

## 3.6 多边形游戏

多边形游戏是一个单人玩的游戏，开始时有一个由 $n$ 个顶点构成的多边形。每个顶点被赋予一个整数值，每条边被赋予一个运算符“+”或“\*”。所有边依次用整数从1到 $n$ 编号。

游戏第1步，将一条边删除。

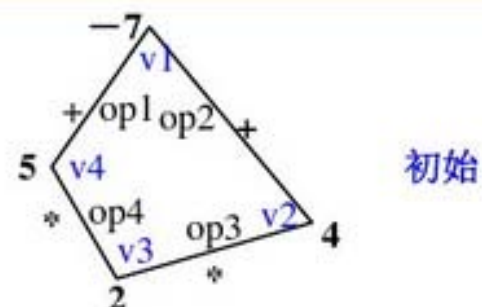
随后 $n-1$ 步按以下方式操作：

- (1) 选择一条边 $E$ 以及由 $E$ 连接着的2个顶点 $V_1$ 和 $V_2$ ；
- (2) 用一个新的顶点取代边 $E$ 以及由 $E$ 连接着的2个顶点 $V_1$ 和 $V_2$ 。将由顶点 $V_1$ 和 $V_2$ 的整数值通过边 $E$ 上的运算得到的结果赋予新顶点。

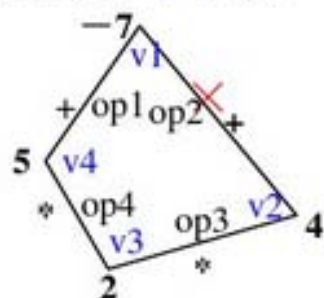
最后，所有边都被删除，游戏结束。游戏的得分就是所剩顶点上的整数值。

问题：对于给定的多边形，计算最高得分。

# 多边形游戏 举例



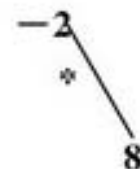
例如：这里画出一个五边形，其中两种选择操作方案，得到的结果也不同。



(1)



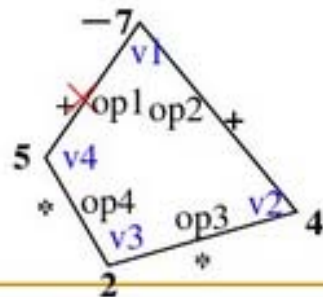
(2)



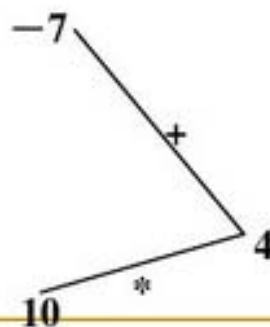
(3)

● -16

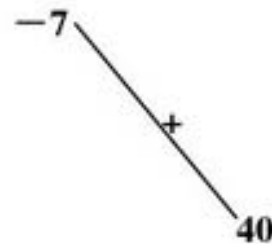
(4)



(1)



(2)



(3)

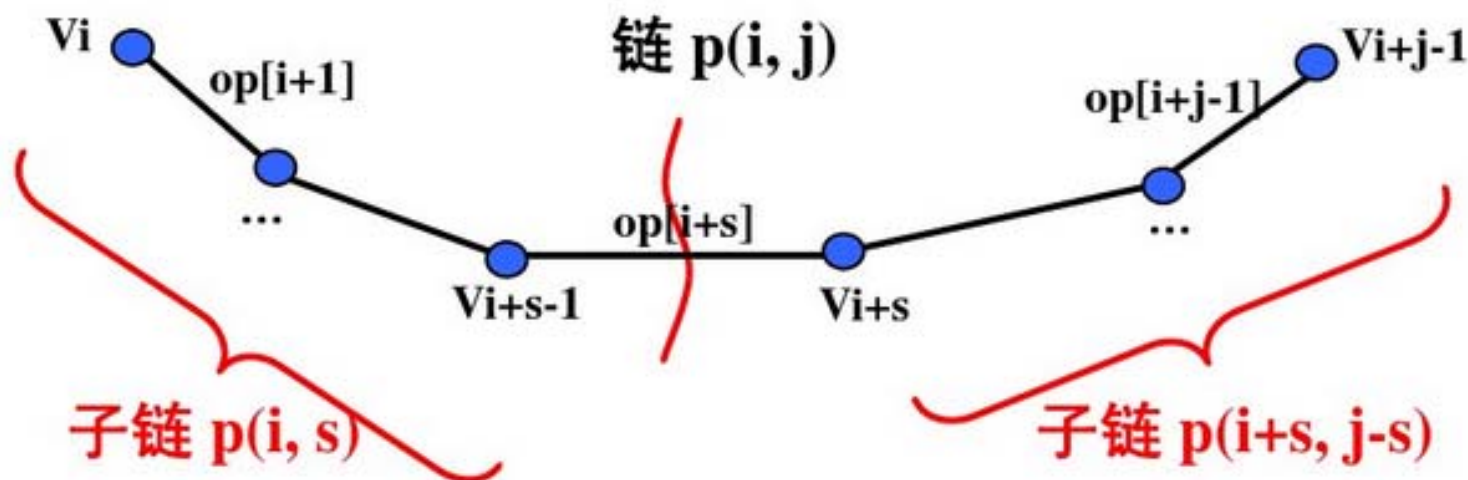
● 33

(4)

## 3.6 多边形游戏

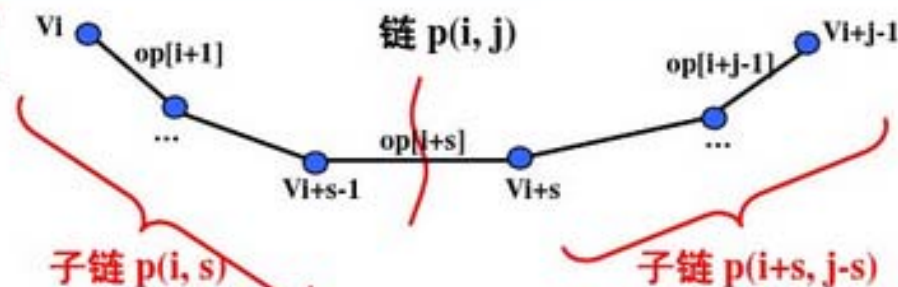
### 最优子结构性质

- 在所给多边形中，从顶点 $i$  ( $1 \leq i \leq n$ ) 开始，长度为 $j$  (链中有 $j$ 个顶点) 的顺时针链 $p(i, j)$  可表示为 $v[i], op[i+1], \dots, v[i+j-1]$ 。
- 如果这条链的最后一次合并运算在 $op[i+s]$  处发生 ( $1 \leq s \leq j-1$ )，则可在 $op[i+s]$  处将链分割为2个子链 $p(i, s)$  和 $p(i+s, j-s)$ 。



## 3.6 多边形游戏

### 最优子结构性质



● 设  $m_1$  是对子链  $p(i, s)$  的任意一种合并方式得到的值，而  $a$  和  $b$  分别是在所有可能的合并中得到的最小值和最大值。  $m_2$  是  $p(i+s, j-s)$  的任意一种合并方式得到的值，而  $c$  和  $d$  分别是在所有可能的合并中得到的最小值和最大值。 依此定义有  $a \leq m_1 \leq b$ ,  $c \leq m_2 \leq d$

● (1) 当  $op[i+s] = '+'$  时，显然有  $a+c \leq m \leq b+d$

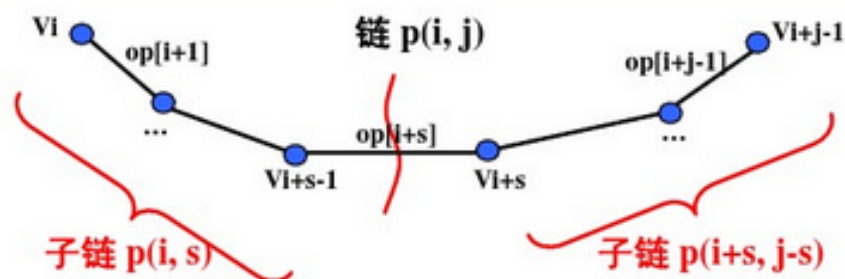
● (2) 当  $op[i+s] = '*'$  时，有  $\min\{ac, ad, bc, bd\} \leq m \leq \max\{ac, ad, bc, bd\}$

● 换句话说，主链的最大值和最小值可由子链的最大值和最小值得到。



## 3.6 多边形游戏

### 递归求解

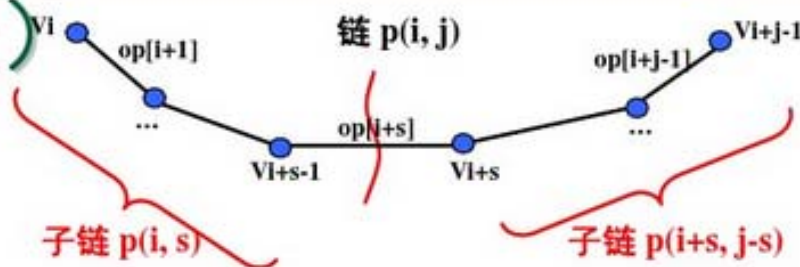


- 定义  $m[i, j, 0]$  为链  $p(i, j)$  合并的最小值，而  $m[i, j, 1]$  为链  $p(i, j)$  合并的最大值。
- 若最优合并在  $op[i+s]$  处，那么  $op[i+s]$  这条边将  $p(i, j)$  分为两个长度小于  $j$  的子链  $p(i, i+s)$  和子链  $p(i+s, j-s)$ ，且从顶点开始的长度小于  $j$  的子链的最大值和最小值均已经计算出了。
- 为叙述方便，记：

$a = m[i, i+s, 0],$	链 $p(i, s)$ 合并的最小值;
$b = m[i, i+s, 1],$	链 $p(i, s)$ 合并的最大值;
$c = m[i+s, j-s, 0],$	链 $p(i+s, j-s)$ 合并的最小值;
$d = m[i+s, j-s, 1],$	链 $p(i+s, j-s)$ 合并的最大值。

## 3.6 多边形游戏

### 递归求解（接上）



● 将  $p(i, j)$  在  $op[i+s]$  处断开的最大值记为  $\max f(i, j, s)$ ，最小值记为  $\min f(i, j, s)$ ，则：

$$\min f(i, j, s) = \begin{cases} a + c & op[i+s] = '+' \\ \min\{ac, ad, bc, bd\} & op[i+s] = '*' \end{cases}$$

$$\max f(i, j, s) = \begin{cases} b + d & op[i+s] = '+' \\ \max\{ac, ad, bc, bd\} & op[i+s] = '*' \end{cases}$$

最优断开的位置  $s$ ，有  $1 \leq s \leq j-1$  共  $j-1$  种情况，由此可得：

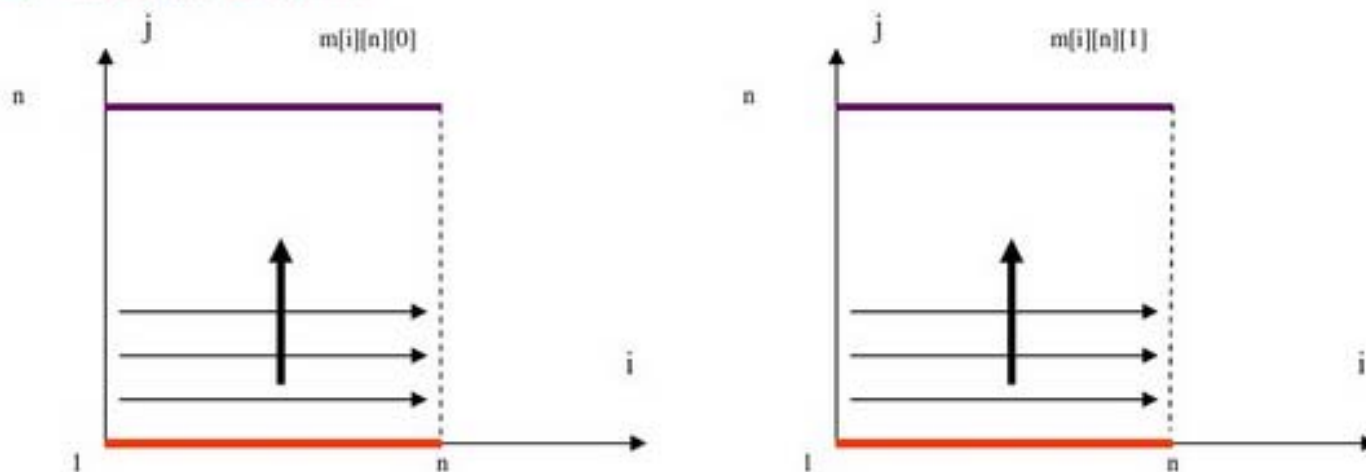
$$m[i, j, 0] = \min_{1 \leq s < j} \{ \min f(i, j, s) \}, \quad 1 \leq i, j \leq n$$

$$m[i, j, 1] = \max_{1 \leq s < j} \{ \max f(i, j, s) \}, \quad 1 \leq i, j \leq n$$

初始边界值显然为：  $m[i, 1, 0] = v[i], \quad 1 \leq i \leq n \quad m[i, 1, 1] = v[i], \quad 1 \leq i \leq n$

## 3.6 多边形游戏

### 计算最优值



$m[i,n,1]$  ( $1 \leq i \leq n$ ) : 即为游戏首次删去第 $i$ 条边后得到的最大得分。

$m[i,n,0]$  ( $1 \leq i \leq n$ ) : 即为游戏首次删去第 $i$ 条边后得到的最小得分。

**算法描述:** 如书P69程序清单。

**复杂性分析:** 算法需要  $O(n^3)$  的计算时间。

```

private static void minMax(int i,int s,int j)
{
    int[] e=new int[5];
    int a=m[i][s][0],
        b=m[i][s][1],
        r=(i+s-1)%n+1,
        c=m[r][j-s][0],
        d=m[r][j-s][1];
    if(op[r]=='+'){
        minf=a+c;
        maxf=b+d;
    }
    else{
        e[1]=a*c;
        e[2]=a*d;
        e[3]=b*c;
        e[4]=b*d;
        minf=e[1];
        maxf=e[1];
        for(int k=2;k<5;k++){
            if(minf>e[k])minf=e[k];
            if(maxf<e[k])maxf=e[k];
        }
    }
}

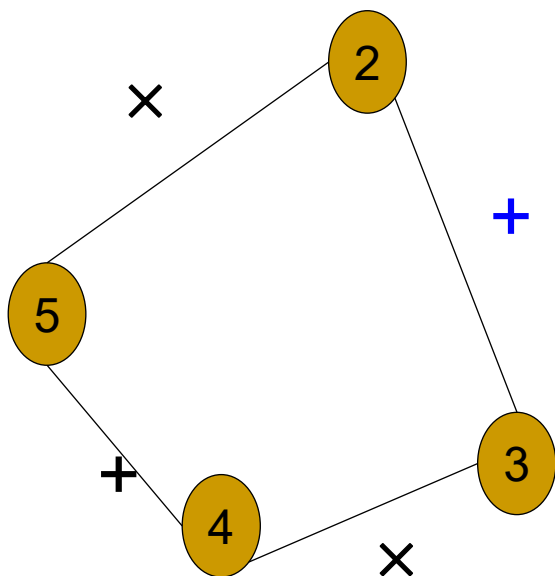
```

```

public static int polyMax()
{
    for (int j=2;j<=n;j++) // 链中有j个顶点
        for (int i=1;i<=n;i++) // 链的起点i
            for (int s=1;s<j;s++){ // 链中断开点s
                minMax(i,s,j);
                if(m[i][j][0]>minf) m[i][j][0]=minf;
                if(m[i][j][0]<maxf) m[i][j][1]=maxf;
            }
    int temp=m[1][n][1];
    for(int i=2;i<=n;i++)
        if(temp<m[i][n][1]) temp=m[i][n][1];
    return temp;
}

```





i

0

1

2

3

4

$m[i][j][0]/m[i][j][1]$

j

1

2

3

4

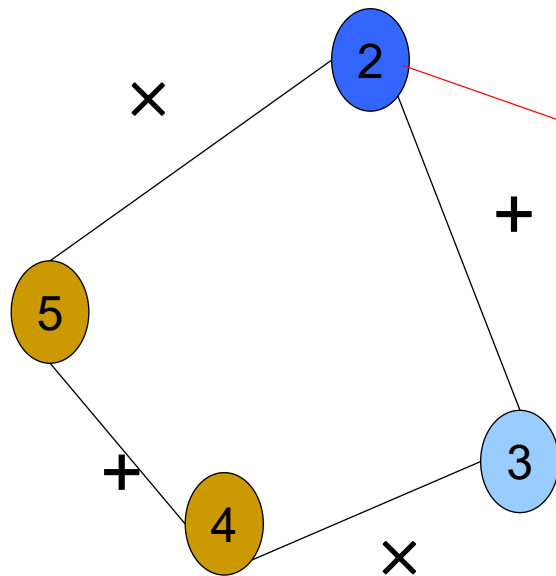
0	0	0	0	0
1	2/2	0/0	0/0	0/0
2	3/3	0/0	0/0	0/0
3	4/4	0/0	0/0	0/0
4	5/5	0/0	0/0	0/0

v

op

i

0	1	2	3	4
	2	3	4	5
+	x	+	x	+



$\text{minMax}(1,1,2)$

$a = m[i][s][0] = m[1][1][0] = 2$

$b = m[i][s][1] = m[1][1][1] = 2$

$r = (i + s - 1) \% n + 1 = 2$

$c = m[r][j - s][0] = m[2][1][0] = 3$

$d = m[r][j - s][1] = m[2][1][1] = 3$

$\text{op}[2] = '+'$

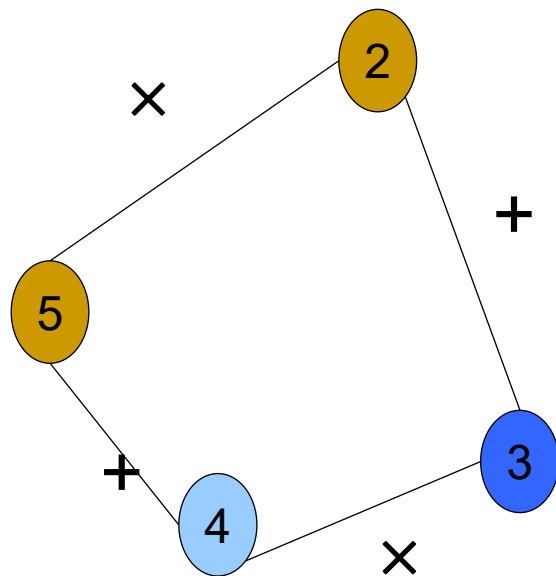
$a + c = 5 > m[1][2][0] = 0$

$\text{maxf} = b + d = 5$

$m[i][j][0] / m[i][j][1]$

		j	1	2	3	4
i	0	0	0	0	0	0
	1	0	2/2	0/5	0/0	0/0
	2	0	3/3	0/0	0/0	0/0
	3	0	4/4	0/0	0/0	0/0
	4	0	5/5	0/0	0/0	0/0

		0	1	2	3	4
v	i	0	1	2	3	4
		+	x	+	x	+



$\text{minMax}(2,1,2)$

$a = m[i][s][0] = m[2][1][0] = 3$

$b = m[i][s][1] = m[2][1][1] = 3$

$r = (i + s - 1) \% n + 1 = 3$

$c = m[r][j-s][0] = m[3][1][0] = 4$

$d = m[r][j-s][1] = m[3][1][1] = 4$

$\text{op}[3] = \text{'x'}$

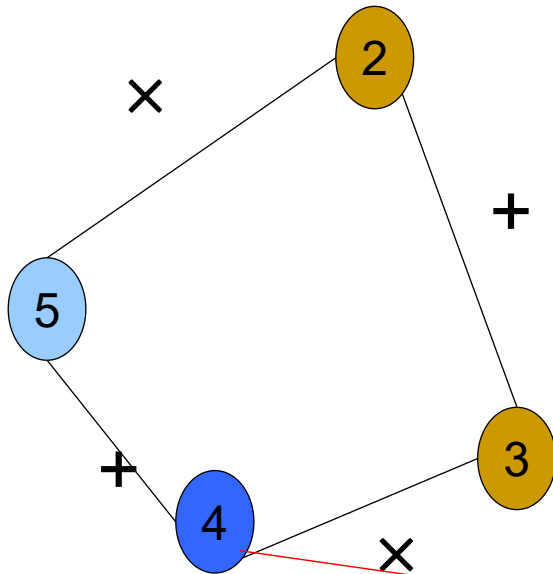
$a \times c = 12 > m[2][2][0] = 0$

$\text{maxf} = a \times c = 12$

$m[i][j][0] / m[i][j][1]$

		j				
		1	2	3	4	
i	0	0	0	0	0	0
	1	0	$2/2$	$0/5$	0/0	0/0
	2	0	$3/3$	$0/12$	0/0	0/0
	3	0	$4/4$	0/0	0/0	0/0
	4	0	$5/5$	0/0	0/0	0/0

v	i	0	1	2	3	4
		+	x	+	x	+



**minMax(3,1,2)**

$a = m[i][s][0] = m[3][1][0] = 4$

$b = m[i][s][1] = m[3][1][1] = 4$

$r = (i + s - 1) \% n + 1 = 4$

$c = m[r][j - s][0] = m[4][1][0] = 5$

$d = m[r][j - s][1] = m[4][1][1] = 5$

$op[3] == '+'$

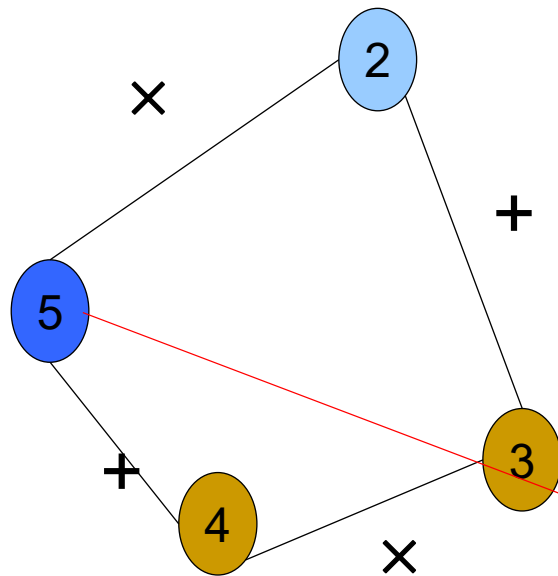
$a + c = 9 > m[3][2][0] = 0$

$maxf = a + c = 9$

$m[i][j][0] / m[i][j][1]$

		j	1	2	3	4
i	0	0	0	0	0	0
	1	0	2/2	0/5	0/0	0/0
	2	0	3/3	0/12	0/0	0/0
	3	0	4/4	0/9	0/0	0/0
	4	0	5/5	0/0	0/0	0/0

i	0	1	2	3	4
		2	3	4	5
	+	x	+	x	+
v					
op					



**minMax(4,1,2)**

$a = m[i][s][0] = m[4][1][0] = 5$

$b = m[i][s][1] = m[4][1][1] = 5$

$r = (i + s - 1) \% n + 1 = 1$

$c = m[r][j-s][0] = m[1][1][0] = 2$

$d = m[r][j-s][1] = m[1][1][1] = 2$

$op[1] == 'x'$

$a \times c = 10 > m[4][2][0] = 0$

$maxf = a \times c = 10$

$m[i][j][0] / m[i][j][1]$

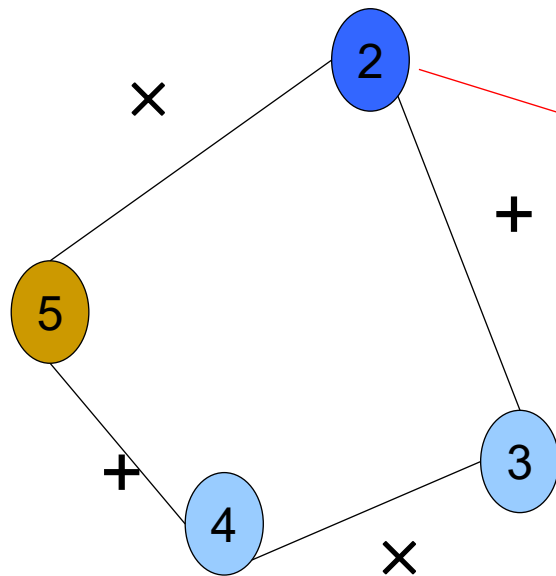
		j				
		1	2	3	4	
i	0	0	0	0	0	0
	1	0	2/2	0/5	0/0	0/0
	2	0	3/3	0/12	0/0	0/0
	3	0	4/4	0/9	0/0	0/0
	4	0	5/5	0/10	0/0	0/0

i	0	1	2	3	4
		2	3	4	5
	+	x	+	x	+

v

op





$m[i][j][0]/m[i][j][1]$

		j	1	2	3	4
i	0	0	0	0	0	0
	1	0	2/2	0/5	0/14	0
	2	0	3/3	0/12	0	0
	3	0	4/4	0/9	0	0
	4	0	5/5	0/10	0	0

$\text{minMax}(1,1,3)$

$a=m[1][1][0]=2$

$b=m[1][1][1]=2$

$r=(i+s-1)\%n+1=2$

$c=m[r][j-s][0]=m[2][2][0]=0$

$d=m[r][j-s][1]=m[2][2][1]=12$

$op[2]='+'$

$a+c=2$

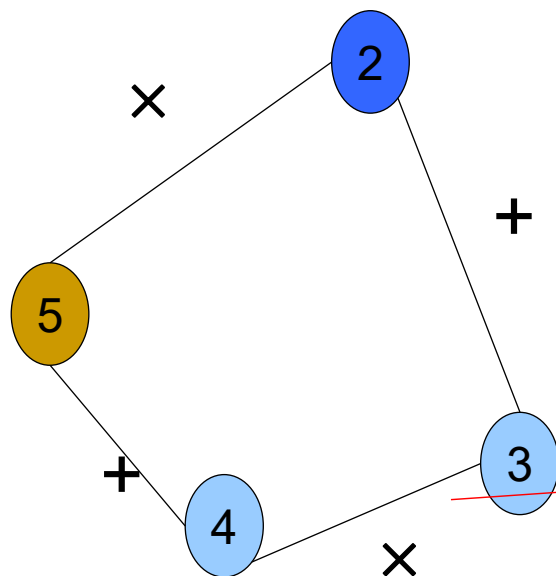
$\text{maxf}=b+d=14$

v

i

op

0	1	2	3	4
+	x	+	x	+



**minMax(1,2,3)**

$a = m[1][2][0] = 0$

$b = m[1][2][1] = 5$

$r = (i + s - 1) \% n + 1 = 3$

$c = m[r][j-s][0] = m[3][1][0] = 4$

$d = m[r][j-s][1] = m[3][1][1] = 4$

$op[3] == 'x'$

$a \times c = 0$

$maxf = b \times d = 5 \times 4 = 20$

v

i

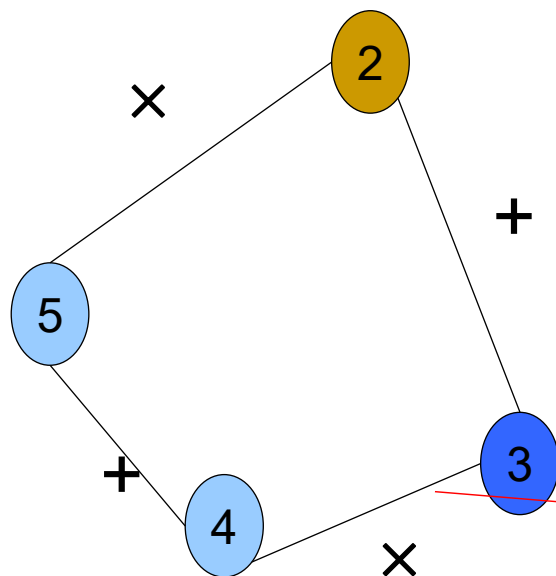
op

$m[i][j][0]/m[i][j][1]$

j      1      2      3      4

i	0	1	2	3	4
0	0	0	0	0	0
1	0	2/2	0/5	0/20	0
2	0	3/3	0/12	0	0
3	0	4/4	0/9	0	0
4	0	5/5	0/10	0	0

i	0	1	2	3	4
+	+	x	+	x	+



**minMax(2,1,3)**

$a = m[2][1][0] = 3$

$b = m[2][1][1] = 3$

$r = (i + s - 1) \% n + 1 = 3$

$c = m[r][j - s][0] = m[3][2][0] = 0$

$d = m[r][j - s][1] = m[3][2][1] = 9$

$op[3] == 'x'$

$a \times c = 0$

$maxf = b \times d = 3 \times 9 = 27$

v

i

op

$m[i][j][0]/m[i][j][1]$

j      1      2      3      4

0

0	0	0	0	0
1	0	2/2	0/5	0/20
2	0	3/3	0/12	0/27
3	0	4/4	0/9	0
4	0	5/5	0/10	0

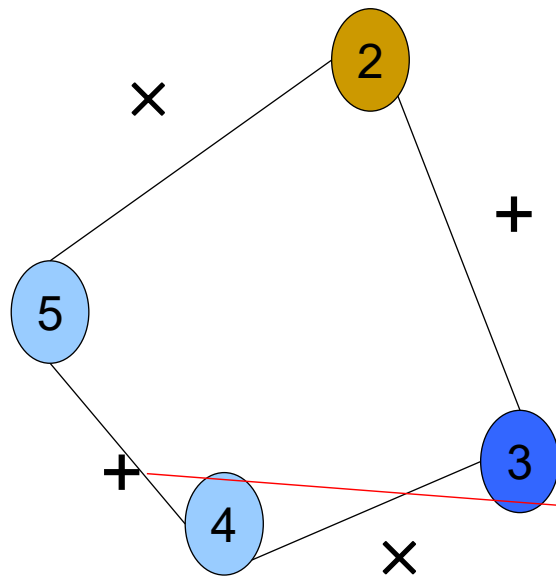
1

2

3

4

0	1	2	3	4
+	x	+	x	+



**minMax(2,2,3)**

$a = m[2][2][0] = 0$

$b = m[2][2][1] = 12$

$r = (i + s - 1) \% n + 1 = 4$

$c = m[r][j - s][0] = m[4][1][0] = 5$

$d = m[r][j - s][1] = m[4][1][1] = 5$

$op[4] == '+'$

$a + c = 5$

$maxf = 27 > b + d = 17$

v

op

$m[i][j][0]/m[i][j][1]$

j      1      2      3      4

i      0

0	0	0	0	0
1	0	2/2	0/5	0/20
2	0	3/3	0/12	0/27
3	0	4/4	0/9	0
4	0	5/5	0/10	0

1

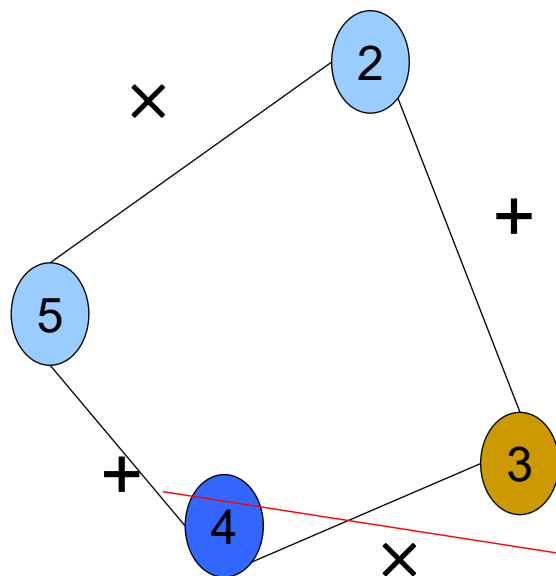
2

3

4

i

0	1	2	3	4
+	x	+	x	+



**minMax(3,1,3)**

$a = m[3][1][0] = 4$

$b = m[3][1][1] = 4$

$r = (i + s - 1) \% n + 1 = 4$

$c = m[r][j-s][0] = m[4][2][0] = 0$

$d = m[r][j-s][1] = m[4][2][1] = 10$

$op[4] == '+'$

$a + c = 4$

$maxf = b + d = 14$

i

0

1

2

3

4

v

i

op

$m[i][j][0]/m[i][j][1]$

j

1

2

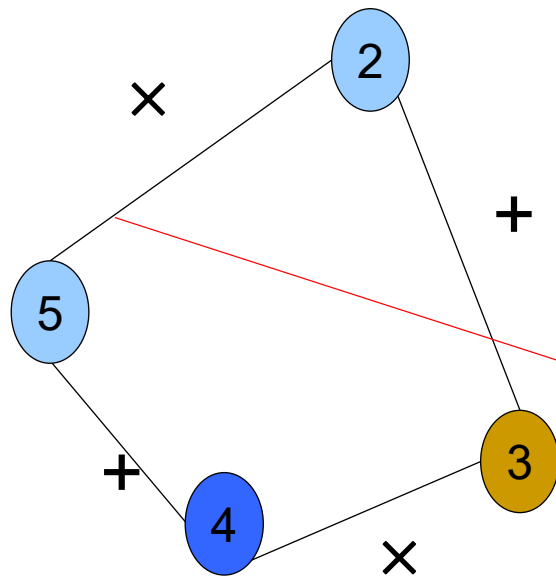
3

4

j	1	2	3	4
i	0	0	0	0
1	0	2/2	0/5	0/20
2	0	3/3	0/12	0/27
3	0	4/4	0/9	0/14
4	0	5/5	0/10	0

0	1	2	3	4
+	x	+	x	+





**minMax(3,2,3)**

$a = m[3][2][0] = 0$

$b = m[3][2][1] = 9$

$r = (i + s - 1) \% n + 1 = 1$

$c = m[r][j - s][0] = m[1][1][0] = 2$

$d = m[r][j - s][1] = m[1][1][1] = 2$

$op[1] == '×'$

$a × c = 0$

$maxf = 14 < b × d = 18$

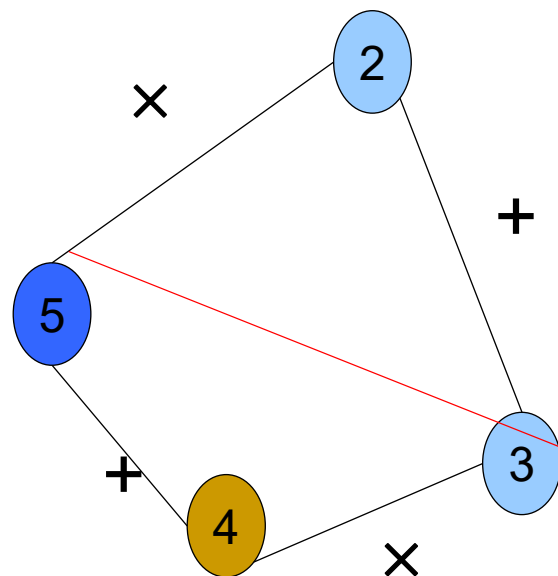
v

op

$m[i][j][0]/m[i][j][1]$

j	1	2	3	4
0	0	0	0	0
0	2/2	0/5	0/20	0
0	3/3	0/12	0/27	0
0	4/4	0/9	0/18	0
0	5/5	0/10	0	0

i	0	1	2	3	4
+	+	×	+	×	+



**minMax(4,1,3)**

$a = m[4][1][0] = 5$

$b = m[4][1][1] = 5$

$r = (i + s - 1) \% n + 1 = 1$

$c = m[r][j - s][0] = m[1][2][0] = 0$

$d = m[r][j - s][1] = m[1][2][1] = 5$

$op[1] = 'x'$

$a \times c = 0$

$maxf = b \times d = 25$

v

op

$m[i][j][0]/m[i][j][1]$

j      1      2      3      4

i      0

0	0	0	0	0
1	0	2/2	0/5	0/20
2	0	3/3	0/12	0/27
3	0	4/4	0/9	0/18
4	0	5/5	0/10	0/25

1

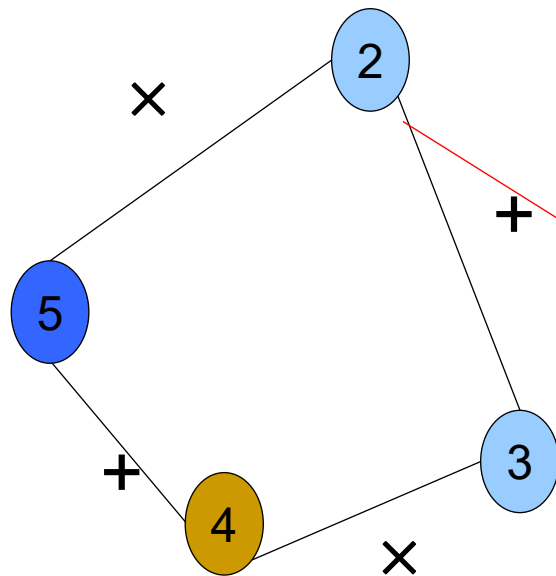
2

3

4

i

0	1	2	3	4
+	x	+	x	+



**minMax(4,2,3)**

$a = m[4][2][0] = 0$

$b = m[4][2][1] = 10$

$r = (i + s - 1) \% n + 1 = 2$

$c = m[r][j - s][0] = m[2][1][0] = 3$

$d = m[r][j - s][1] = m[2][1][1] = 3$

$op[2] == '+'$

$a + c = 3$

$Maxf = 25 > b + d = 13$

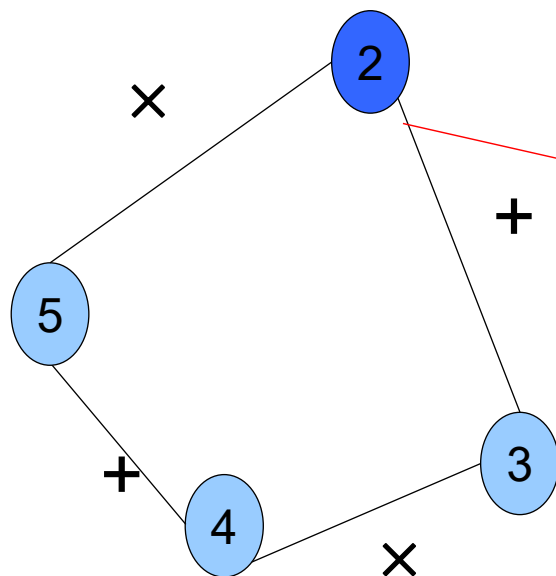
v

op

$m[i][j][0]/m[i][j][1]$

j	1	2	3	4
0	0	0	0	0
1	0	2/2	0/5	0/20
2	0	3/3	0/12	0/27
3	0	4/4	0/9	0/18
4	0	5/5	0/10	0/25

i	0	1	2	3	4
	+	×	+	×	+



**minMax(1,1,4)**

$a = m[1][1][0] = 2$

$b = m[1][1][1] = 2$

$r = (i + s - 1) \% n + 1 = 2$

$c = m[r][j - s][0] = m[2][3][0] = 0$

$d = m[r][j - s][1] = m[2][3][1] = 27$

$op[2] == '+'$

$a + c = 2$

$Maxf = b + d = 29$

i

0

1

2

3

4

v

i

op

$m[i][j][0]/m[i][j][1]$

j

1

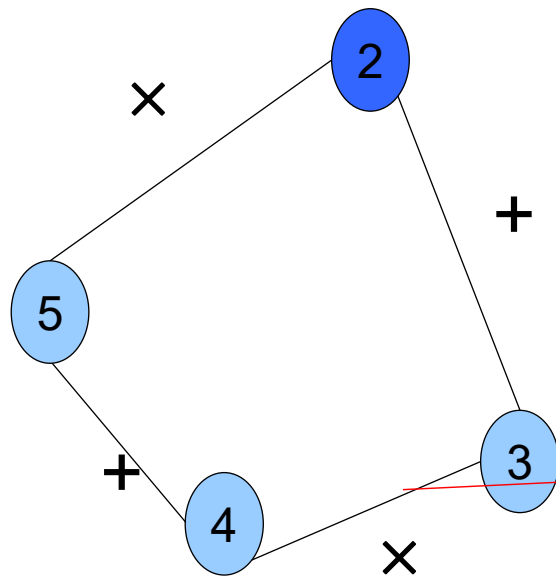
2

3

4

j	1	2	3	4
i	0	0	0	0
1	0	2/2	0/5	0/20
2	0	3/3	0/12	0/27
3	0	4/4	0/9	0/18
4	0	5/5	0/10	0/25

0	1	2	3	4
+	x	+	x	+



**minMax(1,2,4)**

$a = m[1][2][0] = 0$

$b = m[1][2][1] = 5$

$r = (i + s - 1) \% n + 1 = 3$

$c = m[r][j - s][0] = m[3][2][0] = 0$

$d = m[r][j - s][1] = m[3][2][1] = 9$

$op[3] == 'x'$

$a \times c = 2$

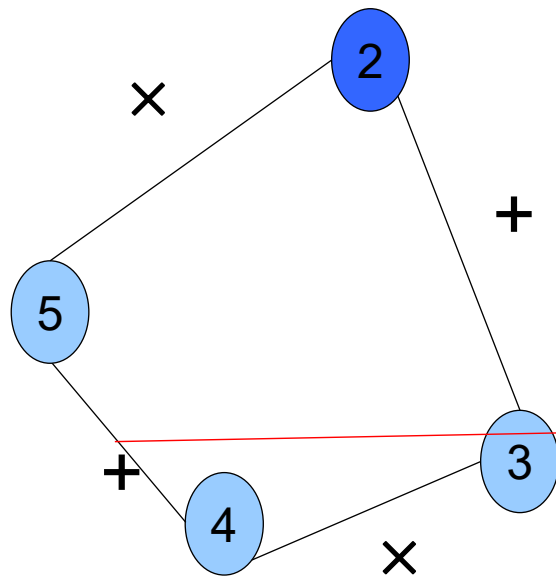
$Maxf = 29 < b \times d = 45$

$m[i][j][0]/m[i][j][1]$

		$m[i][j][0]/m[i][j][1]$				
		j	1	2	3	4
i	0	0	0	0	0	0
	1	0	2/2	0/5	0/20	0/45
	2	0	3/3	0/12	0/27	0
	3	0	4/4	0/9	0/18	0
	4	0	5/5	0/10	0/25	0

0	1	2	3	4
+	×	+	×	+





**minMax(1,3,4)**

$a = m[1][3][0] = 0$

$b = m[1][3][1] = 20$

$r = (i + s - 1) \% n + 1 = 4$

$c = m[r][j - s][0] = m[4][1][0] = 5$

$d = m[r][j - s][1] = m[4][1][1] = 5$

$op[4] == '+'$

$a + c = 5$

$Maxf = 45 > b + d = 25$

i

0

1

2

3

4

v

i

op

$m[i][j][0]/m[i][j][1]$

j

1

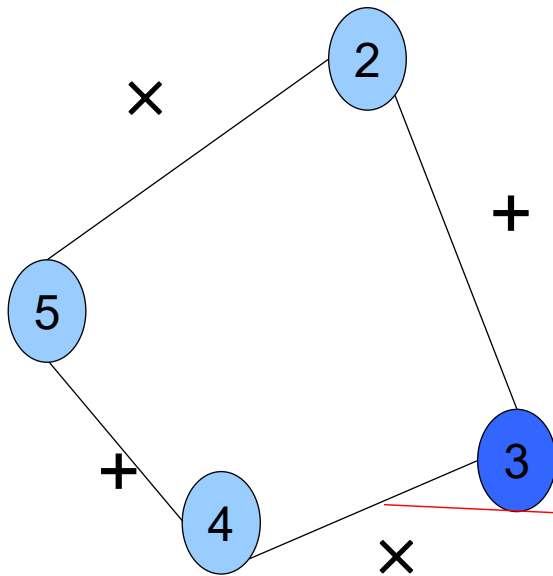
2

3

4

j	1	2	3	4
0	0	0	0	0
1	0	2/2	0/5	0/20
2	0	3/3	0/12	0/27
3	0	4/4	0/9	0/18
4	0	5/5	0/10	0/25

0	1	2	3	4
+	x	+	x	+



**minMax(2,1,4)**

$a = m[2][1][0] = 3$

$b = m[2][1][1] = 3$

$r = (i + s - 1) \% n + 1 = 3$

$c = m[r][j - s][0] = m[3][3][0] = 0$

$d = m[r][j - s][1] = m[3][3][1] = 18$

$op[3] = '×'$

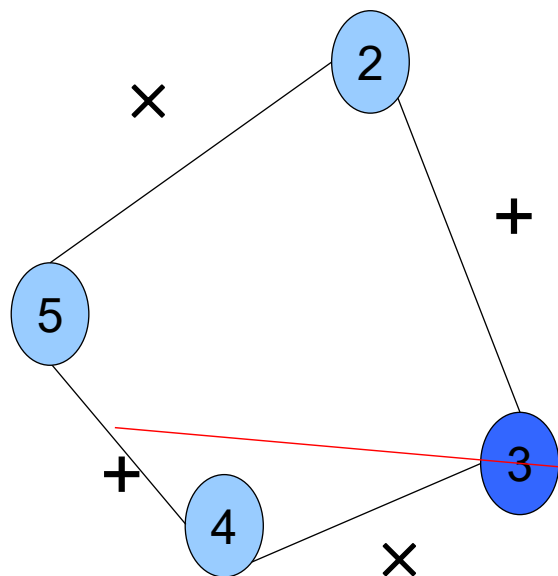
$a × c = 0$

$Maxf = b × d = 54$

$m[i][j][0]/m[i][j][1]$

		$m[i][j][0]/m[i][j][1]$				
		j	1	2	3	4
i	0	0	0	0	0	0
	1	0	2/2	0/5	0/20	0/45
	2	0	3/3	0/12	0/27	0/54
	3	0	4/4	0/9	0/18	0
	4	0	5/5	0/10	0/25	0

i	0	1	2	3	4
v	+	×	+	×	+



**minMax(2,2,4)**

$a = m[2][2][0] = 0$

$b = m[2][2][1] = 12$

$r = (i + s - 1) \% n + 1 = 4$

$c = m[r][j - s][0] = m[4][2][0] = 0$

$d = m[r][j - s][1] = m[4][2][1] = 10$

$op[4] == '+'$

$a + c = 0$

$Maxf = 54 > b + d = 22$

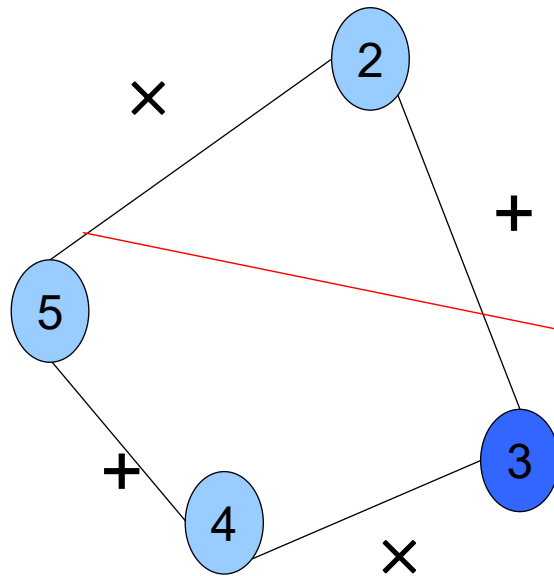
v

op

$m[i][j][0]/m[i][j][1]$

j	1	2	3	4
i				
0	0	0	0	0
1	0	2/2	0/5	0/20
2	0	3/3	0/12	0/54
3	0	4/4	0/9	0/18
4	0	5/5	0/10	0/25

i	0	1	2	3	4
v	+	x	+	x	+



**minMax(2,3,4)**

$a = m[2][3][0] = 0$

$b = m[2][3][1] = 27$

$r = (i + s - 1) \% n + 1 = 1$

$c = m[r][j - s][0] = m[1][1][0] = 2$

$d = m[r][j - s][1] = m[1][1][1] = 2$

$op[1] == '×'$

$a × c = 0$

$Maxf = 54 = b × d = 54$

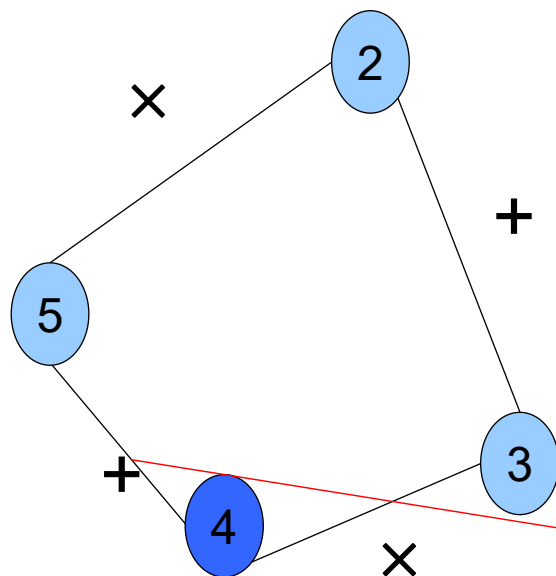
v

op

$m[i][j][0]/m[i][j][1]$

j	1	2	3	4
0	0	0	0	0
0	2/2	0/5	0/20	0/45
0	3/3	0/12	0/27	0/54
0	4/4	0/9	0/18	0
0	5/5	0/10	0/25	0

i	0	1	2	3	4
v	+	×	+	×	+



**minMax(3,1,4)**

$a = m[3][1][0] = 4$

$b = m[3][1][1] = 4$

$r = (i + s - 1) \% n + 1 = 4$

$c = m[r][j - s][0] = m[4][3][0] = 0$

$d = m[r][j - s][1] = m[4][3][1] = 25$

$op[4] == '+'$

$a + c = 4$

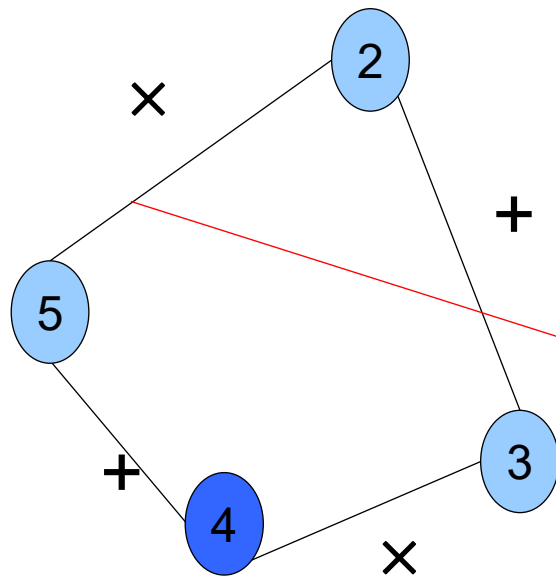
$Maxf = b + d = 29$

$m[i][j][0]/m[i][j][1]$

		$m[i][j][0]/m[i][j][1]$				
		j	1	2	3	4
i	0	0	0	0	0	0
	1	0	2/2	0/5	0/20	0/45
	2	0	3/3	0/12	0/27	0/54
	3	0	4/4	0/9	0/18	0/29
	4	0	5/5	0/10	0/25	0

0	1	2	3	4
+	×	+	×	+





**minMax(3,2,4)**

$a = m[3][2][0] = 0$

$b = m[3][2][1] = 9$

$r = (i + s - 1) \% n + 1 = 1$

$c = m[r][j - s][0] = m[1][2][0] = 0$

$d = m[r][j - s][1] = m[1][2][1] = 5$

$op[1] == '×'$

$a × c = 0$

$Maxf = 29 < b × d = 45$

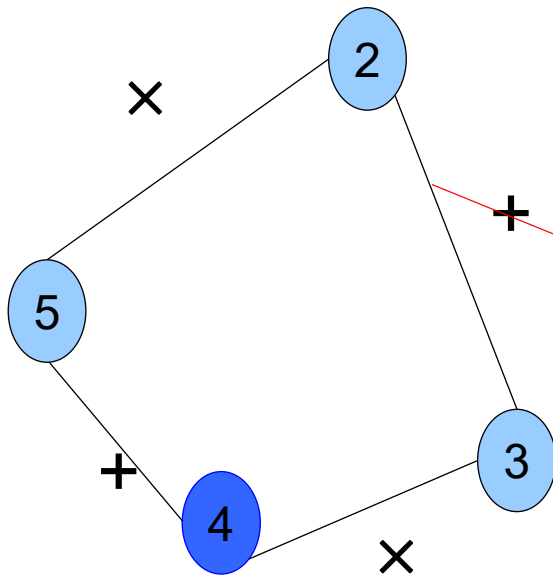
v

op

$m[i][j][0]/m[i][j][1]$

j	1	2	3	4
i				
0	0	0	0	0
1	0	2/2	0/5	0/20
2	0	3/3	0/12	0/54
3	0	4/4	0/9	0/18
4	0	5/5	0/10	0

i	0	1	2	3	4
v	+	×	+	×	+



**minMax(3,3,4)**

$a = m[3][3][0] = 4$

$b = m[3][3][1] = 4$

$r = (i + s - 1) \% n + 1 = 2$

$c = m[r][j - s][0] = m[2][1][0] = 2$

$d = m[r][j - s][1] = m[2][1][1] = 12$

$op[2] == '+'$

$a + c = 2$

$Maxf = 45 > b + d = 16$

v

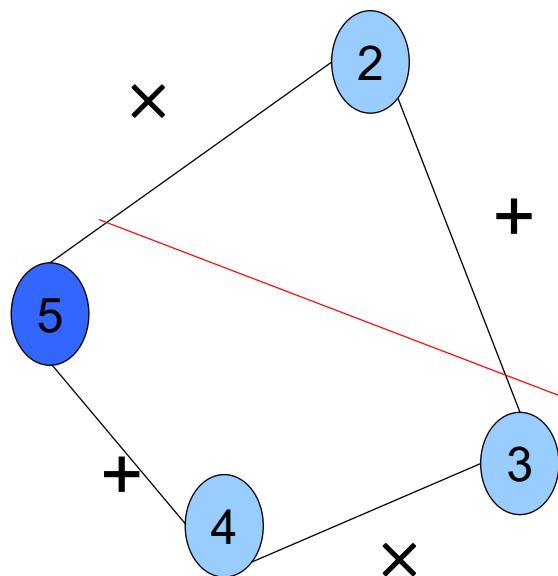
i

op

$m[i][j][0]/m[i][j][1]$

j	1	2	3	4
0	0	0	0	0
1	0	2/2	0/5	0/20
2	0	3/3	0/12	0/54
3	0	4/4	0/9	0/18
4	0	5/5	0/10	0/25

0	1	2	3	4
+	x	+	x	+



**minMax(4,1,4)**

$a = m[4][1][0] = 5$

$b = m[4][1][1] = 5$

$r = (i + s - 1) \% n + 1 = 1$

$c = m[r][j - s][0] = m[1][3][0] = 0$

$d = m[r][j - s][1] = m[1][3][1] = 20$

$op[1] == 'x'$

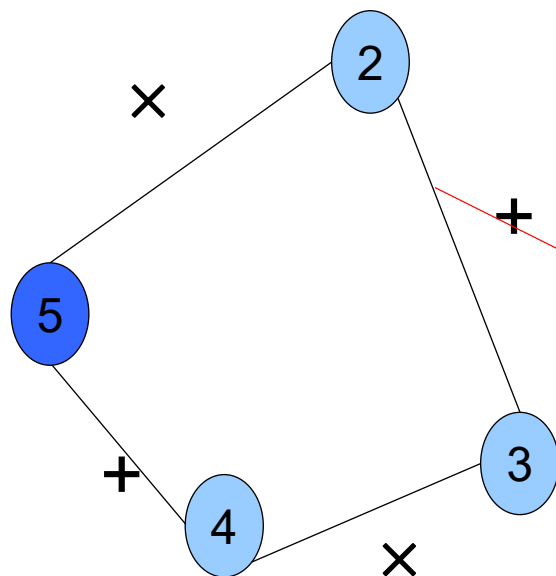
$a \times c = 0$

$Maxf = b \times d = 100$

$m[i][j][0]/m[i][j][1]$

j	1	2	3	4
0	0	0	0	0
0	2/2	0/5	0/20	0/45
0	3/3	0/12	0/27	0/54
0	4/4	0/9	0/18	0/45
0	5/5	0/10	0/25	0/100

i	0	1	2	3	4
op	+	x	+	x	+



**minMax(4,2,4)**

$a = m[4][2][0] = 0$

$b = m[4][2][1] = 10$

$r = (i + s - 1) \% n + 1 = 2$

$c = m[r][j - s][0] = m[2][2][0] = 0$

$d = m[r][j - s][1] = m[2][2][1] = 12$

$op[2] == '+'$

$a + c = 0$

$Maxf = 100 > b + d = 22$

v

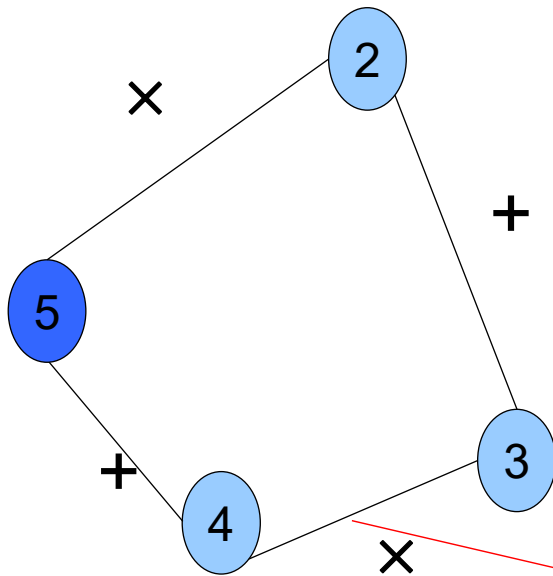
i

op

$m[i][j][0]/m[i][j][1]$

j	1	2	3	4
0	0	0	0	0
1	0	2/2	0/5	0/20
2	0	3/3	0/12	0/27
3	0	4/4	0/9	0/18
4	0	5/5	0/10	0/25

0	1	2	3	4
+	x	+	x	+



**minMax(4,3,4)**

$a = m[4][3][0] = 0$

$b = m[4][3][1] = 25$

$r = (i + s - 1) \% n + 1 = 3$

$c = m[r][j - s][0] = m[3][1][0] = 4$

$d = m[r][j - s][1] = m[3][1][1] = 4$

$op[3] == 'x'$

$a \times c = 0$

$Maxf = 100 = b \times d = 100$

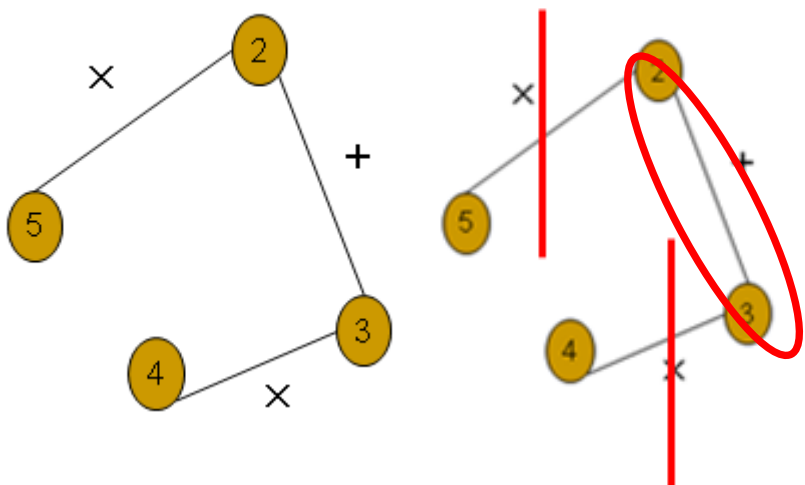
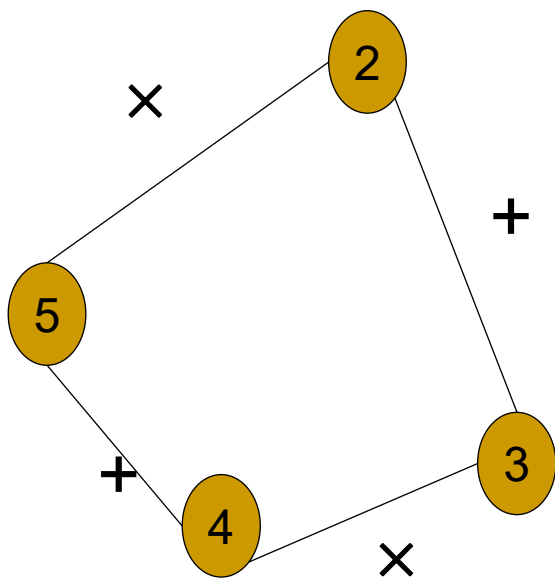
v

op

$m[i][j][0]/m[i][j][1]$

j	1	2	3	4
i				
0	0	0	0	0
1	0	2/2	0/5	0/20
2	0	3/3	0/12	0/27
3	0	4/4	0/9	0/18
4	0	5/5	0/10	0/25

i	0	1	2	3	4
v	+	x	+	x	+

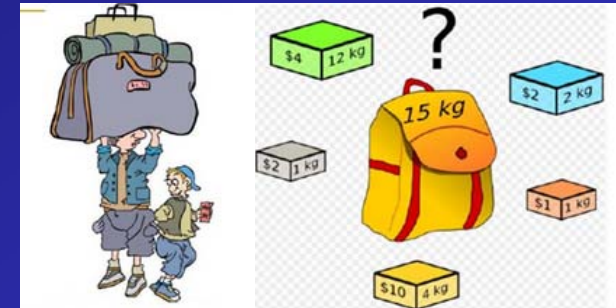


lo[i][j]记载s的值

j	1	2	3	4
0	0	0	0	0
1	0	1	2	2
2	0	1	1	1
3	0	1	2	2
4	0	1	1	1

## 3.10 0-1背包问题

- 引入：海盗问题
- 问题：



给定  $n$  种物品和一个背包。物品 $i$ 的重量是 $w_i$ ，其价值为 $v_i$ ，背包的容量为 $C$ 。问应如何选择装入背包的物品，使得装入背包中物品的总价值最大？

- 规定：
- 只有两种选择，装入为1，不装为0，因此称为0-1背包问题。注：不能装入多次，也不能只装入部分

## 3.10 0-1背包问题

➤ 表示:

价值总和最大

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

背包容量

$$\begin{cases} \sum_{i=1}^n w_i x_i \leq C \\ x_i \in \{0,1\}, 1 \leq i \leq n \end{cases}$$



## 3.10 0-1背包问题

### ➤ 最优子结构性质:

如果 $(x_1, x_2, \dots, x_n)$ 是所给0/1背包问题的一个最优解,

$$\begin{cases} \sum_{i=1}^n w_i x_i \leq C \\ x_i \in \{0,1\} \quad (1 \leq i \leq n) \end{cases}$$

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

若能证明 $(x_2, \dots, x_n)$ 是下面子问题的最优解:

$$\begin{cases} \sum_{i=2}^n w_i x_i \leq C - w_1 x_1 \\ x_i \in \{0,1\} \quad (2 \leq i \leq n) \end{cases}$$

$$\max \sum_{i=2}^n v_i x_i$$

则0/1背包问题具有最优子结构性质。

## 3.10 0-1背包问题

→ 证明0/1背包问题是最优子结构（反证）。

设 $(x_1, x_2, \dots, x_n)$ 是所给0/1背包问题的一个最优解，则 $(x_2, \dots, x_n)$ 是下面一个子问题的最优解：

$$\begin{cases} \sum_{i=2}^n w_i x_i \leq C - w_1 x_1 \\ x_i \in \{0, 1\} \quad (2 \leq i \leq n) \end{cases}$$

$$\max \sum_{i=2}^n v_i x_i$$

如若不然，设 $(y_2, \dots, y_n)$ 是上述子问题的一个最优解，则

$$\sum_{i=2}^n v_i y_i > \sum_{i=2}^n v_i x_i$$

$$w_1 x_1 + \sum_{i=2}^n w_i y_i \leq C$$

因此，

$$v_1 x_1 + \sum_{i=2}^n v_i y_i > v_1 x_1 + \sum_{i=2}^n v_i x_i = \sum_{i=1}^n v_i x_i$$

这说明 $(x_1, y_2, \dots, y_n)$ 是所给0/1背包问题比 $(x_1, x_2, \dots, x_n)$ 更优的解，从而导致矛盾。

## 3.10 0-1背包问题

### ➤ 重叠子问题性质：

实例：  $C=10$ ，  $W=\{ 7, 3, 4, 5 \}$ ，  $V=\{ 42, 12, 40, 25 \}$

子集	总重量	总价值
$\phi$	0	\$0
{1}	7	\$42
{2}	3	\$12
{3}	4	\$40
{4}	5	\$25
{1, 2}	10	\$54
{1, 3}	11	不可行
{1, 4}	12	不可行
{2, 3}	7	\$52
{2, 4}	8	\$37
{3, 4}	9	<b>\$65</b>

## 3.10 0-1背包问题

0/1背包问题如何用动态规划法解决？

**关键问题：找出动态规划函数**

□ 0/1背包问题可以看作是决策一个序列 $(x_1, x_2, \dots, x_n)$ ，对任一变量 $x_i$ 的决策是决定 $x_i=1$ 还是 $x_i=0$ 。在对 $x_{i-1}$ 决策后，已确定了 $(x_1, \dots, x_{i-1})$ ，在决策 $x_i$ 时，问题处于下列两种状态之一：

- (1) 背包容量不足以装入物品 $i$ ，则 $x_i=0$ ，背包不增加价值；
- (2) 背包容量可以装入物品 $i$ 。

在(2)的状态下，物品 $i$ 有两种情况，装入（则 $x_i=1$ ）或不装入（则 $x_i=0$ ）。在这两种情况下背包价值的最大者应该是对 $x_i$ 决策后的背包价值。

## 3.10 0-1背包问题

令  $V(i, j)$  表示在前  $i (1 \leq i \leq n)$  个物品中能够装入容量为  $j$  ( $1 \leq j \leq C$ ) 的背包中的物品的最大值，则可以得到如下动态规划函数：

$$V(i, 0) = V(0, j) = 0$$

$$V(i, j) = \begin{cases} V(i-1, j) & j < w_i \\ \max\{V(i-1, j), V(i-1, j-w_i) + v_i\} & j \geq w_i \end{cases}$$

表明：把前面  $i$  个物品装入容量为 0 的背包和把 0 个物品装入容量为  $j$  的背包，得到的价值均为 0。

## 3.10 0-1背包问题

$$V(i, j) = \begin{cases} V(i-1, j) & j < w_i \\ \max\{V(i-1, j), V(i-1, j-w_i) + v_i\} & j \geq w_i \end{cases}$$

(1) 第一个式子表明：如果第 $i$ 个物品的重量大于背包的容量，则物品 $i$ 不能装入背包，则装入前 $i$ 个物品得到的最大价值和装入前 $i-1$ 个物品得到的最大价值是相同的。

## 3.10 0-1背包问题

$$V(i, j) = \begin{cases} V(i-1, j) & j < w_i \\ \max\{V(i-1, j), V(i-1, j-w_i) + v_i\} & j \geq w_i \end{cases}$$

(2) 第二个式子表明：如果第 $i$ 个物品的重量小于背包的容量，则会有以下两种情况：

- ① 如果第 $i$ 个物品没有装入背包，则背包中物品的价值就等于把前 $i-1$ 个物品装入容量为 $j$ 的背包中所取得的价值。
- ② 如果把第 $i$ 个物品装入背包，则背包中物品的价值等于把前 $i-1$ 个物品装入容量为 $j-w_i$ 的背包中的价值加上第 $i$ 个物品的价值 $v_i$ ；

显然，取二者中价值较大者作为把前 $i$ 个物品装入容量为 $j$ 的背包中的最优解。

## 3.10 0-1背包问题

按下述方法来划分阶段：第一阶段，只装入前1个物品，确定在各种情况下的背包能够得到的最大价值；第二阶段，只装入前2个物品，确定在各种情况下的背包能够得到的最大价值；依此类推，直到第 $n$ 个阶段。最后， $V(n, C)$ 便是在容量为 $C$ 的背包中装入 $n$ 个物品时取得的最大价值。

为了确定装入背包的具体物品，从 $V(n, C)$ 的值向前推，如果 $V(n, C) > V(n-1, C)$ ，表明第 $n$ 个物品被装入背包，前 $n-1$ 个物品被装入容量为 $C-w_n$ 的背包中；否则，第 $n$ 个物品没有被装入背包，前 $n-1$ 个物品被装入容量为 $C$ 的背包中。依此类推，直到确定第1个物品是否被装入背包中为止。由此，得到如下函数：

$$x_i = \begin{cases} 0 & V(i, j) = V(i-1, j) \\ 1, & j = j - w_i \quad V(i, j) > V(i-1, j) \end{cases}$$



**实例：**有5个物品，其重量分别是{2, 2, 6, 5, 4}，价值分别为{6, 3, 5, 4, 6}，背包的容量为10。

根据动态规划函数，用一个 $(n+1) \times (C+1)$ 的二维表V， $V[i][j]$ 表示把前i个物品装入容量为j的背包中获得的最大价值。

$$V(i, j) = \begin{cases} V(i-1, j) & j < w_i \\ \max \{V(i-1, j), V(i-1, j - w_i) + v_i\} & j \geq w_i \end{cases}$$


		0	1	2	3	4	5	6	7	8	9	10	
	0	0	0	0	0	0	0	0	0	0	0	0	
$w_1=2 \ v_1=6$	1	0	0	6	6	6	6	6	6	6	6	6	$x_1=1$
$w_2=2 \ v_2=3$	2	0	0	6	6	9	9	9	9	9	9	9	$x_2=1$
$w_3=6 \ v_3=5$	3	0	0	6	6	9	9	9	9	11	11	14	$x_3=0$
$w_4=5 \ v_4=4$	4	0	0	6	6	9	9	9	10	11	13	14	$x_4=0$
$w_5=4 \ v_5=6$	5	0	0	6	6	9	9	12	12	15	15	15	$x_5=1$



## 3.10 0-1背包问题

### 算法实现

设 $n$ 个物品的重量存储在数组 $w[n]$ 中，价值存储在数组 $v[n]$ 中，背包容量为 $C$ ，数组 $V[n+1][C+1]$ 存放迭代结果，其中 $V[i][j]$ 表示前 $i$ 个物品装入容量为 $j$ 的背包中获得的最大价值，数组 $x[n]$ 存储装入背包的物品，动态规划法求解0/1背包问题的算法如下：

 算法——0/1背包问题

```
int KnapSack(int n, int w[ ], int v[ ]) {  
    for (i=0; i<=n; i++) //初始化第0列  
        V[i][0]=0;  
    for (j=0; j<=C; j++) //初始化第0行  
        V[0][j]=0;
```

## 3.10 0-1背包问题

C++描述

### 算法——0/1背包问题

```
for (i=1; i<=n; i++) //计算第i行，进行第i次迭代
    for (j=1; j<=C; j++)
        if (j<w[i]) V[i][j]=V[i-1][j];
        else V[i][j]=max(V[i-1][j], V[i-1][j-w[i]]+v[i]);
```

j=C; //求装入背包的物品

```
for (i=n; i>0; i--){
    if (V[i][j]>V[i-1][j]) {
        x[i]=1;
        j=j-w[i];
    }
    else x[i]=0;
}
return V[n][C]; //返回背包取得的最大价值
```

```
}
```

## 3.10 0-1背包问题

0/1背包问题用动态规划法，与其他方法相比，效率如何，效果如何？

- ❑ 在算法中，第一个for循环的时间性能是 $O(n)$ ，第二个for循环的时间性能是 $O(C)$ ，第三个循环是两层嵌套的for循环，其时间性能是 $O(n \times C)$ ，第四个for循环的时间性能是 $O(n)$ ，所以，算法的时间复杂性为 $O(n \times C)$ ，一定能求得最优解。
- ❑ 蛮力法： $O(2^n)$ ，一定能求得最优解
- ❑ 贪心法： $O(n)$ ，不一定能求得最优解

## 0-1背包问题练习题:

$$V(i, j) = \begin{cases} V(i-1, j) & j < w_i \\ \max\{V(i-1, j), V(i-1, j-w_i) + v_i\} & j \geq w_i \end{cases}$$

		0	$j-w_i$	$j$	$W$
$w_i, v_i$	0	0	0	0	0
	$i-1$	0	$v[i-1, j-w_i]$	$v[i-1, j]$	
	$i$	0		$v[i, j]$	
	$n$	0			目标

实例：

$$V(i, j) = \begin{cases} V(i-1, j) & j < w_i \\ \max\{V(i-1, j), V(i-1, j-w_i) + v_i\} & j \geq w_i \end{cases}$$

物品	重量	价值	承重量
1	2	\$12	W=5
2	1	\$10	
3	3	\$20	
4	2	\$15	

w <sub>i</sub>	v <sub>i</sub>	i \ j	0	1	2	3	4	5
		0	0	0	0	0	0	0
2	12	1	0	0	0	0	0	0
1	10	2	0	0	0	0	0	0
3	20	3	0	0	0	0	0	0
2	15	4	0	0	0	0	0	0

w <sub>i</sub>	v <sub>i</sub>	$\begin{array}{c} j \\ \diagdown \\ i \end{array}$	0	1	2	3	4	5
		0	0	0	0	0	0	0
2	12	1	0	0	12	12	12	12
1	10	2	0	0	0	0	0	0
3	20	3	0	0	0	0	0	0
2	15	4	0	0	0	0	0	0

$$v[1][0] \sim v[1][1] = 0, \quad v[1][2] \sim v[1][5] = 12$$

$$V(i, j) = \begin{cases} V(i-1, j) & j < w_i \\ \max\{V(i-1, j), V(i-1, j-w_i) + v_i\} & j \geq w_i \end{cases}$$

w <sub>i</sub>	v <sub>i</sub>	i \ j	0	1	2	3	4	5
		0	0	0	0	0	0	0
2	12	1	0	0	12	12	12	12
1	10	2	0	10	12	22	22	22
3	20	3	0	0	0	0	0	0
2	15	4	0	0	0	0	0	0

$v[2][0] = 0,$    
  $v[2][1] = 10,$    
 $v[2][2] = 12,$    
 $v[2][3] \sim v[2][5] = 22$

$$V(i, j) = \begin{cases} V(i-1, j) & j < w_i \\ \max\{V(i-1, j), V(i-1, j-w_i) + v_i\} & j \geq w_i \end{cases}$$



w <sub>i</sub>	v <sub>i</sub>	$\begin{array}{c} j \\ \diagdown \\ i \end{array}$	0	1	2	3	4	5
		0	0	0	0	0	0	0
2	12	1	0	0	12	12	12	12
1	10	2	0	10	12	22	22	22
3	20	3	0	10	12	22	30	32
2	15	4	0	0	0	0	0	0

$v[3][0] = 0, \quad v[3][1] = 10, \quad v[3][2] = 12, \quad v[3][3] = 22$

$v[3][4] = 30, \quad v[3][5] = 32$

$$V(i, j) = \begin{cases} V(i-1, j) & j < w_i \\ \max\{V(i-1, j), V(i-1, j-w_i) + v_i\} & j \geq w_i \end{cases}$$

w <sub>i</sub>	v <sub>i</sub>	$\begin{array}{c} i \backslash j \\ 0 \end{array}$	0	1	2	3	4	5
		0	0	0	0	0	0	0
2	12	1	0	0	12	12	12	12
1	10	2	0	10	12	22	22	22
3	20	3	0	10	12	22	30	32
2	15	4	0	10	15	25	30	37

$v[4][0] = 0, \quad v[4][1] = 10, \quad v[4][2] = 15, \quad v[4][3] = 25$

$v[4][4] = 30, \quad v[4][5] = 37$

$$V(i, j) = \begin{cases} V(i-1, j) & j < w_i \\ \max\{V(i-1, j), V(i-1, j-w_i) + v_i\} & j \geq w_i \end{cases}$$

## 最优子集的组成元素：

$v[4,5] \neq v[3,5]$ ,  $i_4$ 在最优解中。  $j=5-2$

$v[3,3] = v[2,3]$

$v[2,3] \neq v[1,3]$ ,  $i_2$ 在最优解中。  $j=3-1$

$v[1,2] \neq v[0,2]$ ,  $i_1$ 在最优解中。  $j=2-2$

$w_i$	$v_i$	$i \backslash j$	0	1	2	3	4	5
		0	0	0	0	0	0	0
2	12	1	0	0	12	12	12	12
1	10	2	0	10	12	22	22	22
3	20	3	0	10	12	22	30	32
2	15	4	0	10	15	25	30	37

## 递推关系式(向前)

设所给0-1背包问题的子问题

$$\begin{cases} \max \sum_{k=i}^n v_k x_k \\ \sum_{k=i}^n w_k x_k \leq j \\ x_k \in \{0,1\}, i \leq k \leq n \end{cases}$$

的最优值为 $m(i, j)$ , 即 $m(i, j)$ 是背包容量为 $j$ , 可选择物品为 $i, i+1, \dots, n$ 时0-1背包问题的最优值。由0-1背包问题的最优子结构性质, 可以建立计算 $m(i, j)$ 的递归式如下。

$$m(i, j) = \begin{cases} \max \{m(i+1, j), m(i+1, j-w_i) + v_i\} & j \geq w_i \\ m(i+1, j) & 0 \leq j < w_i \end{cases}$$

$$m(n, j) = \begin{cases} v_n & j \geq w_n \\ 0 & 0 \leq j < w_n \end{cases}$$

例子：

物品	重量	价值	承重量
1	2	\$12	W=5
2	1	\$10	
3	3	\$20	
4	2	\$15	

w <sub>i</sub>	v <sub>i</sub>	$\begin{matrix} j \\ i \end{matrix}$	0	1	2	3	4	5
		0	0	0	0	0	0	0
2	12	1	0	0	0	0	0	0
1	10	2	0	0	0	0	0	0
3	20	3	0	0	0	0	0	0
2	15	4	0	0	0	0	0	0

w <sub>i</sub>	v <sub>i</sub>	$\begin{matrix} j \\ i \end{matrix}$	0	1	2	3	4	5
		0	0	0	0	0	0	0
2	12	1	0	0	0	0	0	0
1	10	2	0	0	0	0	0	0
3	20	3	0	0	0	0	0	0
2	15	4	0	0	15	15	15	15

$m[4][0] \sim m[4][1] = 0, \quad m[4][2] \sim m[4][5] = 15$

$$m(n, j) = \begin{cases} v_n & j \geq w_n \\ 0 & 0 \leq j < w_n \end{cases}$$

w <sub>i</sub>	v <sub>i</sub>	$\begin{matrix} j \\ i \end{matrix}$	0	1	2	3	4	5
		0	0	0	0	0	0	0
2	12	1	0	0	0	0	0	0
1	10	2	0	0	0	0	0	0
3	20	3	0	0	15	20	20	35
2	15	4	0	0	15	15	15	15

$m[3][0] \sim m[3][2] = 0, m[3][3] \sim m[3][4] = 20$

$m[3][5] = 35$

$$m(i, j) = \begin{cases} \max\{m(i+1, j), m(i+1, j - w_i) + v_i\} & j \geq w_i \\ m(i+1, j) & 0 \leq j < w_i \end{cases}$$

w <sub>i</sub>	v <sub>i</sub>	$\begin{array}{c} j \\ \diagdown \\ i \end{array}$	0	1	2	3	4	5
		0	0	0	0	0	0	0
2	12	1	0	0	0	0	0	0
1	10	2	0	10	15	25	30	35
3	20	3	0	0	15	20	20	35
2	15	4	0	0	15	15	15	15

$m[2][0] = 0, \quad m[2][1] = 10, \quad m[2][2] = 15, \quad m[2][3] = 25$

$m[2][4] \sim m[2][5] = 30$

$$m(i, j) = \begin{cases} \max\{m(i+1, j), m(i+1, j - w_i) + v_i\} & j \geq w_i \\ m(i+1, j) & 0 \leq j < w_i \end{cases}$$



w <sub>i</sub>	v <sub>i</sub>	$\begin{array}{c} j \\ \backslash \\ i \end{array}$	0	1	2	3	4	5
		0	0	0	0	0	0	0
2	12	1	0	10	15	25	30	37
1	10	2	0	10	15	25	30	35
3	20	3	0	0	15	20	20	35
2	15	4	0	0	15	15	15	15

$m[1][0] = 0, \quad m[1][1] = 10, \quad m[1][2] = 15, \quad m[1][3] = 25$

$m[1][4] = 27, \quad m[1][5] = 37$

$$m(i, j) = \begin{cases} \max\{m(i+1, j), m(i+1, j - w_i) + v_i\} & j \geq w_i \\ m(i+1, j) & 0 \leq j < w_i \end{cases}$$

## 最优子集的组成元素：

$m[1,5] \neq m[2,5]$ ,  $i_1$ 在最优解中。  $j=5-2$

$m[2,3] \neq m[3,3]$ ,  $i_2$ 在最优解中。  $j=3-1$

$m[3,2] = m[4,2]$

$m[4,2] > 0$ ,  $i_4$ 在最优解中。  $j=2-2$

$w_i$	$v_i$	$i \backslash j$	0	1	2	3	4	5
		0	0	0	0	0	0	0
2	12	1	0	10	15	25	30	37
1	10	2	0	10	15	25	30	35
3	20	3	0	0	15	20	20	35
2	15	4	0	0	15	15	15	15

## 0-1背包问题的算法复杂度分析：

空间：

$w[n]$ ,  $v[n]$ ,  $m[n][c]$ ,  $O(nc)$

时间：

由 $m(i, j)$ 的递归式，算法需要 $O(nc)$ 计算时间。

构造最优解时间： $O(n)$ 。

当背包容量 $c$ 很大时，算法需要的计算时间较多。

例如，当 $c > 2^n$ 时，算法需要 $\Omega(n2^n)$ 计算时间。