区间dp

区间型模型

区间型动态规划是线性动态规划的拓展，它将区间长度作为阶段，长区间的答案与短区间有关。

在求解长区间答案前需先将短区间答案求出。

区间型动态规划的典型应用有石子合并。

T127589 石子合并（线性）ch5301

有N堆石子(N≤300排成一排。现要将石子合并成一堆.规定每次只能选相邻的两堆合并成一堆新的石子,并将新的一堆的石子数,记为该次合并的得分.

选择一种合并石子的方案,使得做N-1次合并,得分的总和最少。

输入数据:

第一行为石子堆数N;

第二行为每堆石子数。

输出数据 :

合并石子后得到的最小得分。

应该怎么合并呢?

8 3 6

3堆石子合并方案:

((8 +3)+6)

(8+(3+6))

Ans=Min(28,26)=26

8 5 5 8

8 5 5 8　8 5 5 8　8 5 5 8　8 5 5 8　8 5 5 8

N=4时，4堆一共合并了几次？

最后一次合并成一堆前的那两堆什么样？

8，18 或者 13,13 或者 18,8

哪种情况是理想的情况：

Min（8+18,13+13,+18+8）=26

子问题变成3堆和2堆的情况。

5堆石子:

(1,5)=min{

(1,1)+(2,5);

(1,2)+(3,5);

(1,3)+(4,5);

(1,4)+(5,5)}+sum[1,5]

n 堆石子：n-1次合并

$$a_1, a_2, a_3, \dots, a_{n-1}, a_n$$

- (1,n)=min{
- (1,1)+(2,n);
- (1,2)+(3,n)；

  …
- (1,n-1)+(n,n)}+sum[1,n]

- Min{(1..k)+(k+1..n)} +sum[1,n]

  枚举：k=1..n-1

动态规划算法：

定义f[i,j]表示从第i到第j堆间合并为一堆的最小代价。

状态转移方程：f[i,j]:={f[i,k]+f[k+1,j]}+sum[i,j]

枚举位置k：i<=k<=j-1

初始：f[i,i]=0;　　ans=f[1,n]

最好理解的方程之一

n=6

3 4 6 5 2 4

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

前缀和：

a[i]:记录第i堆石子数量。

s[i]=a[1]+a[2]+…+a[i]。//前缀和

a[i],……,a[j]　　共有j-i+1堆石子

sum[i,j]=a[i]+a[i+1]+…+a[j]

sum[i,j]=s[j]-s[i-1].

实现方法1：记忆化搜索

```
int dp(int i,int j){
    if(i==j)return f[i][j]=0;
    if(f[i][j]<INF)return f[i][j];
    //f[i][j]=INF;//可以预处理初始化为无穷大
    for(int k=i;k<j;k++)
        f[i][j]=min(f[i][j],dp(i,k)+dp(k+1,j));
    f[i][j]+=s[j]-s[i-1];
    return f[i][j];
}
```

观察方程的结构：f[i,j]:={f[i,k]+f[k+1,j]}+sum[i,j]

f[2,5]=min(f[2,2]+f[3,5];f[2,3]+f[4,5];f[2,4]+f[5,5])+sum[2,5]
=min(20,10+7,25)+17=34
f[i][j]的值由左下方区域的子问题来决策的。

目标

	1	2	3	4	5	6
1	3 0	7	20	36	47	61
2		4 0	10	25	34	48
3			6 0	11	20	34
4				5 0	7	17
5					2 0	6
6						4 0

实现方法2：递推：枚举区间长度

沿着对角线求：按i到j之间石子的数量从小到大合并
枚举len=i-j+1，长度区间[2..n]

	1	2	3	4	5	6
1	0	7	20	36	47	61
2		0	10	25	34	48
3			0	11	20	34
4				0	7	17
5					0	6
6						0

3 4 6 5 2 4

沿着对角线求：

a[i]..a[j]供j-i+1堆石子

外层循环变量len：从i开始的连续len堆石子:len=j-i+1

```
for(int len=2;len<=n;len++)
    for(int i=1;i+len-1<=n;i++){
        //根据j-i+1=len推出j以及i的上限
        int j=i+len-1;
        for(int k=i;k<j;k++)
            f[i][j]=min(f[i][j],f[i][k]+f[k+1][j])
        f[i][j]+=s[j]-s[i-1];
    }
cout<<f[1][n]<<endl;
```

时间:$O(n^3)$

方法3：倒序按行优先求

	1	2	3	4	5	6
1	3 0	7	20	36	47	61
2		4 0	10	25	34	48
3			6 0	11	20	34
4				5 0	7	17
5					2 0	6
6						4 0

```
for(int i=n-1;i>=1;i--)
    for(int j=i+1;j<=n;j++){
        for(int k=i;k<j;k++)
            f[i][j]=min(f[i][j],f[i][k]+f[k+1][j]);
        f[i][j]+=s[j]-s[i-1];
    }
cout<<f[1][n]<<endl;
```

方法4：正向列优先，每列倒着求

	1	2	3	4	5	6
1	3 0	7	20	36	47	61
2		4 0	10	25	34	48
3			6 0	11	20	34
4				5 0	7	17
5					2 0	6
6						4 0

```
for(int j=2;j<=n;j++)
    for(int i=j-1;i>=1;i--){
        for(int k=i;k<j;k++)
            f[i][j]=min(f[i][j],f[i][k]+f[k+1][j]);
        f[i][j]+=s[j]-s[i-1];
    }
cout<<f[1][n]<<endl;
```

总结本题：

1、前缀和的应用。

2、区间的dp的求解4种方法：

关键明确是以区间长度的大小划分阶段。

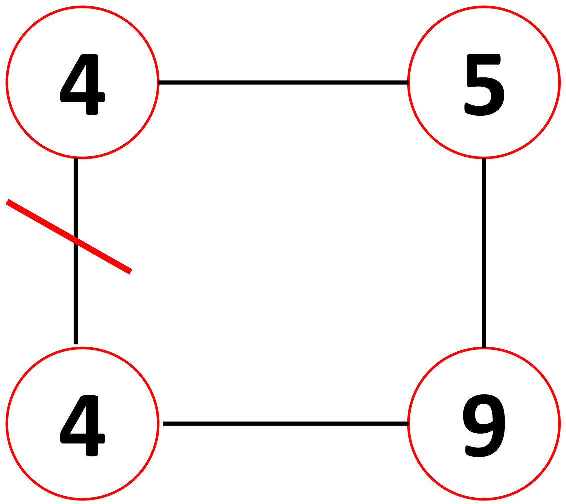
然后选择合适的求解顺序。

扩展：P1880 [NOI1995]石子合并

石子由一排改为围成一个环的形状？

4 5 9 4

4 5 9 4 4 5 9



4 5 9 4 4 5 9

拆开变为线性：n种拆法，变为长度2n-1。

[1, n], [2, n+1], …, [n, 2n-1]

环形石子合并算法:

环变成线性:长度: 2n-1

a[1],a[2]......a[n],a[n+1].....a[2n-1];

a[n+i]=a[i]

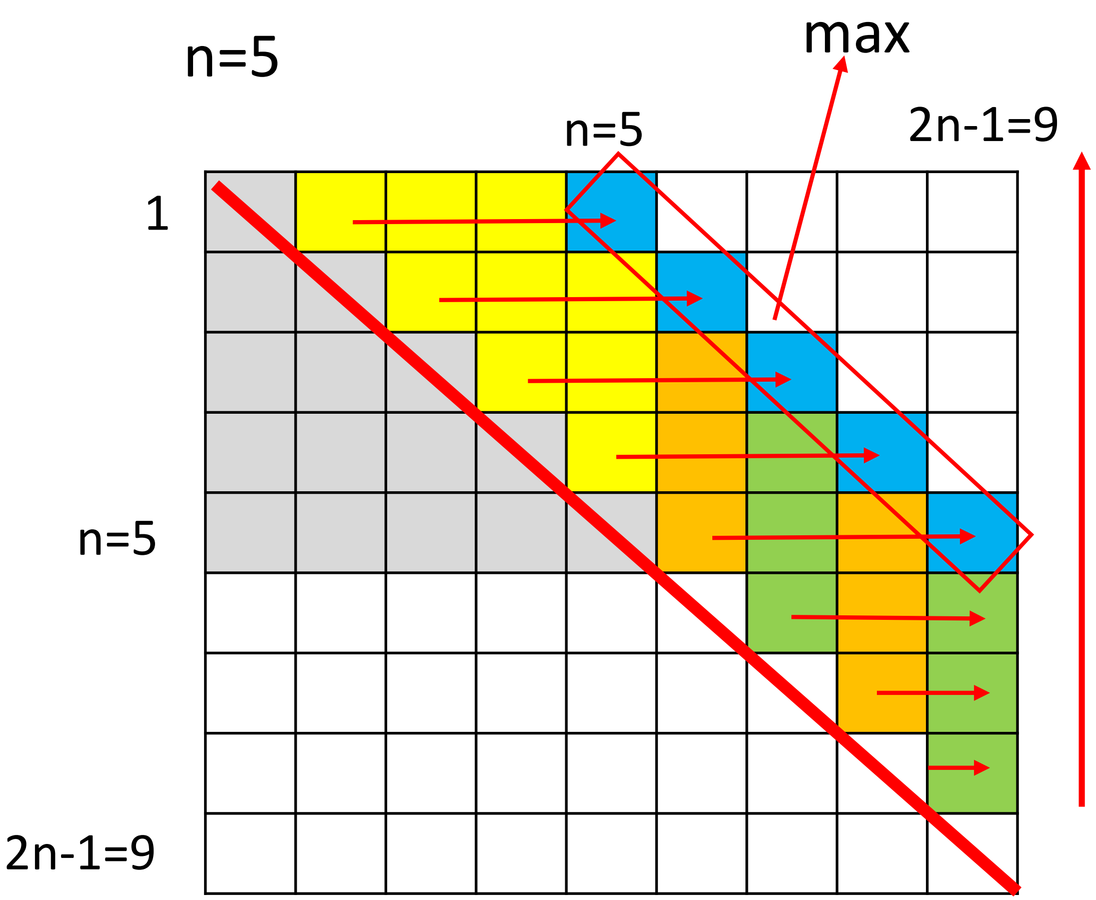
f[i,j]:合并i到j堆的最小得分。

f[i,j]=min{f[i,k]+f[k+1,j]}+s[j]-s[i-1] .
(i<=k<=j-1)

目标: ans=min{f[1,n],f[2,n+1],...,f[n,2n-1]}

时间: O(n^3)}

n=5



```
memset(f,0x3f,sizeof(f));
memset(g,0,sizeof(g));
cin>>n;
for(int i=1;i<=n;i++)cin>>a[i],a[i+n]=a[i];
for(int i=1;i<2*n;i++)s[i]=s[i-1]+a[i],f[i][i]=0;
for(int i=2*n-2;i>=1;i--)
    for(int j=i+1;j<=min(2*n-1,n+i-1);j++){
        for(int k=i;k<j;k++){
            f[i][j]=min(f[i][j],f[i][k]+f[k+1][j]+s[j]-s[i-1]);
            g[i][j]=max(g[i][j],g[i][k]+g[k+1][j]+s[j]-s[i-1]);
        }
    }
int ans1=f[1][n],ans2=g[1][n];
for(int i=2;i<=n;i++){
    ans1=min(ans1,f[i][i+n-1]);
    ans2=max(ans2,g[i][i+n-1]);
}
cout<<ans1<<endl;
cout<<ans2<<endl;
```

训练题目

P1063 能量项链（NOIP2006）

P4342 [IOI1998]Polygon

T127645：括号序列1

UVA1626 括号序列 Brackets sequence

P1220 关路灯

UVA1362 Exploring Pyramids ch5302