n$

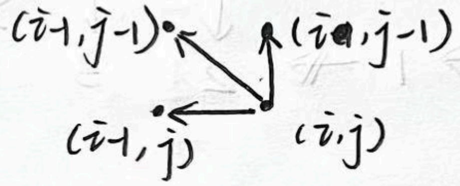
② $x_m \neq \cancel{z_k}\, y_n$, $x_m \neq z_k$, 则 Z 是 X_{m-1} 与 Y 的最长公共子序列

③ $x_m \neq y_n$, $y_n \neq z_k$, 则 Z 是 X 与 Y_{n-1} 的最长公共子序列

最长公共子序列长度

$$c[i][j] = \begin{cases} 0, & i=0, j=0 \\ c[i-1][j-1]+1, & i,j>0,\ x_i = y_j \\ \max\{c[i-1][j],\ c[i][j-1]\}, & i,j>0,\ x_i \neq y_j \end{cases}$$

最长公共子序列路径:



SORTBOT:

SubProblem: $LCS(i,j)$

Original Problem: $LCS(m,n)$

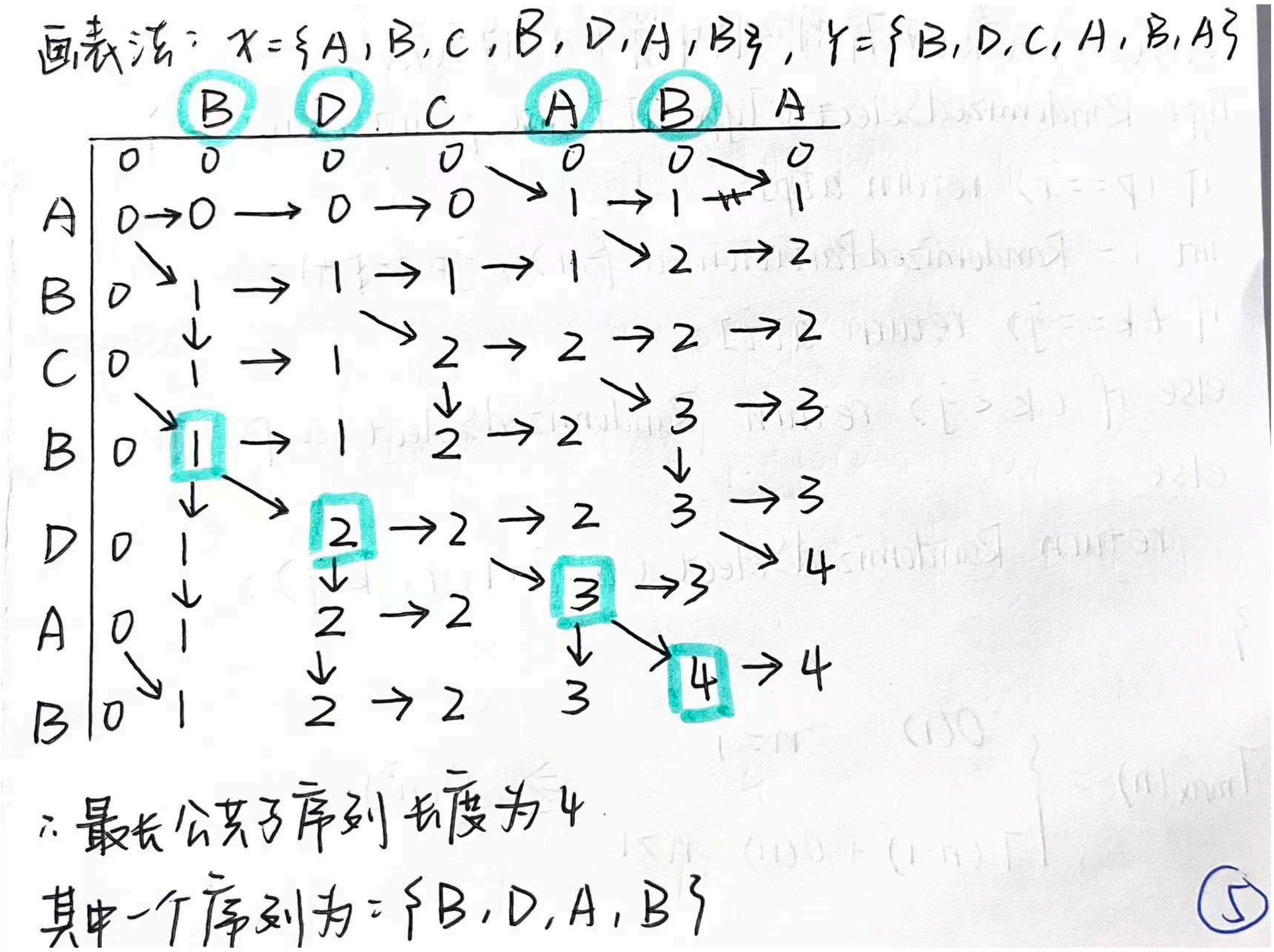
Relation: $LCS(i,j) = \begin{cases} LCS(i-1, j-1)+1, & x_i = y_j \\ \max\{LCS(i-1, j), LCS(i, j-1)\}, & x_i \neq y_j \end{cases}$

Topology: $LCS(i,j)$, $0 <= i <= m$, $0 <= j <= n$

Base Case: $LCS(0,j)=0$, $LCS(i,0)=0$

Original Solution: $mark(i,j)$

Time Complexity: 长度: $O(mn)$ 路径 $O(m+n)$



- 伪代码展示

```
//计算最长公共子序列长度
void LCE_length(m,n,X,Y,c,b){
    for (int i=1;i<=m,i++) c[i][0]==0;
    for (int j=1;j<=n;j++) c[0][j]==0;
    for(int i=0;i<=m;i++){
        for(int j=0;j<=n;j++){
            if (X[i]==Y[j]){
                c[i][j]=c[i-1][j-1]+1;
                b[i][j]=1;
            }
            else if (c[i-1][j]>c[i][j-1]) {
                c[i][j]=c[i-1][j];
                b[i][j]=2;
            }
            else
                c[i][j]=c[i][j-1];
                b[i][j]=3;
        }
    }
}
//构造最长公共子序列
void LCS(i,j,x,b){
    if(i==0||j==0) return;
```

```
    if (b[i][j]='1'){
        LCS(i-1,j-1,x,b);
        cout<<x[i];
    }
    else if (b[i][j]='2'){
        LCS(i-1,j,x,b);
    }
    else LCS(i,j-1,x,b)
}
```

最长回文子序列

- 分别取正序和逆序的序列，取相同的前半段进行复制
- B A C B A C A
- A C A B C A B
- **A C A C A**

木条切割

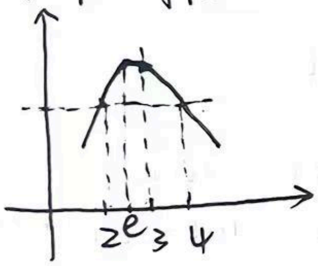
条切割问题（切割后为整数长度）

长度为 n 的条，进行切割后，使总分最大：所有木条长度总乘积

设切割后每块为 x 时，得到的总分最大

即 $y = x^{\frac{n}{x}}$，取对数得：$\ln y = \frac{n}{x}\ln x$

求单调性：$\frac{y'}{y} = \frac{n}{x^2}(1-\ln x)$　当 $x=e$ 时，到达顶点



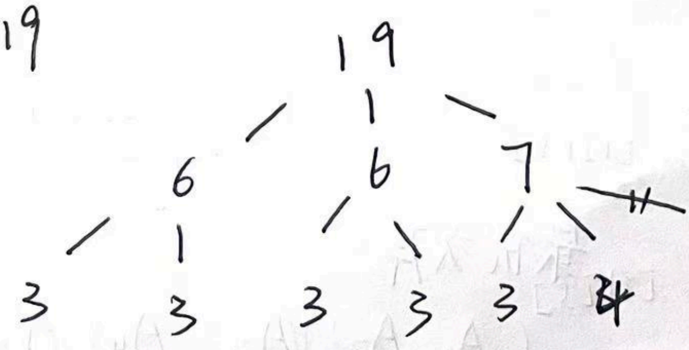
∴ 3 是最接近 e 的合适整数，

即木块尽可能切割成长度为三得分最高

Topology：$dp(n) = \max(dp(i) + dp(n-i))$　$0 \le i \le \frac{n}{2}$

Base Case：$dp(n) = n$　即不分割　　时间复杂度 $O(n^2)$

如 $n = 19$



$\Rightarrow 3 \times 3 \times 3 \times 3 \times 4 \times 3 = 972$

矩阵连乘

给定n个矩阵$\{A_1, A_2, \dots, A_n\}$，求矩阵连乘的最优计算次序

$A[i:j] = A_i \cdot A_{i+1} \cdot \cdots \cdot A_j$

$m[i][j]$: $A[i:j]$所需的最少乘法次数

断点: $s[i][j] = k$

$$m[i][j] = \begin{cases} 0, & i=j \\ \min\{m[i][k] + m[k+1][j] + P_{i-1} \cdot P_k \cdot P_j\}, & i \le j \end{cases}$$

设$n=5$，$P = [\overset{P_0}{10}, \overset{P_1}{5}, \overset{P_2}{1}, \overset{P_3}{10}, \overset{P_4}{2}, \overset{P_5}{10}]$，求矩阵的最优计算次序

i \ j	1	2	3	4	5
1	0	50	150	90	190
2		0	50	30	90
3			0	20	40
4				0	200
5					0

$m[1][2] = m[1][1] + m[2][2] + P_0 \cdot P_1 \cdot P_2 = 50$

$m[2][3] = m[2][2] + m[3][3] + P_1 \cdot P_2 \cdot P_3 = 50$

$m[3][4] = m[3][3] + m[4][4] + P_2 \cdot P_3 \cdot P_4 = 20$

$m[4][5] = m[4][4] + m[5][5] + P_3 \cdot P_4 \cdot P_5 = 200$

$$m[1][3] = \begin{cases} m[1][1] + m[2][3] + P_0 \cdot P_1 \cdot P_3 = 550 \\ m[1][2] + m[3][3] + P_0 \cdot P_2 \cdot P_3 = 150 \checkmark \end{cases}$$

$$m[2][4] = \begin{cases} m[2][2] + m[3][4] + P_1 \cdot P_2 \cdot P_4 = 30 \checkmark \\ m[2][3] + m[4][4] + P_1 \cdot P_3 \cdot P_4 = 150 \end{cases}$$

$$m[3][5] = \begin{cases} m[3][3] + m[4,5] + P_2 \cdot P_3 \cdot P_5 = 300 \\ m[3][4] + m[5][5] + P_2 \cdot P_4 \cdot P_5 = 40 \checkmark \end{cases}$$

$$m[1][4] = \begin{cases} m[1][1] + m[2][4] + P_0 \cdot P_1 \cdot P_4 = 130 \\ m[1][2] + m[3][4] + P_0 \cdot P_2 \cdot P_4 = 90 \checkmark \\ m[1][3] + m[4][4] + P_0 \cdot P_3 \cdot P_4 = 350 \end{cases}$$

$$m[2][5] = \begin{cases} m[2][2] + m[3][5] + P_1 \cdot P_2 \cdot P_5 = 90 \checkmark \\ m[2][3] + m[4][5] + P_1 \cdot P_3 \cdot P_5 = 750 \\ m[2][4] + m[5][5] + P_1 \cdot P_4 \cdot P_5 = 130 \end{cases}$$

$$m[1][5] = \begin{cases} m[1][1] + m[2][5] + P_0 \cdot P_1 \cdot P_5 = 590 \\ m[1][2] + m[3][5] + P_0 \cdot P_2 \cdot P_5 = 190 \checkmark \\ m[1][3] + m[4][5] + P_0 \cdot P_3 \cdot P_5 = 1315 \\ m[1][4] + m[5][5] + P_0 \cdot P_4 \cdot P_5 = 290 \end{cases}$$

构造解:

i \ j	1	2	3	4	5	
1	0	1	2	2	2	A[1:5]
2		0	2	2	2	
3			0	3	4	A[3:5]
4				0	4	
5					0	

最优次序: $(A_1 \ A_2)((A_3 \ A_4) \ A_5)$

最少次数: $m[1][5] = 190$

伪代码展示

```
void MatrixChain(int *p,int n, int **m,int **s){
    for (int i=1;i<=n;i++) m[i][i]=0;//初始化base case
    for (int r=2;r<=n;r++){//r个矩阵连乘
        for (int i=1;i<=n-r+1;i++){//i为本轮矩阵连乘的开始矩阵
            int j=i+r-1;//j是本轮矩阵连乘的最后一个矩阵
```

```
            m[i][j]=m[i][i]+m[i+1][j]+p[i-1]*p[i]*p[j];
            s[i][j]=i;//记录最优断点
            for (int k=i+1;k<j;k++){//遍历断点划分找最优解
                int t=m[i][k]+m[k+1][j]+p[i-1]*p[k]*p[j];
                if (k<m[i][j]){
                    m[i][j]=t;
                    s[i][j]=k;
                }
            }
        }
    }
}
```

- 时间复杂度：$T(n) = O(n^3)$

最大子段和

给定由 n 个整数组成的序列 $a_1, a_2, \cdots, a_n$，求该序列形如 $\sum_{k=i}^{j} a_k$ 的子段和的最大值。特殊地，当所有整数均为负整数时定义其最大子段和为 0。

$$\max\{0, \max_{1\le i\le j\le n} \sum_{k=i}^{j} a_k\}$$

伪代码展示

```
int MaxSum(int n,int a[]){
    int sum=0,b=0;
    for (int i=0;i<n;i++){
        if(b>0){
            b+=a[i];
        }
        else b=a[i];
        if (b>sum) sum=b;
    }
}
```

凸多边形最优三角剖分(类似矩阵连乘，找最优断点)

- $t[i][j]=\begin{cases}0 & i=j\\ min(t[i][k]+t[k+1][j]+w(v_{i-1},v_k,v_j)) & i<j\end{cases}$
- 伪代码展示：

```
void MinWeightTriangle(int n, int **t,int **s){
    for (int i=1;i<=n;i++) t[i][i]=0;//初始化base case
    for (int r=2;r<=n;r++){
        for (int i=1;i<=n-r+1;i++){
            int j=i+r-1;
            t[i][j]=t[i][i]+t[i+1][j]+w(v_{i-1},v_i,v_j);
            s[i][j]=i;//记录最优断点
            for (int k=i+1;k<j;k++){//遍历断点划分找最优解
                int c=t[i][k]+t[k+1][j]+w(v_{i-1},v_k,v_j);
                if (k<t[i][j]){
                    ti][j]=c;
                    s[i][j]=k;
                }
            }
        }
    }
}
```

图像压缩

用像素点灰度值序列{$p_1, p_2, \dots, p_n$}表示图像，$0 \le p_i \le 255 \Rightarrow$ 8位二进制存储

2^3

为减少存储空间，采用变长方式存储

$$\sum_{i=1}^{m} \underset{\text{表示的序列长度}}{l(i)} \times \underset{\text{存储二进制位数}}{b(i)} + 11m$$

$11m \rightarrow$ 表示变长信息（8位长度 + 3位存储位数）

思路：设 s[i] 是像素序列{$p_1, p_2, \dots, p_i$}的最优分段所需的存储位数

此时，则 s[i] = s[i-k] + 保存最后 k 个像素的代价

后者 = k * max{k个灰度值二进制位数} + 11

递归求解 s[i-k]

btw= 位数是从后往前数的

例：求像素序列 9, 11, 10, 12, 15, 127, 178, 220, 10 的最优分段

		9 (4)	11 (4)	10 (4)	12 (4)	15 (4)	127 (~~7~~ 8)	178 (8)	~~128~~ 220 (8)	~~220~~ 10 (4)	~~10~~
i	0	1	2	3	4	5	6	7	8	9	
s(i)	0	15	19	23	27	31	49	58	66	~~78~~ 74	
1		1×4+11 15	15+4+11 ~~19~~ 30	19+4+11 34	23+4+11 38	27+4+11 42	31+7+11 49	49+8+11 68	58+8+11 77	66+4+11 81	
2			2×4+11 19	4×2+4+11×2 34	38	42	4×4+7×2+11×2 52	4×5+8×2+11×2 58	7×6+8×2+11×2 80	8×7+8×2+11×2 94	
3				3×4+11 23	38	42	4×3+7×3+11×2 55	62	4×5+8×3+11×2 66	7×6+8×3+11×2 88	
4					4×4+11 27	42	58	66	70	4×5+8×4+11×2 74	
5						5×4+11 31	61	70	74	78	
6							7×6+11 53	74	78	82	
7								8×~~6~~7+11 ~~59~~ 67	82	86	
8									8×8+11 ~~67~~ 75	90	
9										8×9+11 83	

最优解：9, 11, 10, 12, 15, | 127, 178, 220, 10（从后往前数第4位）

电路布线

- $T(n)=O(n^2)$

电路布线

- 同一绝缘层上连线不相交，第 i 条连线 $(i,\pi(i))$
- 若 $1\le i<j\le n$ 时，$\pi(i)>\pi(j)$，则第 i 条与第 j 条导线相交

设最大不相交子集为 MNS

SubProblem：$Size(i,j)$，$MNS(i,j)$

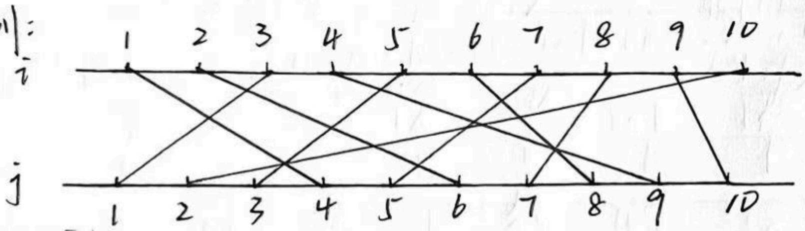
Original Problem：$Size(n,n)$，$MNS(n,n)$

Relation：

$i=1$ 时，$Size(i,j)=\begin{cases}0, & j<\pi(1)\\ 1, & j\ge\pi(1)\end{cases}$

$i>1$ 时，$Size(i,j)=\begin{cases}Size(i-1,j), & j<\pi(i)\\ \max\{Size(i-1,j),\ Size(i-1,\pi(i)-1)+1\}, & j\ge\pi(i)\end{cases}$

例：



i \ j	0	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	1	1	1	1	1	1	1
2	0	0	0	0	1	1	2	2	2	2	2
3	0	$\boxed{1}$	1	1	1	1	2	2	2	2	2
4	0	1	1	1	1	1	2	2	2	3	3
5	0	1	1	$\boxed{2}$	2	2	2	2	2	3	3
6	0	1	1	2	2	2	2	2	3	3	3
7	0	1	1	2	2	$\boxed{3}$	3	3	3	3	3
8	0	1	1	2	2	3	3	$\boxed{4}$	4	4	4
9	0	1	1	2	2	3	3	4	4	4	$\boxed{5}$
10	0	1	2	2	2	3	3	4	4	4	5

∴最大元素个数为5，导线为 $(3,1)$，$(5,3)$，$(7,5)$，$(8,7)$，$(9,10)$

0-1背包

0-1 背包

有n种物品和1个背包，物品i的重量是w_i，其价值为v_i，背包容量为C

如何使装入背包中物品的总价值最大？

目标：$\max \sum_{i=1}^{n} v_i x_i$，$x_i=1$为放入，$x_i=0$为不放入

约束：$\sum_{i=1}^{n} w_i x_i \le C$

边界：$m(n,j)=\begin{cases} 0, & j<w_n \text{ 不放入} \\ v_i, & j>w_n \text{ 放入} \end{cases}$，设j为背包容量

Relation：$m(i,j)=\begin{cases} m(i+1,j), & 0\le j<w_i \\ \max\{m(i+1,j),\ v_i+m(i+1,j-w_i)\}, & j\ge w_i \end{cases}$

例：$n=5$，$C=10$，$w=\{2,2,6,5,4\}$，$v=\{6,3,5,4,6\}$

i \ j	0	1	2	3	4	5	6	7	8	9	10
1	0	0	6	6	6	6	9	9	9	10	15
2	0	0	3	3	6	6	9	9	9	10	11
3	0	0	0	0	6	6	6	6	6	10	11
4	0	0	0	0	6	6	6	6	6	10	10
5	0	0	0	0	6	6	6	6	6	6	6

∴最终装入的序列为(1,1,0,0,1)　　装载量，价值

跳跃点递归：更通用，可用小数，跳跃点(j, m(i,j))，有控制关系

例：$n=5$，$C=10$，$w=\{1,5,4,2,2\}$，$v=\{4,6,4,3,2\}$

$P[6]=\{(0,0)\} \xrightarrow{+(2,2)} \{(2,2)\}=q[6]$

$P[5]=\{(0,0),(2,2)\} \xrightarrow{+(2,3)} q[5]=\{(2,3),(4,5)\}$

$P[4]=\{(0,0),(2,3),(4,5)\} \xrightarrow{+(4,4)} q[4]=\{(4,4),(6,7),(8,9)\}$

$P[3]=\{(0,0),(2,3),(4,5),(6,7),(8,9)\} \xrightarrow{+(5,6)} q[3]=\{(5,6),(7,9),(9,11)\}$ 11超限

$P[2]=\{(0,0),(2,3),(4,5),(5,6),(6,7),(7,9),(9,11)\}$ ~~+(1,4)~~

$\xrightarrow{+(1,4)} q[2]=\{(1,4),(3,7),(5,9),(6,10),(7,11),(8,13),(10,15)\}$

$P[1]=\{(0,0),(1,4),(3,7),(5,9),(6,10),(7,11),(8,13),(10,15)\}$

最终装入的序列(1,1,0,1,1)

流水调度

流水作业调度

确定n个作业的最优加工顺序，使加工完成所需时间最小（有M_1，M_2两台机器）

⇒ M_2等待的时间越短越好t，最短$T(S,t)$，最优$T(N,0)$

Johnson算法：

① $N_1=\{i \mid a_i<b_i\}$，$N_2=\{i \mid a_i>b_i\}$

② N_1中依a_i非减序排序（从小到大）

N_2中依b_i非增序排序（从大到小）

③ N_1中作业接N_2中作业构成满足Johnson法则的最优调度

例

任务	J_1	J_2	J_3	J_4	J_5	J_6
打字(M_1)	30	120	50	20	90	110
打印(M_2)	80	100	90	60	30	10

① $N_1=\{J_1,J_3,J_4\}$　　$N_2=\{J_2,J_5,J_6\}$

② 排序后：$S_1=\{J_4,J_1,J_3\}$　　$S_2=\{J_2,J_5,J_6\}$

③ 最终顺序：$J_4\to J_1\to J_3\to J_2\to J_5\to J_6$

M_1　$20\to 50\to 100\to 240\to \cancel{330}\,310\to \cancel{440}\,420$

M_2　$80\to 160\to 250\to 350\to 380\to \cancel{45}\,430$

• Johnson是充分条件，非必要条件