动态规划

任务：

P是出发点，从P到A，求最短路径（图1）

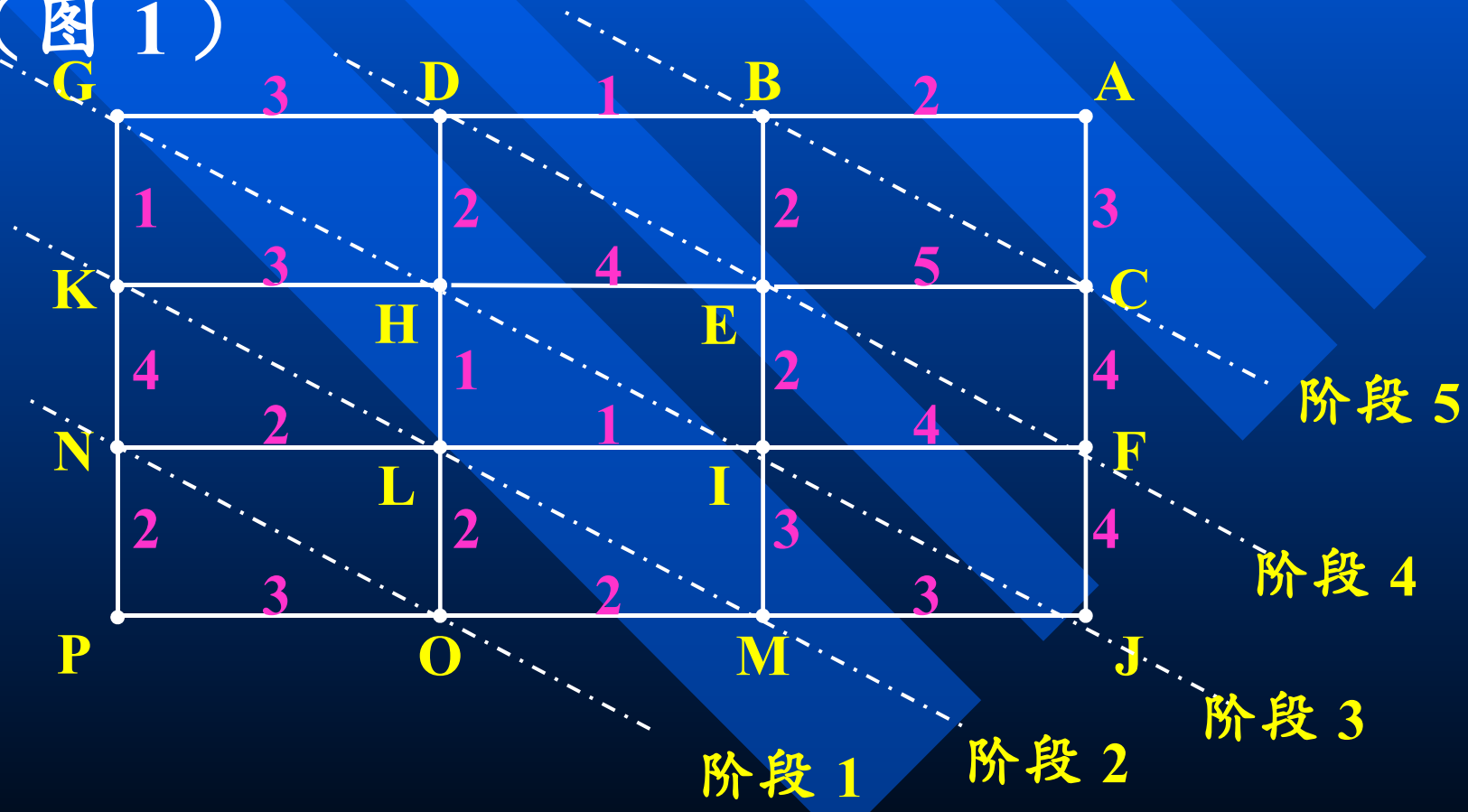


图 1

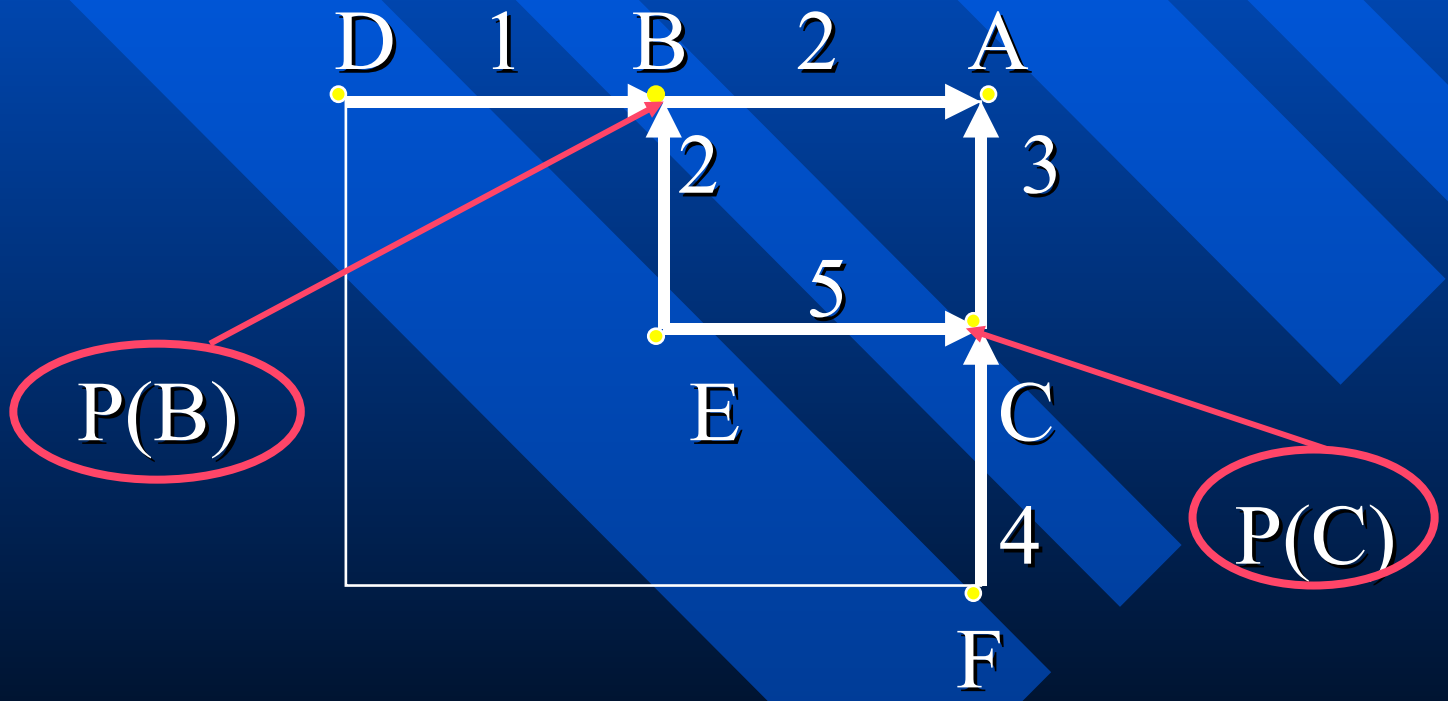
思路

1. 先看第 5 阶段，到达 A 点有两条路
   - B → A，需要 2km
   - C → A，需要 3km
2. 令
   - 从 P → A 的最短路径为 P(A)；
   - 从 P → B 的最短路径为 P(B)；
   - 从 P → C 的最短路径为 P(C) ……
   - P(A) = min{P(B)+**2**, P(C)+**3**}；
   - P(B) = min{P(D)+**1**, P(E)+**2**}；
   - P(C) = min{P(E)+**5**, P(F)+**4**}；

$P(A) = \min\{P(B)+2,\ P(C)+3\}$;

$P(B) = \min\{P(D)+1,\ P(E)+2\}$;

$P(C) = \min\{P(E)+5,\ P(F)+4\}$;



⋮

P(N) = 2;

P(O) = 3;

上述递推公式告诉我们，要求 P(A) 需要先求出阶段 5 中的 P(B) 和 P(C)；要求 P(B)（或者 P(C)），又要先求出阶段 4 中的 P(D) 和 P(E)（或 P(F) 和 P(E))……

显然，要依照上述递推过程求解，需要倒过来，从 P(P) 出发，先求出第一阶段的 P(O) 和 P(N)，再求第二阶段的 P(K)，P(L)，P(M)；……，最后得到 P(A)。

1. 选择数据结构，将每条路经的长度存在数组中。

东西方向上的道路长度存在两维数组 h[4][3] 中规定数组的第一维为行号，第二维为列号。

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

h[4][3] = { {3,2,3}, {2,1,4}, {3,4,5}, {3,1,2} };

南北方向上道路长度存至数组 v[3][4] 中，也规定第一维为行号，第二维为列号。

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

v[3][4] = {{2, 2, 3, 4},{4, 1, 2, 4},{1, 2, 2, 3}};

1. 为了计算方便，将图 1 改为图 2

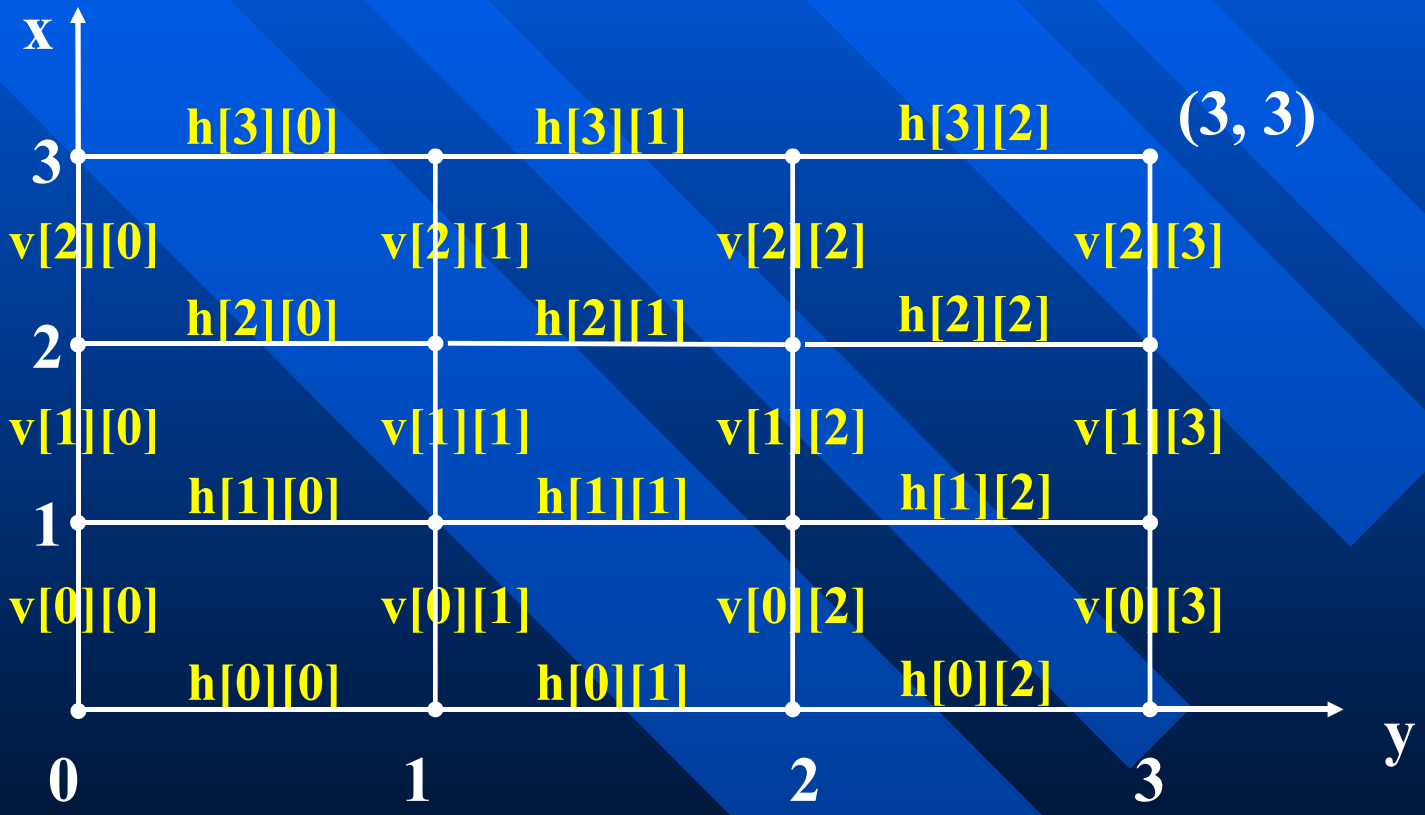


图 2

1. 求解过程为从 (0, 0) 到 (3, 3) 求最短路径问题

定义二维数组， P[4][4]={{0,0,0,0},{0,0,0,0},{0,0,0,0},{0,0,0,0}}, 第一维为行，第二维为列。这时 P(O) 为 P[0][1]； P(N) 为 P[1][0]； … P(A) 为 P[3][3]，以下为分阶段递推求解过程。

P[0][0] = 0;

对于阶段 1:

P[0][1] = P[0][0]+h[0][0] = 0+3 = 3;

P[1][0] = P[0][0]+v[0][0] = 0+2 = 3;

对于阶段 2

P[1][1] = min{ P[0][1]+v[0][1], P[1][0]+h[1][0]}

= min{3+1, 2+2} = 4

P[0][2] = P[0][1]+h[1][0] = 3+2 = 5

P[2][0] = P[1][0]+v[1][0] = 2+4 = 6

对于阶段 3

P[1][2] = min{ P[0][2]+v[0][2], P[1][1]+h[1][1]}

= min{5+3, 4+1} = 5

P[0][3] = P[0][2]+h[0][2] = 5+3 = 8

P[2][1] = min{ P[1][1]+v[1][1], P[2][0]+h[2][0]}

= min{4+1, 6+1} = 5

P[3][0] = P[2][0]+v[2][0] = 6+1 = 7

对于阶段 4

P[1][3] = min{ P[0][3]+v[0][3], P[1][2]+h[1][2]}

= min{8+4, 5+4} = 9

P[2][2] = min{ P[1][2]+v[1][2], P[2][1]+h[2][1]}

= min{5+2, 5+4} = 7

P[3][1] = min{ P[2][1]+v[2][1], P[3][0]+h[3][0]}

= min{5+2, 7+3} = 7

对于阶段 5

P[2][3] = min{ P[1][3]+v[1][3], P[2][2]+h[2][2]}

= min{9+4, 7+5} = 12

P[3][2] = min{ P[2][2]+v[2][2], P[3][1]+h[3][1]}

= min{7+2, 7+1} = 8

最后

P[3][3] = min{ P[2][3]+v[2][3], P[3][2]+h[3][2]}

= min{12+3, 8+2} = 10

综上，数组 P 的通项表示为

P[i][j]= min((p[i-1][j]+v[i-1][j]), (p[i][j-1]+h[i][j-1])) (i, j>0)

P[0][j]=P[0][j-1]+h[0][j-1] (i=0, j>0)

P[i][0]=P[i-1][0]+v[i-1][0] (i>0, j=0)

下面给出 P 数组中的数据

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

数组 P 的通项表示为

$$P[i][j]= \min((p[i-1][j]+v[i-1][j]), (p[i][j-1]+h[i][j-1])) \quad (i, j>0)$$

$$P[0][j]=P[0][j-1]+h[0][j-1] \quad (i=0, j>0)$$

$$P[i][0]=P[i-1][0]+v[i-1][0] \quad (i>0, j=0)$$

- 画出用动态规划思想求出的各个路口对 P 点的最小距离。图中圆圈里就是这个距离。箭头表示所寻得的最佳行走路径。（图 3）

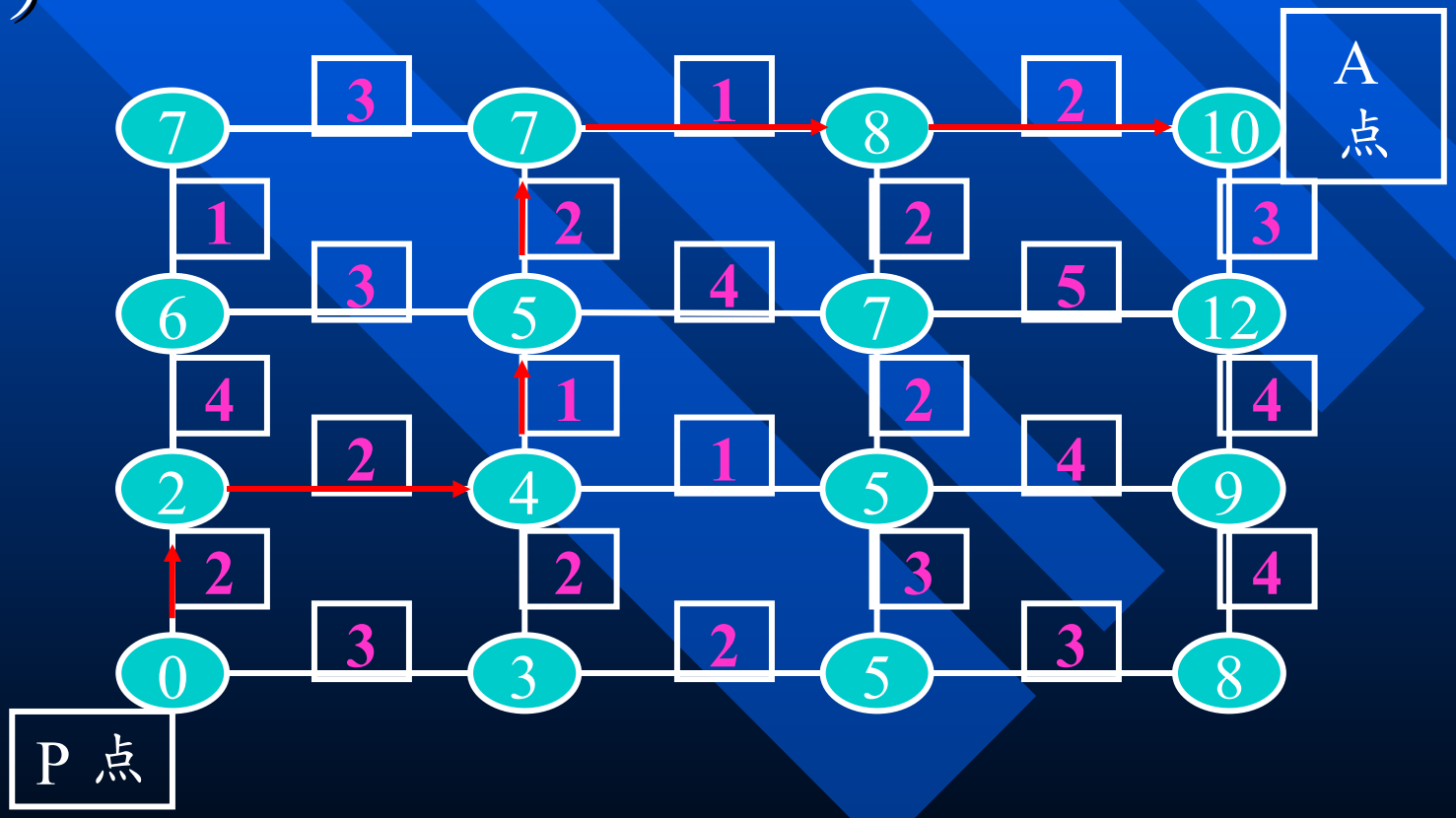


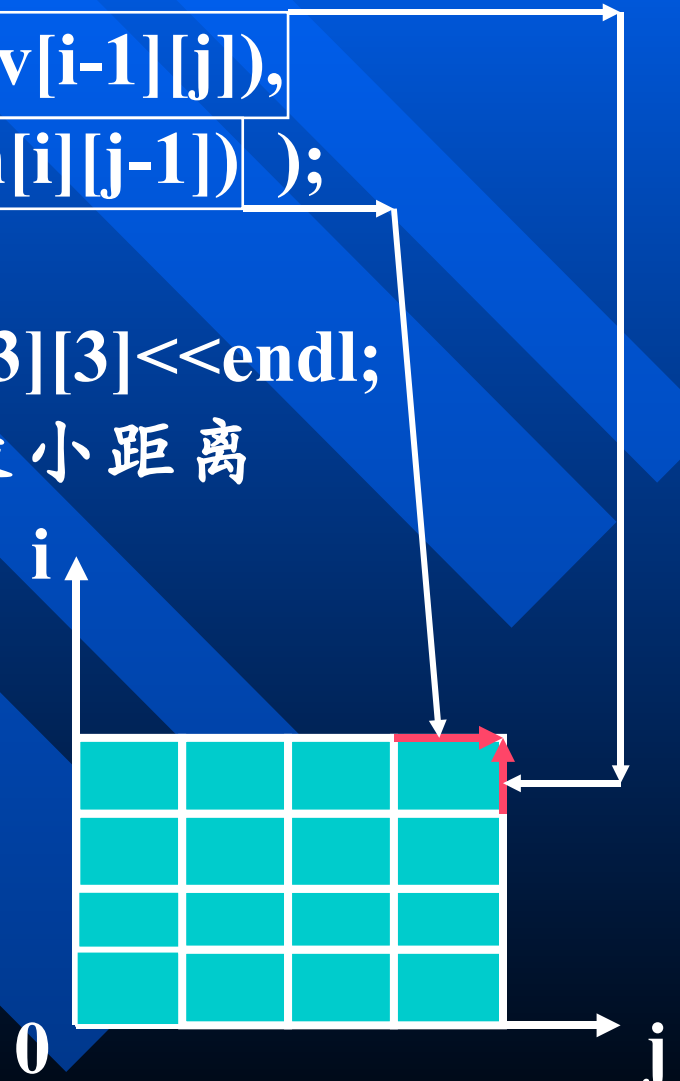
图 3

参考程序如下

```
#include <iostream.h>
int min(int,int);
void main(void)                    //  主函数
 { int i,j;
   int h[4][3]={ {3,2,3},{2,1,4},{3,4,5},{3,1,2} };
   int v[3][4]={ {2,2,3,4},{4,1,2,4},{1,2,2,3} };
   int p[4][4]={ {0,0,0,0},{0,0,0,0},{0,0,0,0},{0,0,0,0} };
   p[0][0]=0;
   for(j=1;j<4;j++)                          //y 轴上的点
      p[0][j]=p[0][j-1]+h[0][j-1];
   for(i=1;i<4;i++)                          //x 轴上的点
      p[i][0]=p[i-1][0]+v[i-1][0];
```

```
for(i=1;i<4;i++)                    // 内部的点
    for(j=1;j<4;j++)
        p[i][j]=min( ( p[i-1][j]+v[i-1][j]),
                     (p[i][j-1]+h[i][j-1]) );

cout<<"from P to A is "<<p[3][3]<<endl;
// 输出每个路口对 P 点的最小距离
for(i=3;i>=0;i--)
{
   for(j=0;j<=3;j++)
   {  cout<<p[i][j]<<"  ";}
    cout<<endl;
 }
}
```



```
int min(int a,int b)
{
    if (a<=b )  return a;
    else  return b;
}
```

动态规划的几个概念：

阶段：据空间顺序或时间顺序对问题的求解划分阶段。

状态：描述事物的性质，不同事物有不同的性质，因而用不同的状态来刻画。对问题的求解状态的描述是分阶段的。

决策：根据题意要求，对每个阶段所做出的某种选择性操作。

状态转移方程：用数学公式描述与阶段相关的状态间的演变规律。

动态规划是运筹学的一个重要分支，是解决多阶段决策过程最优化的一种方法。

所谓多阶段决策过程，是将所研究的过程划分为若干个相互联系的阶段，在求解时，对每一个阶段都要做出决策，前一个决策确定以后，常常会影响下一个阶段的决策。

动态规划所依据的是“最优性原理”。

“最优性原理”可陈述为：不论初始状态和第一步决策是什么，余下的决策相对于前一次决策所产生的新状态，构成一个最优决策序列。

最优决策序列的子序列，一定是局部最优决策子序列。

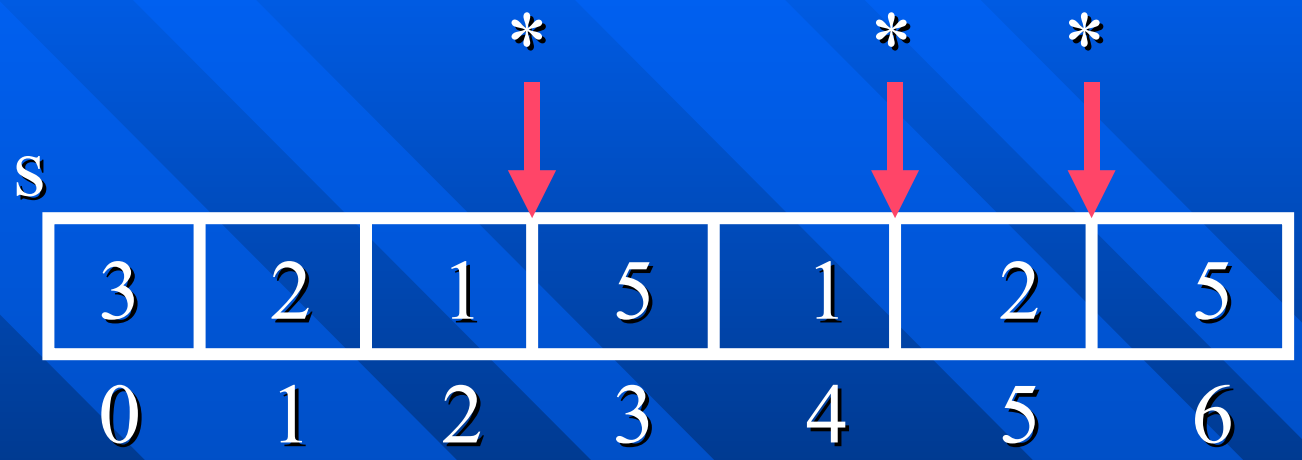
包含有非局部最优的决策子序列，一定不是最优决策序列。

动态规划的指导思想是：在做每一步决策时，列出各种可能的局部解，之后依据某种判定条件，舍弃那些肯定不能得到最优解的局部解。这样，在每一步都经过筛选，以每一步都是最优的来保证全局是最优的。筛选相当于最大限度地有效剪枝（从搜索角度看），效率会十分高。但它又不同于贪心法。贪心法只能做到局部最优，不能保证全局最优，因为有些问题不符合最优性原理。

两种算法的差别在于，贪心法产生一个按贪心策略形成的判定序列，该序列不保证解是全局最优的。而动态规划会产生许多判定序列，再按最优性原理对这些序列加以筛选，去除那些非局部最优的子序列。

举例说明动态规划思路

问题： 在数字串中插入若干乘号使
总的乘积最大



请插入 3 个乘号使乘积最大

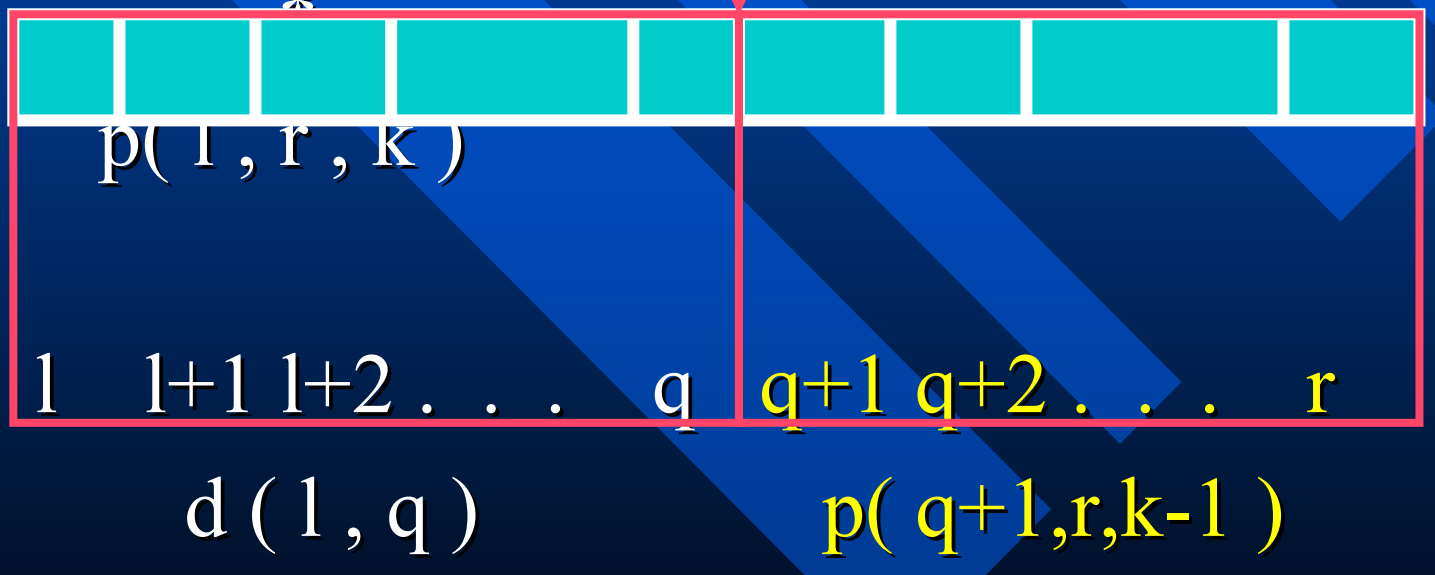
32*15*12*5=28800

3*215*12*5=38700

321*51*2*5=163710

解题思路

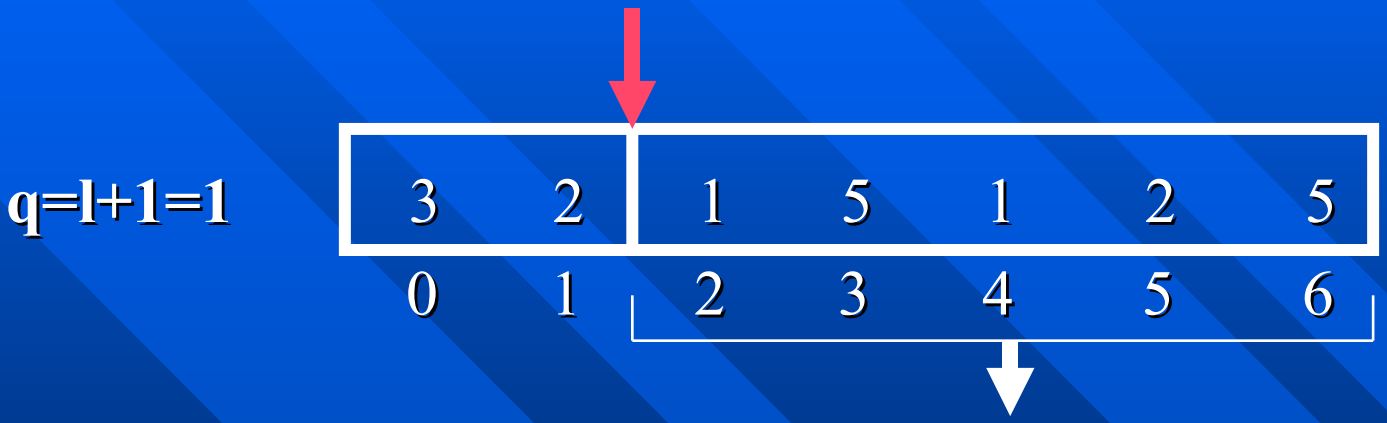
定义：从 1 到 r 加入 k 个乘号的最大乘积值



$$p(l,r,k) = \max\{ d(l,q) * p(q+1, r, k-1) \}$$

$d(l,q) = d(0,0) = 3$ $\quad p(q+1,r,k-1) = p(1,6,2)$

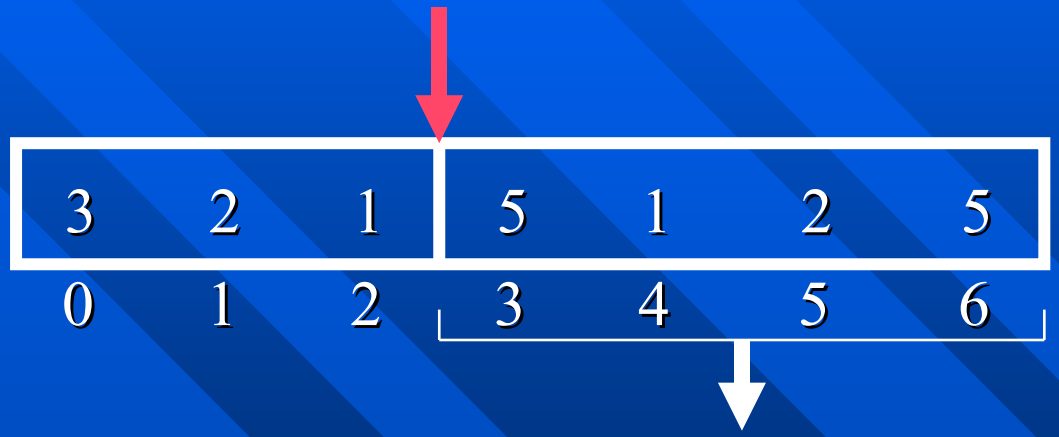
$(p(0,6,3)|q=0) = 3 * p(1,6,2)$



d(l,q)=d(0,1)=32 p(q+1,r,k-1)=p(2,6,2)

(p(0,6,3)|q=1) = 32 * p(2,6,2)

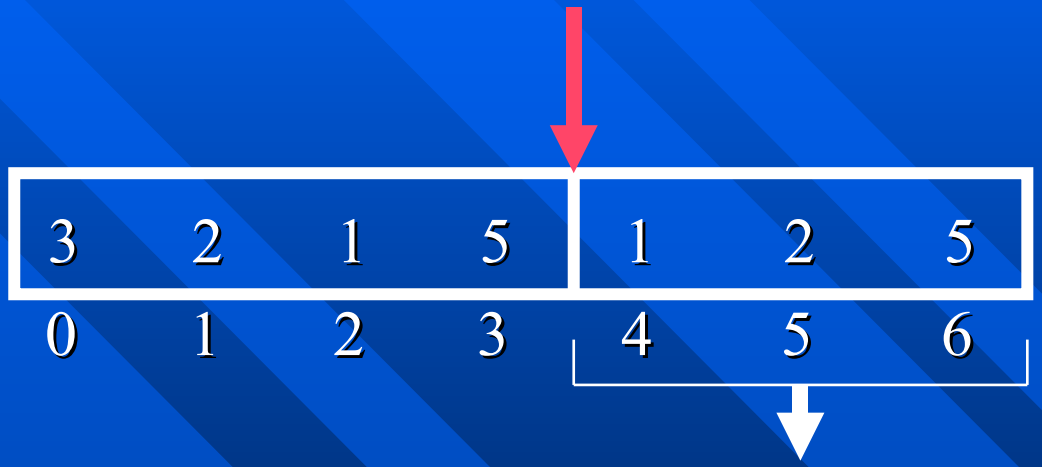
q=l+2=2



d(l,q)=d(0,2)=321 p(q+1,r,k-1)=p(3,6,2)

(p(0,6,3)|q=2) = 321 * p(3,6,2)

q=l+3=3



d(l,q)=d(0,3)=3215 p(q+1,r,k-1)=p(4,6,2)

(p(0,6,3)|q=3) = 3215 * p(4,6,2)

$$p(0,6,3) = \max\{\ 3 * p(1,6,2)\ ,\quad //\ q=0$$

$$32 * p(2,6,2)\ ,\quad //\ q=1$$

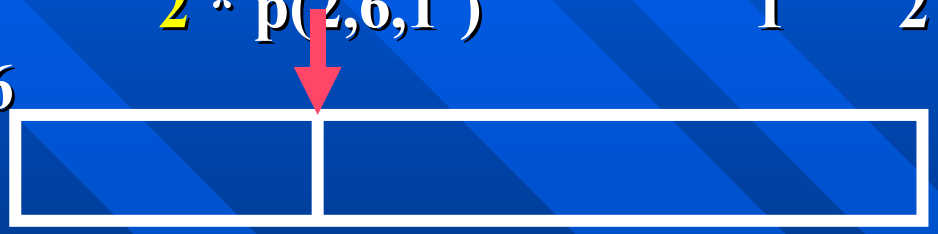
$$321 * p(3,6,2)\ ,\quad //\ q=2$$

$$3215 * p(4,6,2)\ \}\quad //\ q=3$$

p(1,6,2)

2 * p(2,6,1)

1 2 3 4 5 6



21 * p(3,6,1)

1 2 3 4 5 6

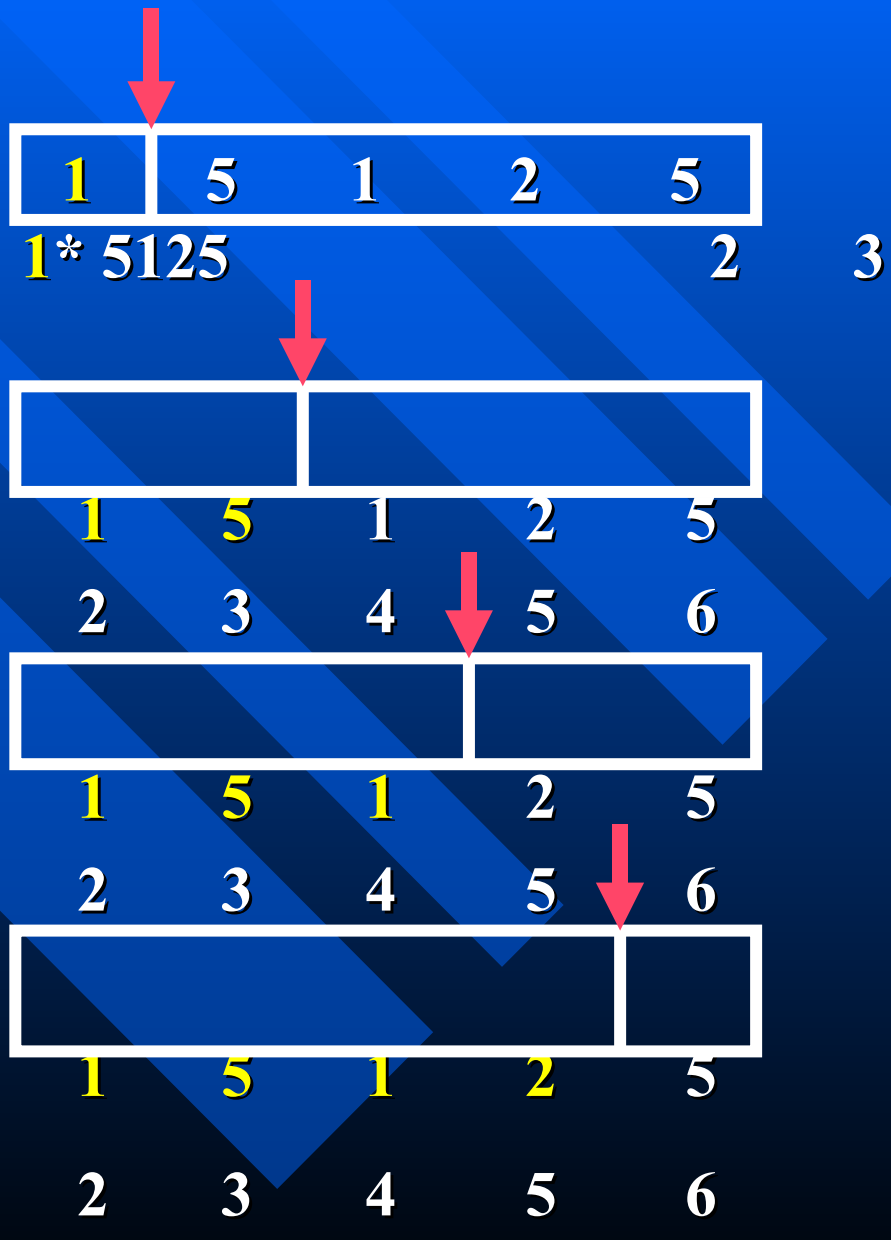
215 * p(4,6,1)

1 2 3 4 5 6

2151 * p(5,6,1)

1 2 3 4 5 6

p(2,6,1)


1 5 1 2 5
1* 5125
2 3
4 5 6
1 5 1 2 5
15 * 125
2 3 4 5 6
1 5 1 2 5
15 1* 25
2 3 4 5 6
1 5 1 2 5
1512 *5
2 3 4 5 6

$$p(2,6,1) = \max \{$$

$$1 * 5125 ,$$

$$15 * 125 ,$$

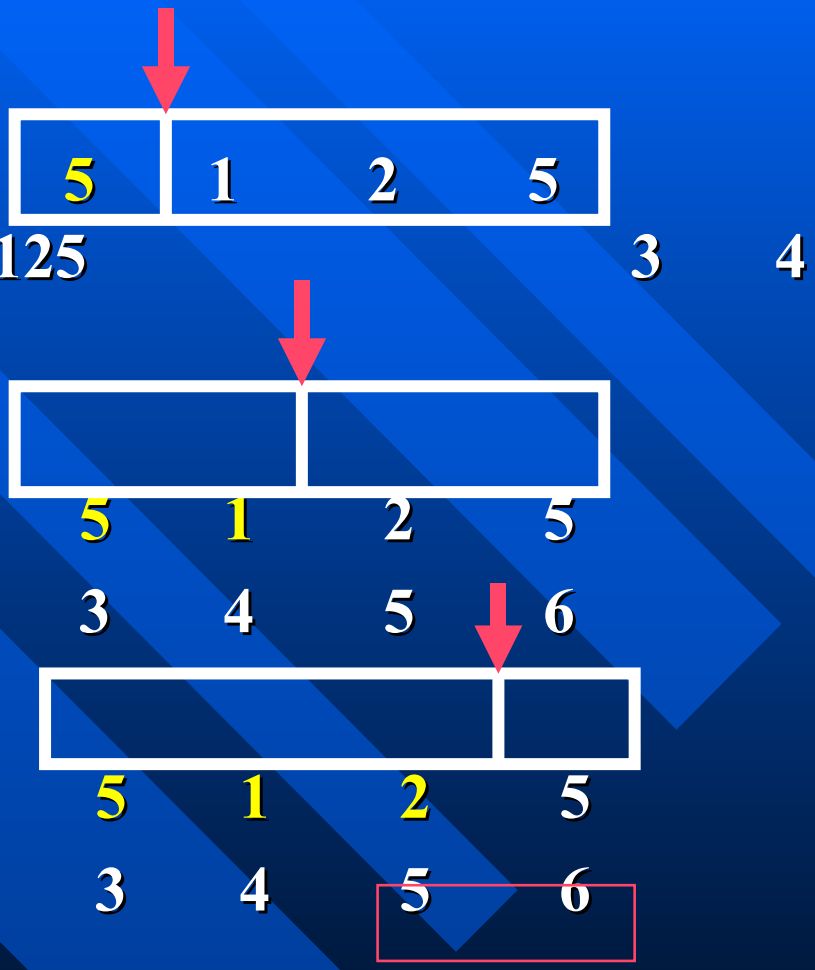
$$151 * 25 ,$$

$$1512 * 5$$

$$\}$$

$$= 7560$$

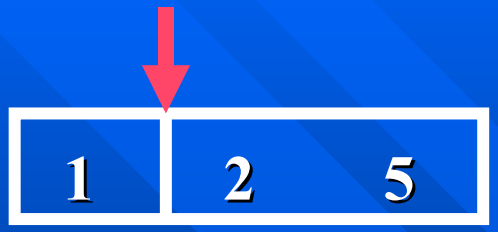
p(3,6,1)



p(3,6,1)=max{5*125, 51*25, 512*5 }

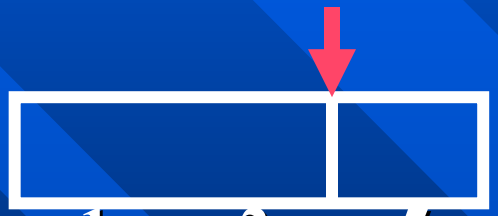
= 2560

p(4,6,1)



1 * 25

4 5 6



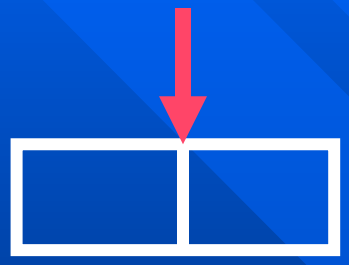
12 * 5

4 5 6

p(4,6,1) = max { 1 * 25, 12 * 5 }

= 60

p(5,6,1)



2 5

2 * 5

5 6

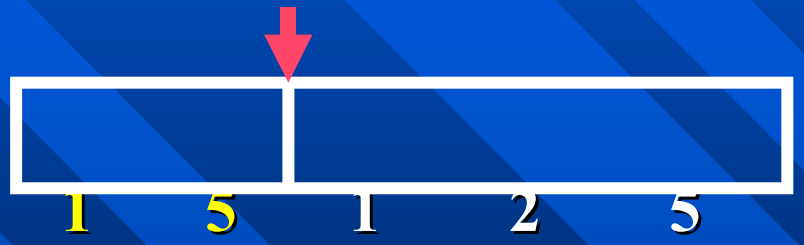
p(5,6,1) = 10

p(2,6,2)

1* p(3,6,1)

2 3

4 5 6



15 * p(4,6,1)

2 3 4 5 6

151 * p(5,6,1)

2 3 4 5 6

p(2,6,2)= { 1 * 2560, 15 * 60, 151 * 10 }

= 2560

p(3,6,2)

5 * p(4,6,1)

3 4 5

6

5 1 2 5

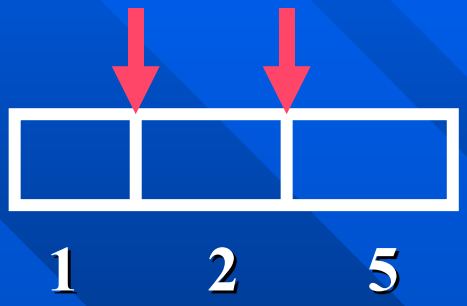
51 * p(5,6,1)

3 4 5 6

p(3,6,2) = { 5 * 60 , 51 * 10 }

= 510

p(4,6,2)

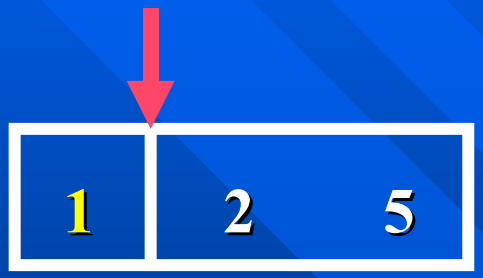


1 * 2 * 5

4 5 6

p(4,6,2) = 10

p(4,6,2)



1* p(5,6,1)

4 5 6

p(5,6,1) = 2 * 5 = 10

p(4,6,2) = 1 * p(5,6,1) = 10

p(0,6,3)= max{ 3 * p(1,6,2) , // q=0

32 * p(2,6,2) , // q=1

321* p(3,6,2) , // q=2

3215 * p(4,6,2) } // q=3

p(1,6,2) = 53760

p(2,6,2) = 2560

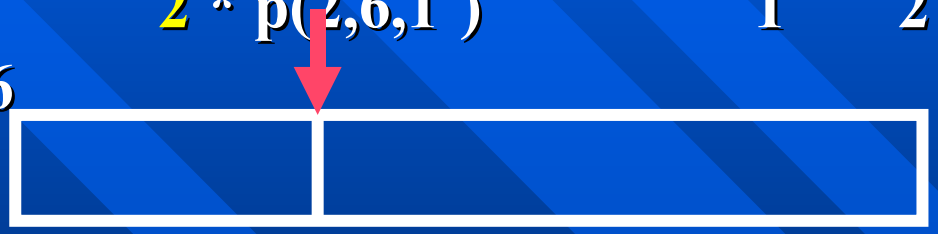
p(3,6,2) = 510

p(4,6,2) = 60

p(1,6,2)

2 * p(2,6,1)

1 2 3 4 5 6



21 * p(3,6,1)

1 2 3 4 5 6

215 * p(4,6,1)

1 2 3 4 5 6

2151 * p(5,6,1)

1 2 3 4 5 6

$$p(1,6,2) = \max\{\ 2 * p(2,6,1),\ 21 * p(3,6,1),\ 215 * p(4,6,1),\ 2151 * p(5,6,1)\ \}$$

$$= \max\{2*7560, 21*2560, 215*60, 2151*10\}$$

$$= 53760$$

p(0,6,3)= max{ 3 * p(1,6,2) , // q=0

32 * p(2,6,2) , // q=1

321* p(3,6,2) , // q=2

3215 * p(4,6,2) } // q=3

p(1,6,2) = 53760

p(2,6,2) = 2560

p(3,6,2) = 510

p(4,6,2) = 60

p(0,6,3)= max{ 3 * 53760 ,

32 * 2560 ,

321* 510,

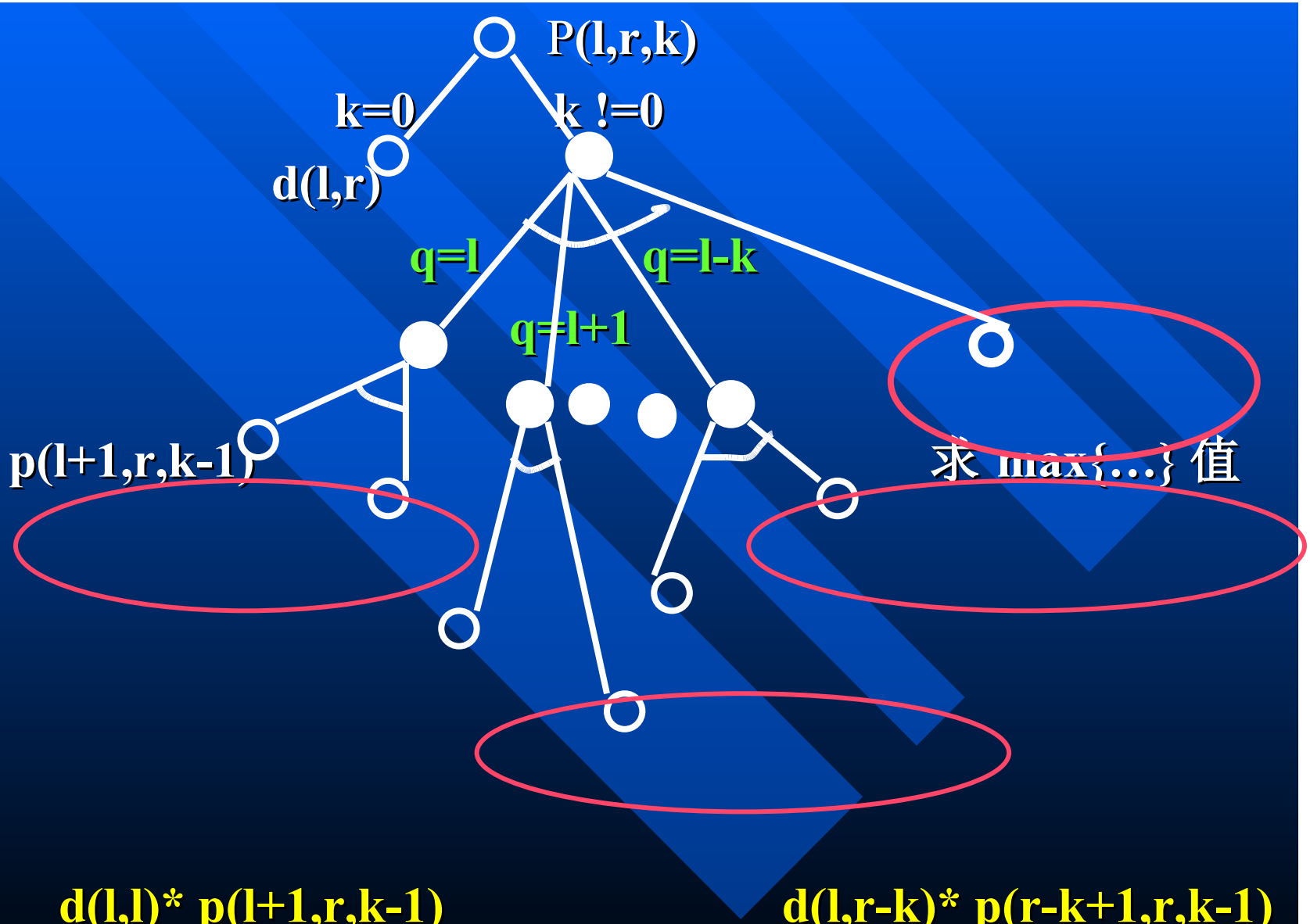
3215 * 60 } = 163710

p(1,6,2) = 53760

p(2,6,2) = 2560

p(3,6,2) = 510

p(4,6,2) = 60


P(l,r,k)
k=0
k !=0
d(l,r)
q=l
q=l-k
q=l+1
p(l+1,r,k-1)
求 max{…} 值
d(l,l)* p(l+1,r,k-1)
d(l,r-k)* p(r-k+1,r,k-1)
p(r-k+1,r,k-1)

d[i][j] j	0	1	2	3	4	5	6
i							
0	3	32	321	3215	32151	321512	3215125
1		2	21	215	2151	21512	215125
2			1	15	151	1512	15125
3				5	51	512	5125
4					1	12	125
5						2	25

怎样计算这张表 ？

d[i][6], i=0,1,2,3,4,5,6.

d[0][6]=s = 3215125

d[1][6]= 215125

= 3215125 % 1000000

= s % 1000000 s1=1000000

= s % s1

s1= s1/10

d[2][6]= d[1][6] % s1

```
s1=1000000;  d[0][6]=s;
for( i=1; i<= 6; i++ )
  {
     d[i][6] = d[i-1][6] % s1;
     s1 = s1/10;
  }
```

怎样求　d[i][5],d[i][4],…d[i][0]　?

i=0,1,2,3,4,5,6

```
for( j=5; j>=0; j-- )
    for( i=0; i<= j;  i ++ )
        {
            d[i][j]=d[i][j+1]/10;
        }
```

参　考　程　序

```
#include<iostream>  // 预编译命令

#include<cstring>    // 预编译命令

using namespace std; // 使用名字空间


 const int S=3215125; // 定义常整数
 int d[7][7];                  // 定义二维数组
```

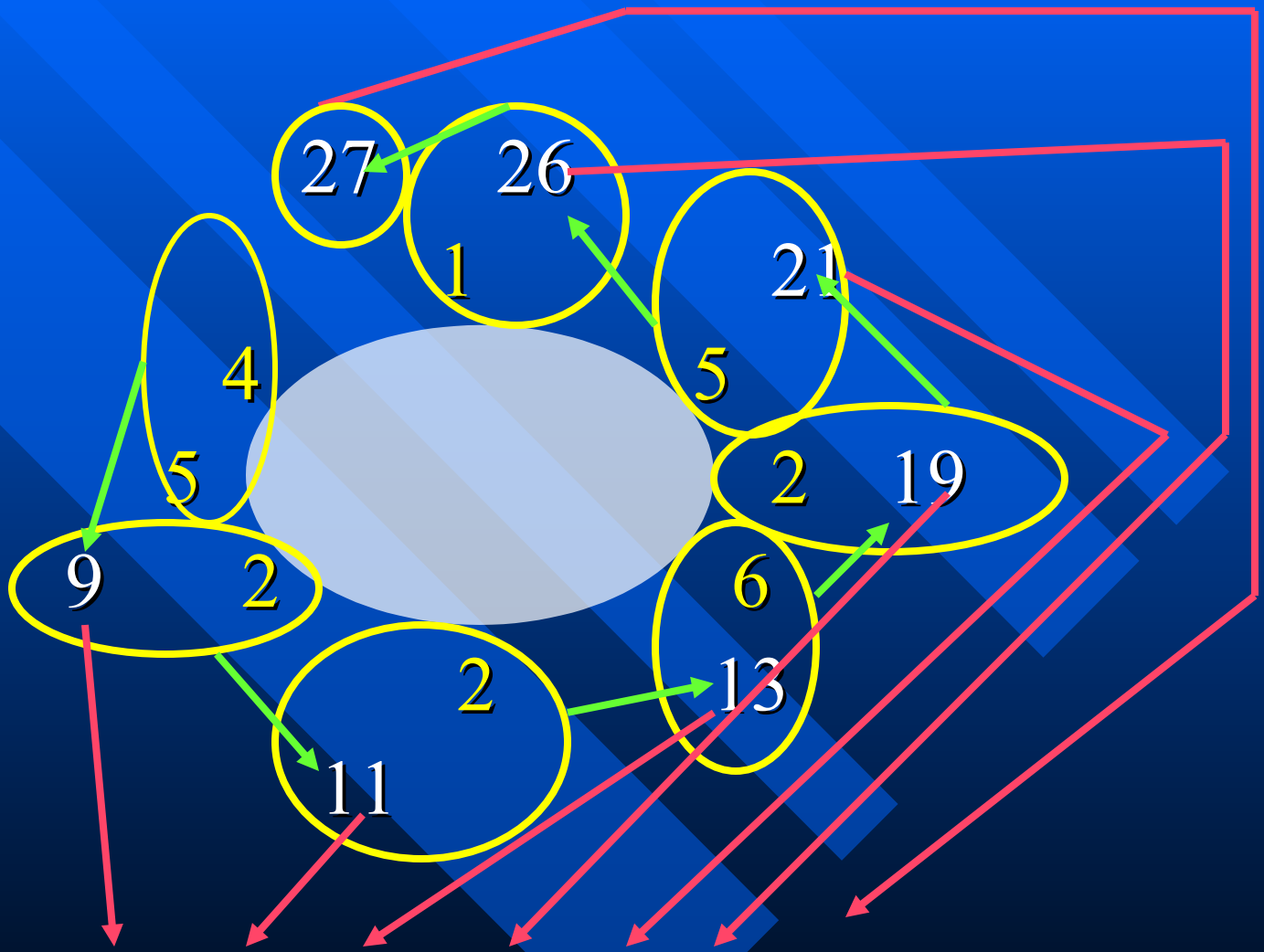
```
int P( int l, int r, int k ) // 计算 P 函数值

{
   if ( k==0) return d[l][r];
   int x, ans=0;
   for( int q=1; q<=r-k; q++ )
      {   x=d[l][q]*P( q+1,r,k-1);
          if( x>ans ) ans=x;   }
   return ans;
}
```

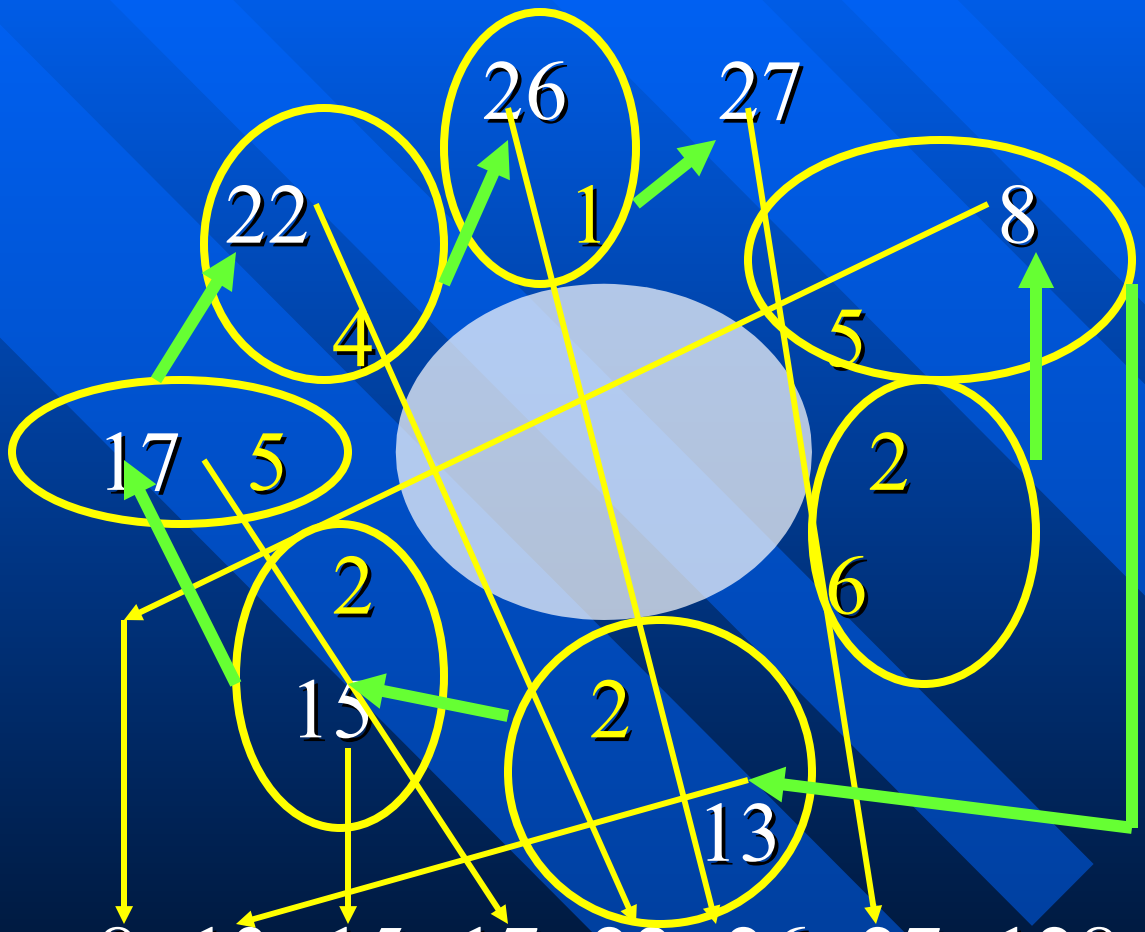
```
int main()
{
    memset( d,0,sizeof(d));
    int s1,I,j;
    s1=1000000;  d[0][6]=s;
    for( i=1; i<= 6; i++ )
    {
        d[i][6] = d[i-1][6] % s1;
        s1 = s1/10;
    }
```

```
    for( j=5; j>=0; j-- )
        for( i=0; i<= j;  i ++ )
            {
                d[i][j]=d[i][j+1]/10;
            }
    cout<<P( 0,6,3 )<<endl;
    return 0;
}
```

石子合并

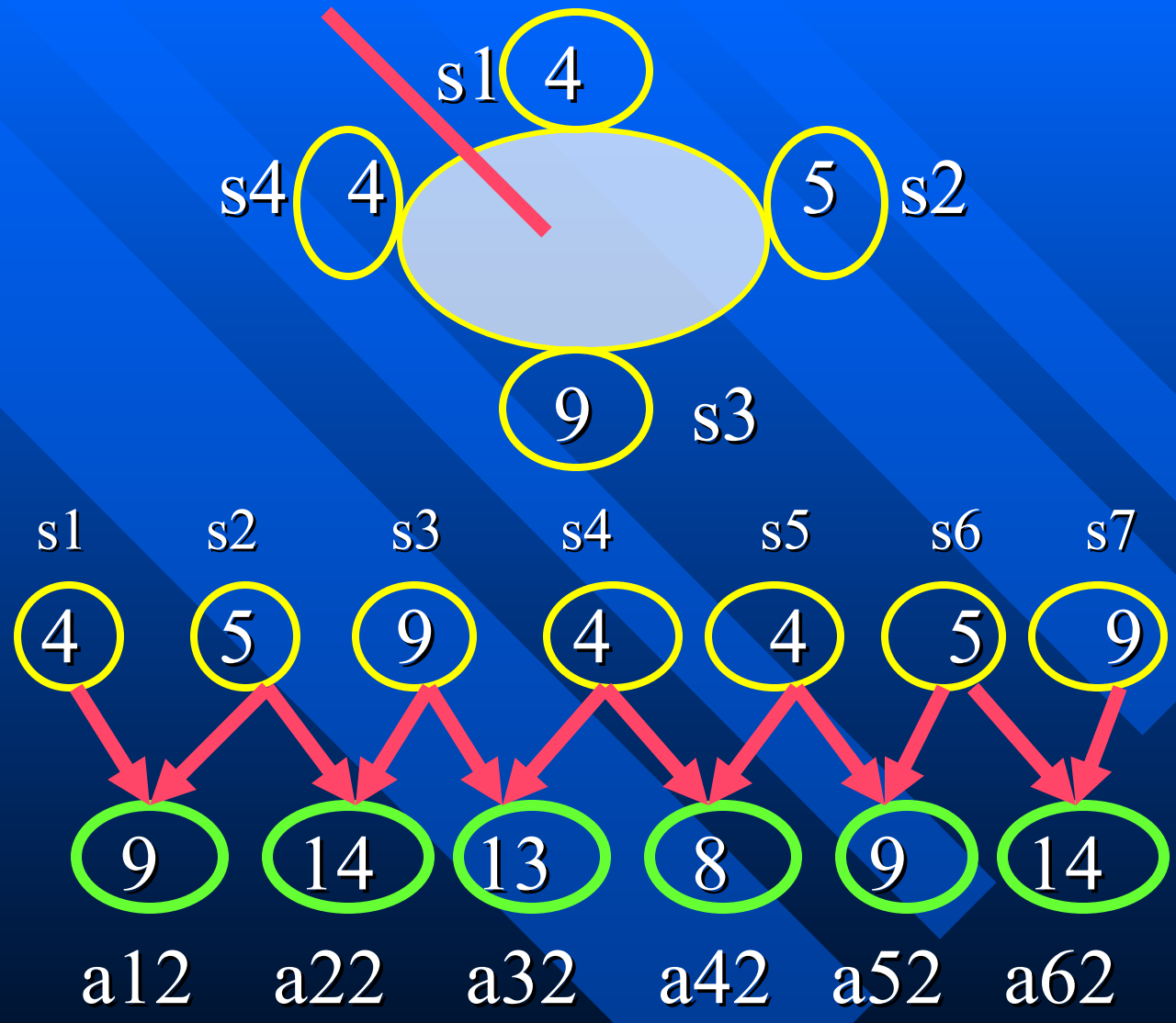


9+11+13+19+21+26+27=126



8+13+15+17+22+26+27=128

此例说明贪心法
得不出最优解
试用动态规划解
题


s1
4
s4
4
5
s2
9
s3
s1
s2
s3
s4
s5
s6
s7
4
5
9
4
4
5
9
9
14
13
8
9
14
a12
a22
a32
a42
a52
a62

$$a_{ij} = \sum_{m} Sm, \quad j=1,2,3,4, \quad i=1,2,\ldots,8-j$$

$$m=i,i+1,\ldots,i+j-1$$

i \ j	1	2	3	4
1	4	9	18	22
2	5	14	18	22
3	9	13	17	22
4	4	8	13	22
5	4	9	18	
6	5	14		
7	9			

a [i][j]

定义

d_{ij}

表示从第 i 堆开始合并 j 堆的得分

$$d_{ij} = \max \{ d_{i,k} + d_{i+k,j-k} \} + a_{ij}$$

$$1 <= k <= j-1 , \quad j = 2, 3, 4$$

$$d_{i1} = 0 , \quad i = 1,2,\ldots,7$$

$$d_{i,2} = \max_{k=1} \{ d_{i,1} + d_{i+1,1} \} + a_{i2}$$

$d_{i,1}$: $\uparrow_0$ $d_{i+1,1}$: $\uparrow_0$

s1	s2	s3	s4	s5	s6	s7
4	5	9	4	4	5	9

a12	a22	a32	a42	a52	a62
9	14	13	8	9	14

d12	d22	d32	d42	d52	d62
9	14	13	8	9	14

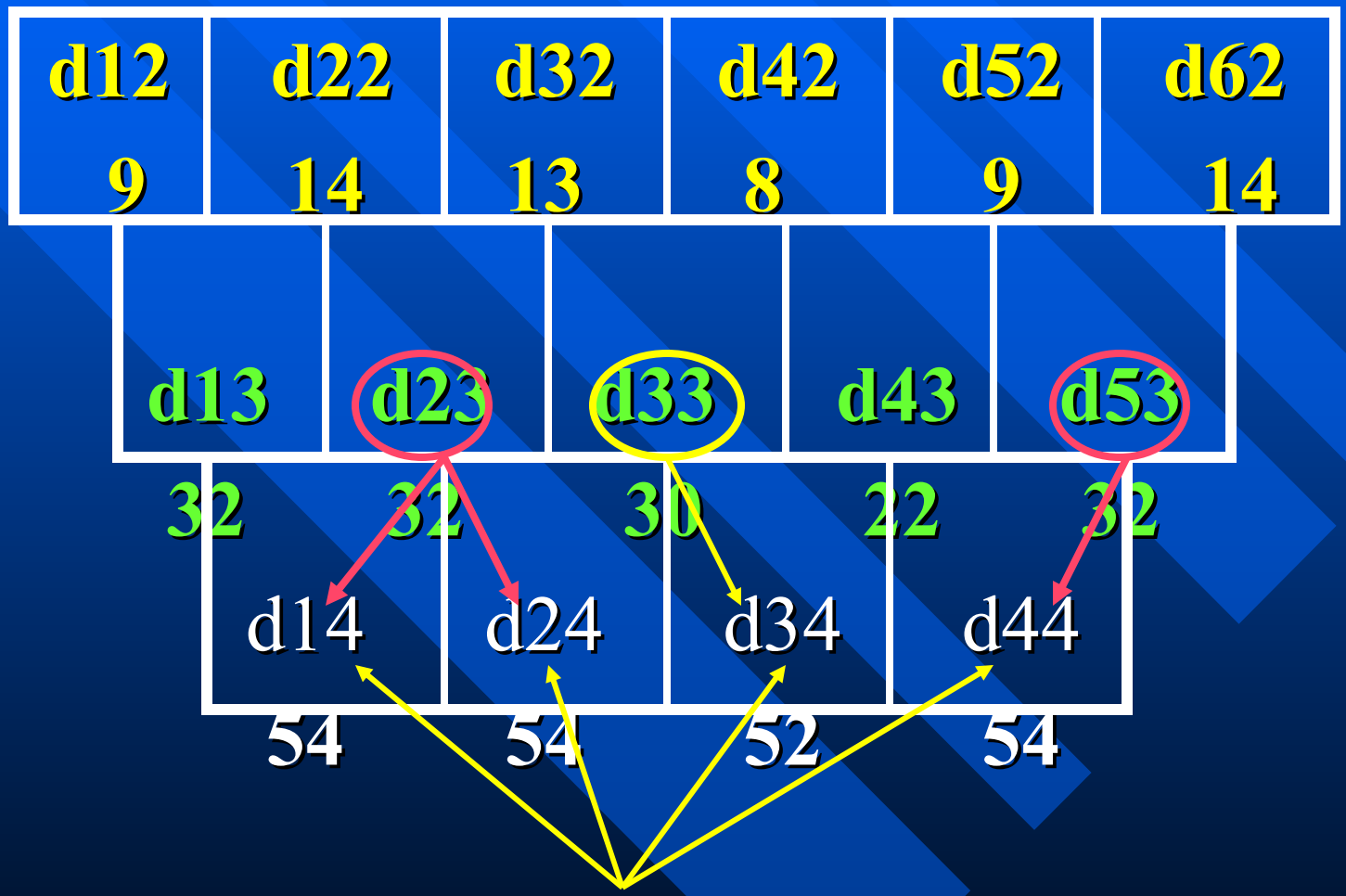
$$d_{i,3} = \max_{k=1,2} \{ d_{i,k} + d_{i+k,3-k} \} + a_{i3}$$

$$= \max \{ d_{i,2}, d_{i+1,2} \} + a_{i3}$$

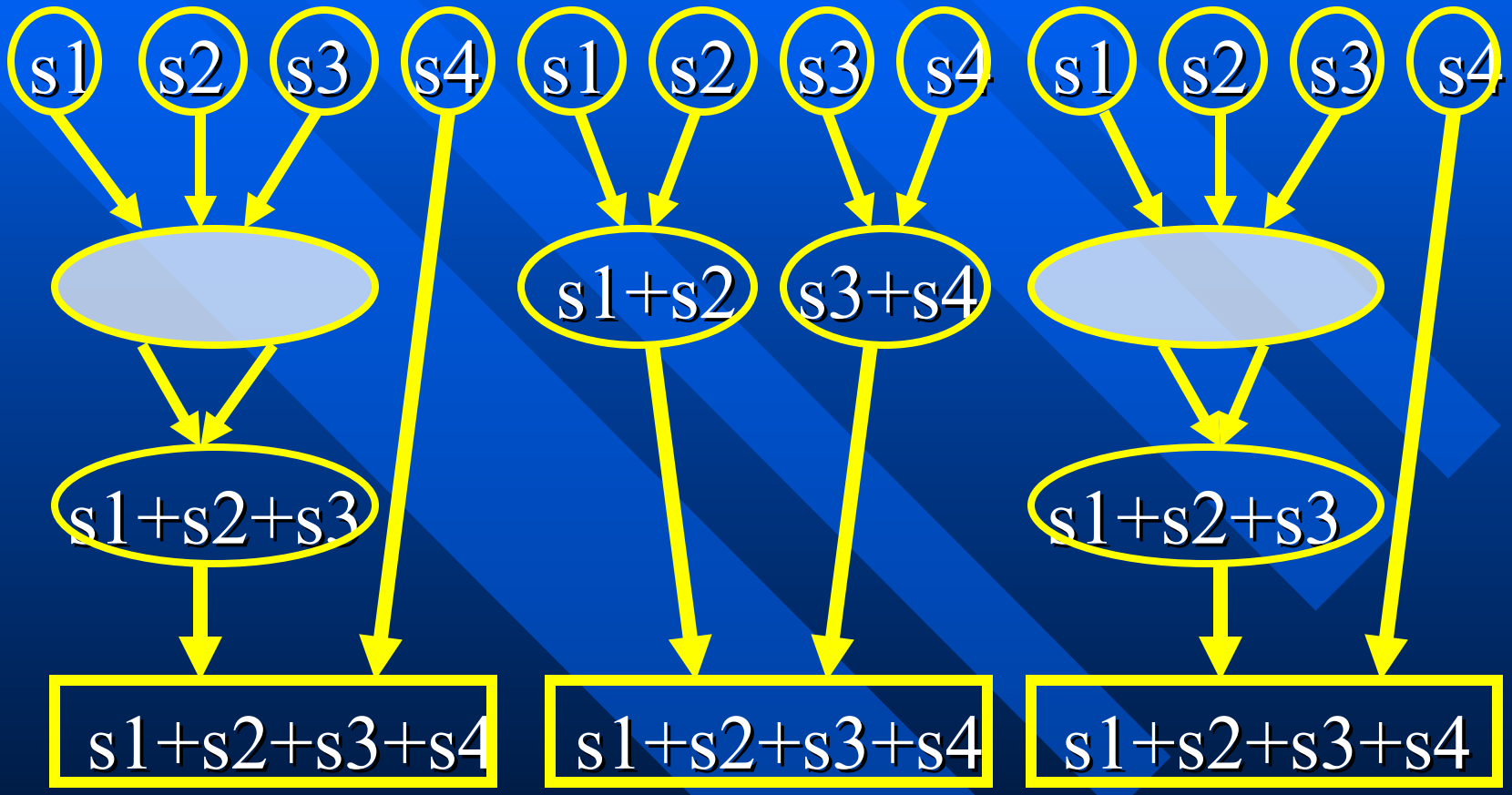
$$d_{i,4} = \max_{k=1,2,3} \{ d_{i,k} + d_{i+k,\,4-k} \} + a_{i4}$$

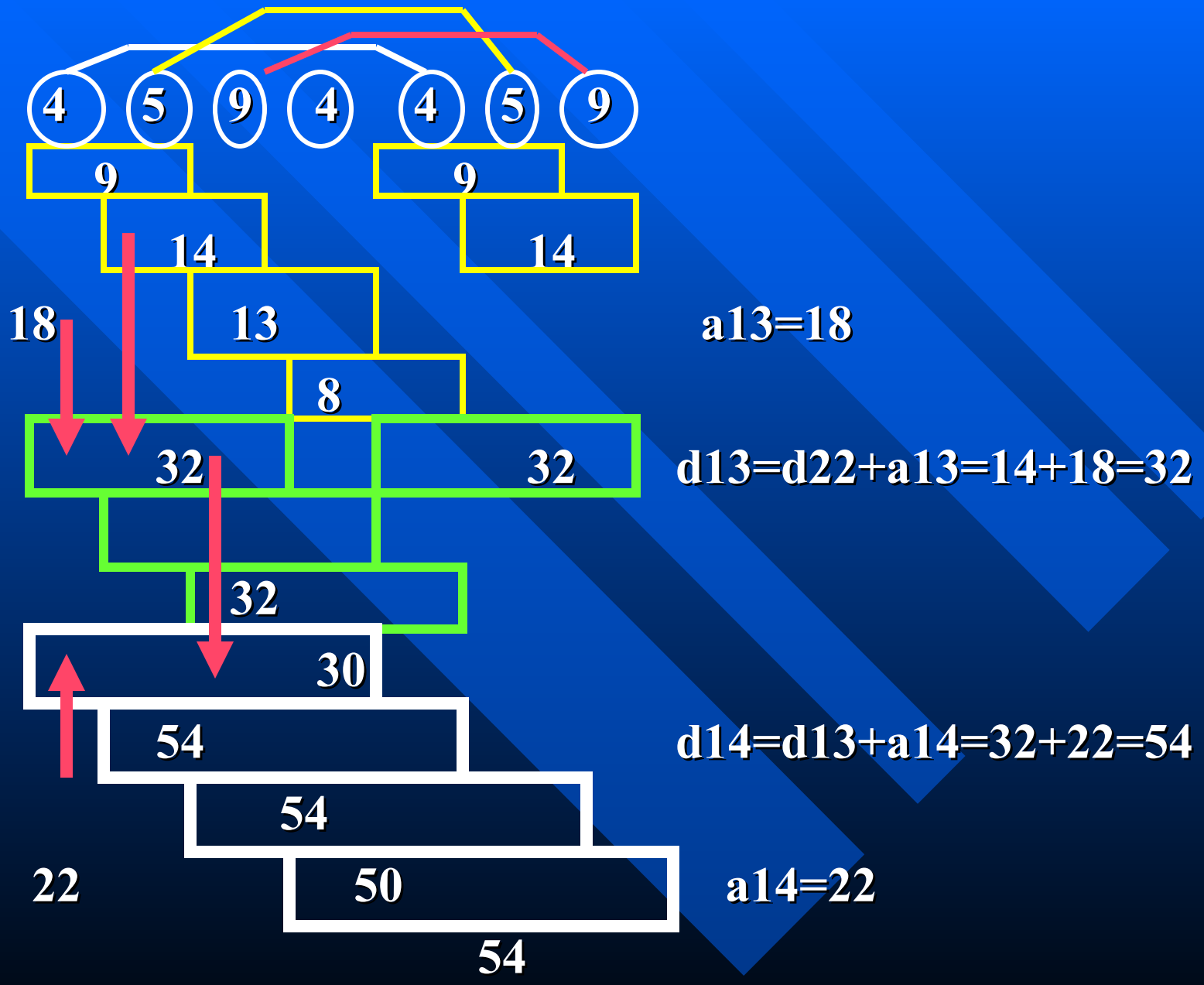
$$= \max \{ d_{i,1} + d_{i+1,3} ;\; d_{i,2} + d_{i+1,2};\; d_{i,3} + d_{i+1,1} \} + a_{i4}$$

$$= \max \{ d_{i+1,3} ; d_{i,2} + d_{i+1,2}; d_{i,3} \} + a_{i4}$$


d12
9
d22
14
d32
13
d42
8
d52
9
d62
14
d13
32
d23
32
d33
30
d43
22
d53
32
d14
54
d24
54
d34
52
d44
54

a i4 = 22


s1
s2
s3
s4
s1
s2
s3
s4
s1
s2
s3
s4
s1+s2
s3+s4
s1+s2+s3
s1+s2+s3
s1+s2+s3+s4
s1+s2+s3+s4
s1+s2+s3+s4


4
5
9
4
4
5
9
9
9
14
14
18
13
a13=18
8
32
32
d13=d22+a13=14+18=32
32
30
54
d14=d13+a14=32+22=54
54
22
50
a14=22
54

hanoi 塔问题的动态规划解法

令　m ———柱子数

n ———圆盘数

hanoi(m,n) 表示具有 m 根柱子 n 个圆盘的搬移

设　起始柱为　from

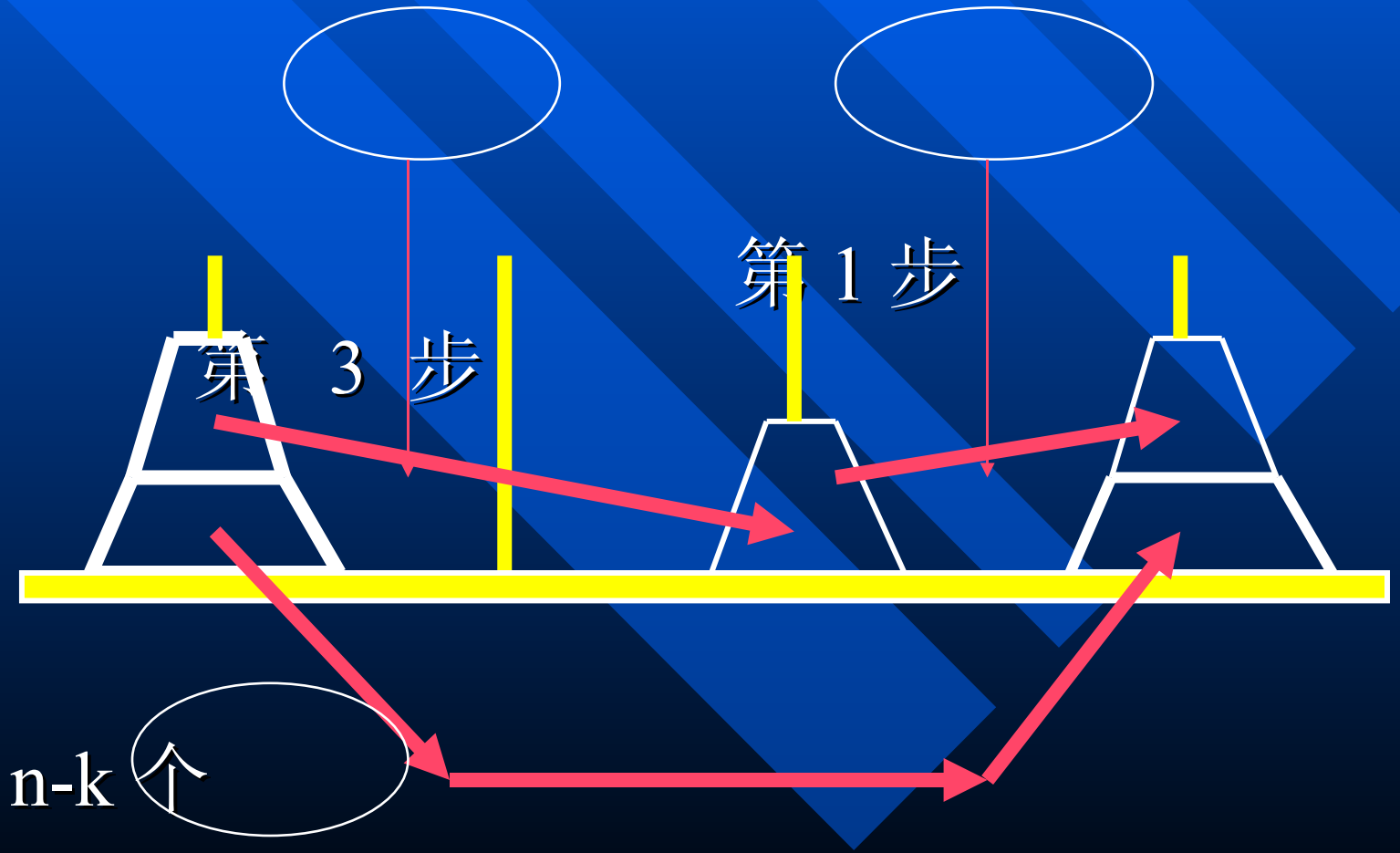
思路

将 from 上的圆盘分成上下两部分， 下面的盘数为 k, 上面的为 n-k 。

先用 hanoi(m,n-k) 将 from 上的 n-k 个圆盘

借助于其他圆盘搬至 temp[m-2] (也可以用

别的); 然后再用 hanoi(m-1,k) 将下面 k 个从 from 移至 to; 之后再将 temp[m-2] 上的 n-k 个圆盘 , 借助于其他圆盘 , 用 hanoi(m,n-k) 搬至 to


第 1 步
第 3 步
n-k 个

搬移过程　　广义数学式

hanoi(m,n) = hanoi(m,n-k) +
hanoi(m-1,k) +
hanoi(m,n-k)

抽象为　　h=a+b+c

令　s(m,n)---hanoi(m,n) 的最少移动步数

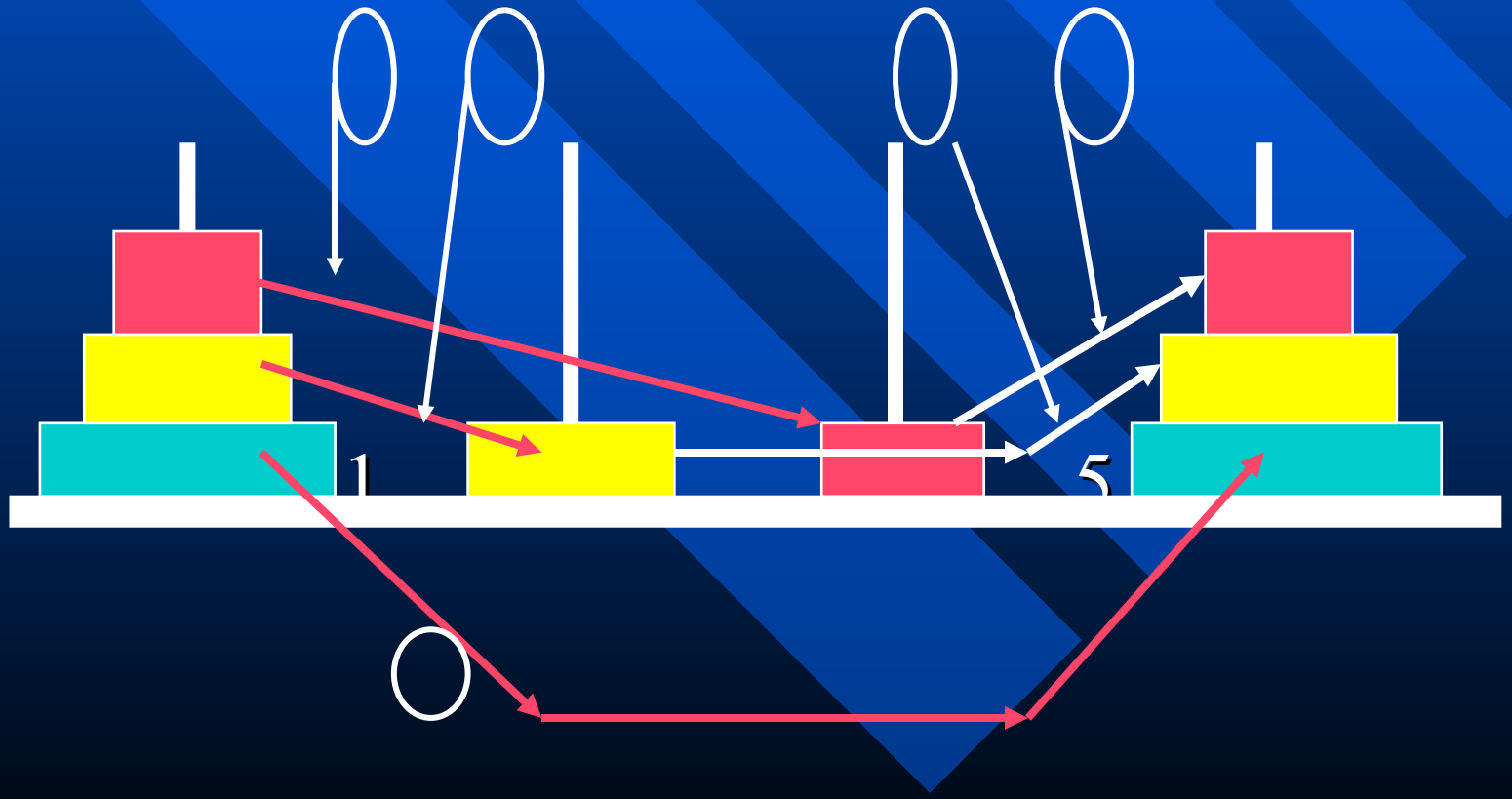
s(m,n) =min{ s(m,n-k)+s(m-1,k)+s(m,n-k)}

k

k=1,2,…,n　　m>3

$$s(m,n)=\min_{k}\{2*s(m,n-k)+s(m-1,k)\}$$

1. $s(m,n)=1$, $m=2, n=1$
2. $s(m,n)=1$, $m=3, n=1$
3. $s(m,n)=3$, $m=3, n=2$
4. $s(m,n)=3$, $m=4, n=2$
5. $s(m,n)=7$, $m=3, n=3$
6. $s(m,n)=5$, $m=4, n=3$


1
5

分析 ： k 值的选择是关键

1. 问题的解决有若干个阶段，是一个多步决策的过程。
2. 对于阶段 b 而言，阶段 a 的解决与之无关，相关的只是阶段 a 解决后的状态。问题的阶段划分满足无后效性的要求。
3. 问题的最优策略是各阶段最优子策略的组合，若问题 h 取得最优解，则其在阶段 a, b, c 上也必然取得最优解。问题满足动态规划的最优性原理

动态规划一般式

$$s(m,n)=\begin{cases}\min\{\, s(m,n-k)+s(m-1,k)+s(m,n-k)\,\} & \\ \qquad k & \\ \qquad k=1,2,\ldots,n \quad m>3 \quad n>1 & \\ 1 & k=1 \quad m=3 \quad n>1 \\ & m=2 \quad n=1\end{cases}$$

m=3, n=2

k=1

$s(m,n)=\min\{ 2* s(m,n-k)+s(m-1,k) \}$

$s(3,2)=\min\{ 2*s(3,2-1)+s(3-1,1)\}$

$=\min\{ 2*s(3,1)+s(2,1)\}$

$=\min\{ 2*1 + 1\}$

$= 3$

m=3, n=3

k=1

$$s(m,n)=\min\{ 2* s(m,n-k)+s(m-1,k) \}$$

$$s(3,3)=\min\{ 2*s(3,3-1)+s(3-1,1)\}$$

$$=\min\{ 2*s(3,2)+s(2,1)\}$$

$$=\min\{ 2*3 + 1\}$$

$$= 7$$

m=3, n=4

k=1

$s(m,n)=\min\{ 2* s(m,n-k)+s(m-1,k) \}$

$s(3,4)=\min\{ 2*s(3,4-1)+s(3-1,1)\}$

$=\min\{ 2*s(3,3)+s(2,1)\}$

$=\min\{ 2*7 + 1\}$

$= 15$

m=3, n=5

k=1

$s(m,n)=\min\{ 2* s(m,n-k)+s(m-1,k) \}$

$s(3,5)=\min\{ 2*s(3,5-1)+s(3-1,1)\}$

$=\min\{ 2*s(3,4)+s(2,1)\}$

$=\min\{ 2*15 + 1\}$

$= 31$

m=3, n=6

k=1

s(m,n)=min{ 2* s(m,n-k)+s(m-1,k) }

s(3,6)=min{ 2*s(3,6-1)+s(3-1,1)}

=min{ 2*s(3,5)+s(2,1)}

=min{ 2*31 + 1}

= 63

m=4, n=3 , k=1, 2, 3

s(m,n)=min{ 2* s(m,n-k)+s(m-1,k) }

s(4,3)=min{ 2*s(4,3-1)+s(4-1,1),

2*s(4,3-2)+s(4-1,2),

2*s(4,3-3)+s(4-1,3) }

=min{ 2*s(4,2)+s(3,1),

2*s(4,1)+s(3,2),

2*s(4,0)+s(3,3) }

=min{ 2*3 + 1, 2*1+3, 2*0+7 }

= 5

m=4, n=4 , k=1, 2, 3, 4

s(m,n)=min{ 2* s(m,n-k)+s(m-1,k) }

s(4,4)=min{ 2*s(4,4-1)+s(4-1,1),
2*s(4,4-2)+s(4-1,2),
2*s(4,4-3)+s(4-1,3),
2*s(4,4-4)+s(4-1,4) }

=min{ 2*s(4,3)+s(3,1),
2*s(4,2)+s(3,2),
2*s(4,1)+s(3,3) ,
2*s(4,0)+s(3,4) }

=min{ 2*5 + 1, 2*3+3, 2*1+7, 2*0+ 15 }

= 9

$s(4,5)=13, \quad k=3$

$s(4,6)=17, \quad k=3$

$s(4,7)=25, \quad k=4$

$s(4,8)=33, \quad k=4$

$s(4,9)=41, \quad k=4$

思路清楚了，请自己编出程序，

并运行通过。

结　　束