动态规划算法3

2018.06.2

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、乘积最大等。

1274：石子合并

- 有N堆石子(N≤100排成一排。现要将石子合并成一堆.规定每次只能选相临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。

- 如:
- 4
- 1 3 5 2
- 最小得分：22

贪心算法：

每次合并相邻两堆和最小的那两堆。

反例：7　4　4　7

- 贪心：8+15+22=45
- 正确：11+11+22=44

应该怎么合并呢？

8　3　6

- 3堆石子合并方案：
- 11+（11+8）=30
- 9+（8+9）=26
- Ans=Min(30, 26)=26

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

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

- (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

重叠子问题和最优子结构性质

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。
- a[i],……,a[j]　　共有j-i+1堆石子
- sum[i,j]=a[i]+a[i+1]+…+a[j]
- 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]

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
- 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]>0)return f[i][j];
    f[i][j]=INF;
    for(int k=i;k<j;k++)
        f[i][j]=min(f[i][j],dp(i,k)+dp(k+1,j)+s[j]-s[i-1]);
    return f[i][j];
}
```

f[i, j]:第i到第j堆间合并为一堆的最小代价。

f[i, j]:={f[i, k]+f[k+1, j]}+sum[i, j]

	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

$$f[2,5]=\min(f[2,2]+f[3,5];f[2,3]+f[4,5];f[2,4]+f[5,5])+sum[2,5]=\min(20,10+7,25)+17=34$$

	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：合并第 i堆 到第 j堆

沿着对角线求：

外层循环变量d：从i开始的连续d堆石子

```
for(int d=2;d<=n;d++)//j-i+1=d
    for(int i=1;i<=n-d+1;i++){//i+d-1<=n
        int j=i+d-1;
        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]);
    }
```

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

总结本题：

- 1、前缀和的应用。
- 2、区间的dp的求解方法：
- 是以区间长度的大小划分阶段。
- 注意求解的顺序。

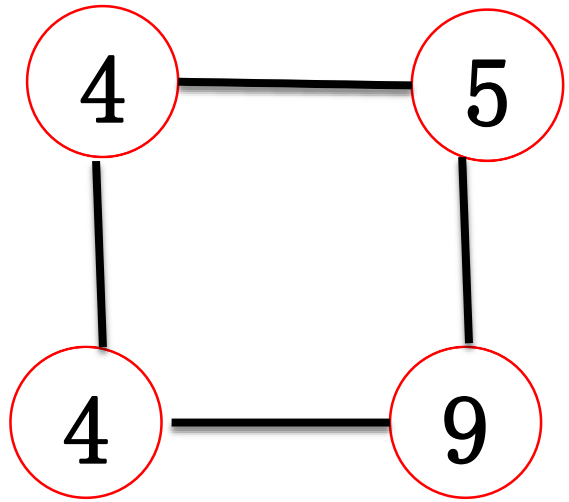
扩展一下：

NOI95

- 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

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]}
- time O(n^3)}

P1063 能量项链（NOIP2006）

例11：括号序列

- 定义一个合法的括号序列：
- （1）空序列是合法的
- （2）假如S是一个合法的序列，则（S）和[S]都是合法的
- （3）假如A 和 B 都是合法的，那么AB和BA也是合法的
- 例如以下是合法的括号序列：
- (), [], (()), ([]), ()[], ()[()]
- 以下是不合法括号序列的：
- (, [,],)(, ([), ([(), ([)]
- 给定一些由‘(’，‘)’，‘[’，‘]’构成的序列，求最少添加多少个括号，能得到一个合法的括号序列。

- 【输入】
- 序列s(长度<=100).
- 【输出：】
- 使序列s成为合法序列添加最少的括号数量。
- 【样例输入】
- ([()
- 【样例输出】
- 2
- 【样例说明】最少好添加2个括号可以得到合法的序列：
- ()[()]或([()])或([])()

分析：

- 设括号序列：$S_iS_{i+1}\cdots..S_{j-1}S_j$
- 最少添加f[i,j]个括号变成合法的括号序列。
- 最后一次把不合法的S变为合法的之前可能情况：
- 1）S形如（S′）或[S′]：
- 只需把S′变合法即可。
- $f[i,j]= f[i+1,j-1]$
- 2）S形如（S′ 或[S′ ：
- 先把S′变为合法的，右边加）或]即可。
- $f[i,j]= f[i+1,j]+1$

- 3）S形如 S′）或S′]：
- 先把S′化为合法的，左边加（或[即可。
- $f[i,j]= f[i,j-1]+1$
- 4）把长度大于1的序列$S_iS_{i+1}\cdots..S_{j-1}S_j$分为两部分：
- $S_i......\ S_k,\ S_{k+1}\cdots..\ S_j$
- 分别化为规则序列.
- 则：$f[i,j]=f[i,k]+f[k+1,j]$ ；$i<=k<=j-1$;
- 上述4种情况取最小值即可。

动态规划方程：

$$S_i S_{i+1} \cdots\cdots S_{j-1} S_j$$

- f[i, j]:=
- min{f[i+1, j-1] ； s[i]与s[j]恰好匹配
- f[i+1, j]+1 ； s[i]=(或[，则右边加)或]
- 1+f[i, j-1] ； s[j]=)or]，则左边加)或]
- f[i, k]+f[k+1, j] ； i<=k<=j-1
- }