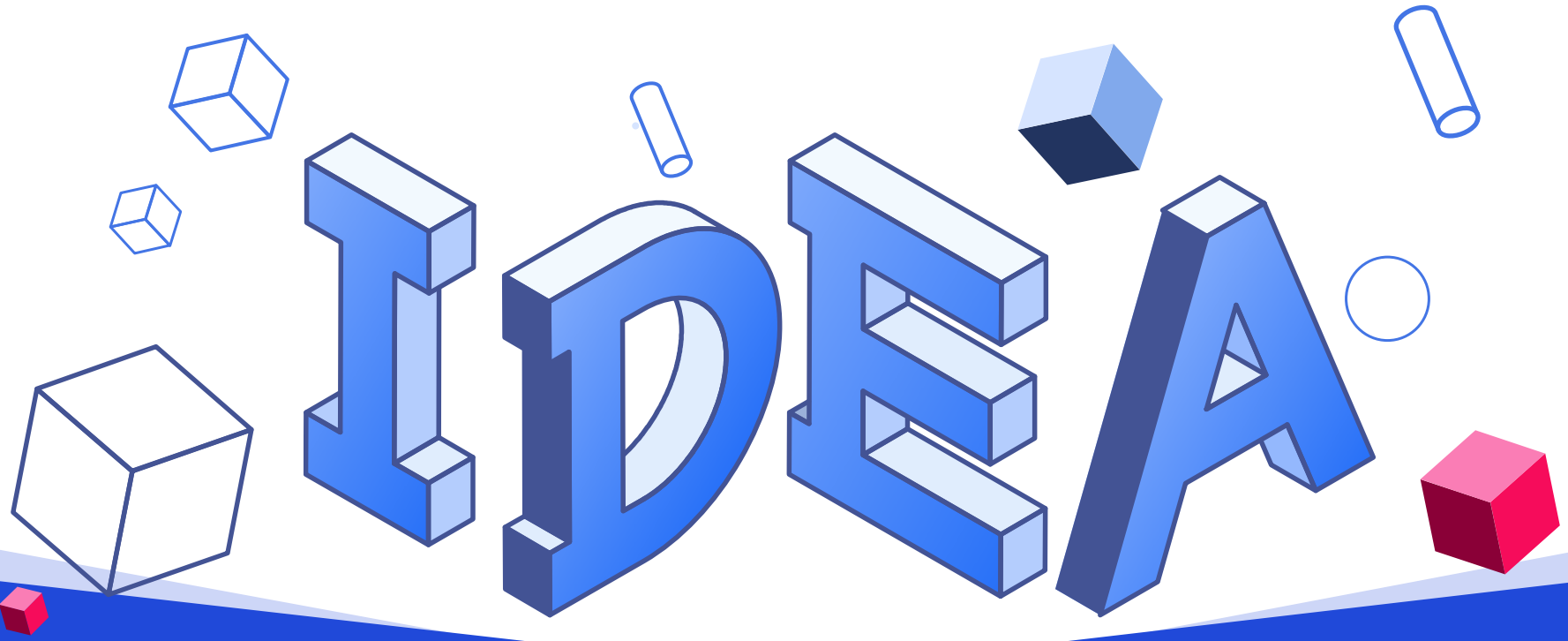


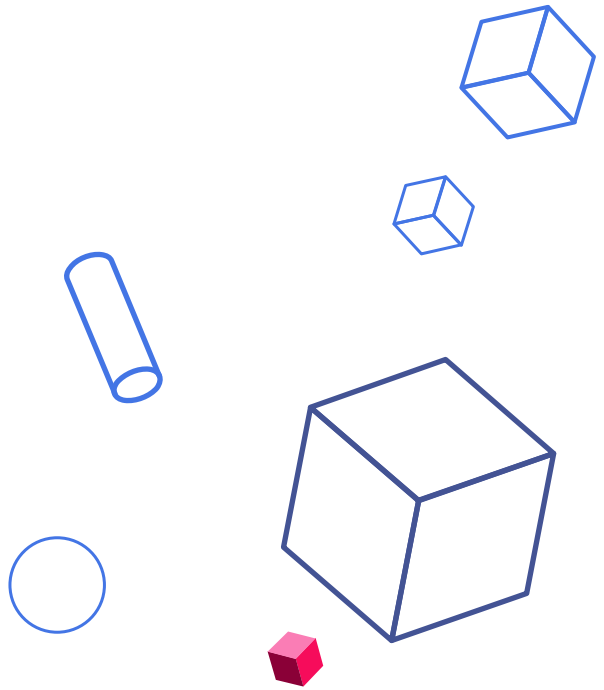
信息学

其他类型DP

西南大学附属中学校

信息奥赛教练组

线性DP



线性DP

状态转移方法是一个线性的转移，每一行依次求解。

LIS和LCS属于线性DP

参考练习题单：https://blog.csdn.net/u011815404/article/details/81870275

最大连续子序列和

问题：对于给定序列 a1,a2,a3……an 寻找它的连续的最大和子数组

状态定义 f[i]：前i个数的最大和

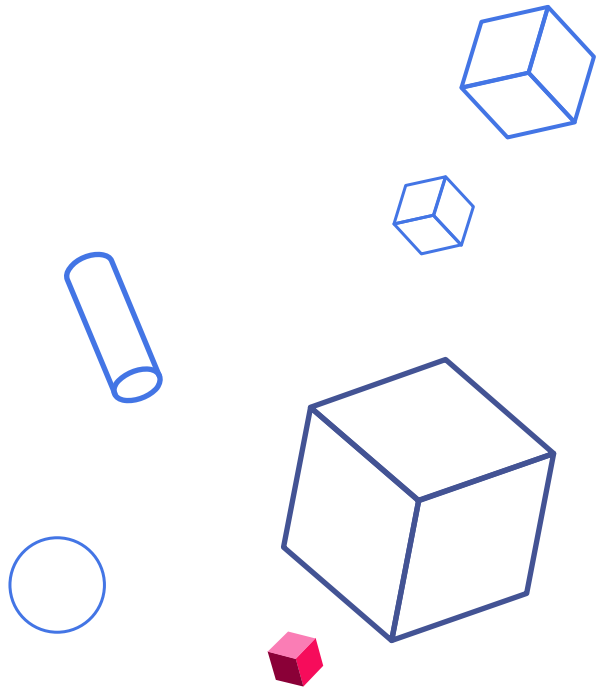
转移方程：

```
如果f[i-1]>0
        f[i]=f[i-1]+a[i]
否则
        f[i]=a[i]
```

二维进阶：最大子矩阵和(请先自行学习一下)

FZOI
西大附中信息学竞赛
High School Affiliated to Southwest University

西南大学附属中学
High School Affiliated to Southwest University



划分型DP

数的划分

题目描述
将整数 n 分成 k 份，且每份不能为空，任意两份不能相同(不考虑顺序)。 例如：
n=7，k=3，下面三种分法被认为是相同的。

1,1,5; 1,5,1; 5,1,1;
问有多少种不同的分法。

输入
n,k(6<n≤200,2≤k≤6)
输出
一个整数，即不同的分法。

样例输入
7 3
样例输出
4

之前的解法：递归、搜索

数的划分(增强版)

题目描述
将整数 n 分成 k 份，且每份不能为空，任意两份不能相同(不考虑顺序)。 例如：
n=7，k=3，下面三种分法被认为是相同的。

1,1,5; 1,5,1; 5,1,1;
问有多少种不同的分法。

输入
n,k(6<n≤3000,2≤k≤3000)
输出
一个整数，即不同的分法。

样例输入
7 3
样例输出
4

之前的解法：递归、搜索
很明显就超时了
尝试用DP解决一下

思考一下题目的DP状态

状态定义：dp[i][j]代表i分成j个部分有几种分法

问题分为两种情况：
1．因为盒子不加区分，那么含1的情况数与“将i-1个小球放到j-1个盒子中”的情况数一样
dp[i][j] = dp[i-1][j-1]
2．没有一个盒子只有一个小球，那么把每个盒子中拿出来一个小球，对应的是“把(n-k)个小球放到k个盒子中的情况数”
dp[i][j] = dp[i-j][j]

最终，和递归的过程类似，dp时可能一直走第一种情况，也可能一直走第二种情况，也可能混着走。

综上，依据分类加法原理：dp[i][j]=dp[i-1][j-1]+dp[i-j][j]; (i>=j)

边界条件： dp[0][0]=1;

核心代码


```
dp[0][0]=1;
for(int i=1;i<=n;i++)  //阶段为n
   for(int j=1;j<=k;j++)
            if(i>=j)
              dp[i][j]=dp[i-1][j-1]+dp[i-j][j];
```

时间复杂度O（n^2）

乘积最大

今年是国际数学联盟确定的“2000——世界数学年”，又恰逢我国著名数学家华罗庚先生诞辰90周年。在华罗庚先生的家乡江苏金坛，组织了一场别开生面的数学智力竞赛的活动，你的一个好朋友XZ也有幸得以参加。活动中，主持人给所有参加活动的选手出了这样一道题目: 设有一个长度N的数字串，要求选手使用K个乘号将它分成K+1个部分，找出一种分法，使得这K+1个部分的乘积能够为最大。
同时，为了帮助选手能够正确理解题意，主持人还举了如下的一个例子： 有一个数字串: 312，当N=3，K=1时会有以下两种分法： 1）3*12=36 2）31*2=62 这时，符合题目要求的结果是： 31*2=62 现在，请你帮助你的好朋友XZ设计一个程序，求得正确的答案。

输入
第一行共有2个自然数N,K (6< =N< =40，0< =K< =5)
第二行是一个长度为N的数字串。
输出
结果输出到文件，相对于输入，应输出所求得的最大乘积（一个自然数）。
样例
样例输入1
4 2
1231
样例输出1
62

思考一下题目的状态

分析

dp[i][j]：前i个数添加j个乘号能得到的最大的乘积

1*23　　12*3

如何找出最优的乘号位置?

枚举k表示第j个乘号在第k位后面
DP信息流动时，比较在当前位置k插入*和不插入*的大小比较
动态转移方程:dp[i][j]=max(dp[i][j],dp[k][j-1]*k位后面的数字大小)

如何快速知道k位后面的数字大小?

数组a[i][j]表示i个字符到第j个字符代表的数

dp[i][j]=max(dp[i][j],dp[k][j-1]*a[k+1][i])

边界条件：dp[i][0] = a[1][i];

```
for(int i=1;i<=n;i++){
  for(int j=i;j<=n;j++){
    a[i][j]=a[i][j-1]*10+s[j]- '0' ;
  }
}
```

还有一种状态方式dp[i][j][k]:i~j之间使用了k个乘号的最大乘积，可以思考一下

核心代码


```
int n,m;
string s;
cin>>n>>m;
cin>>s;  //以字符串的形式输入这个数便于分割
for(int i=1;i<=n;i++){  //预处理a
   for(int j=i;j<=n;j++){
            a[i][j]=a[i][j-1]*10+s[j]- '0' ;
        }
}
for(int i=1;i<=n;i++)  dp[i][0]=a[1][i];  //dp初始化
for(int i=1;i<=n;i++){  //以长度作为阶段
  for(int j=1;j<=m;j++){  //枚举乘号个数
    for(int k=1;k<i;k++){ //寻找乘号的位置
            dp[i][j]=max(dp[i][j],dp[k][j-1]*a[k+1][i]);
          }
        }
}
```

划分型DP

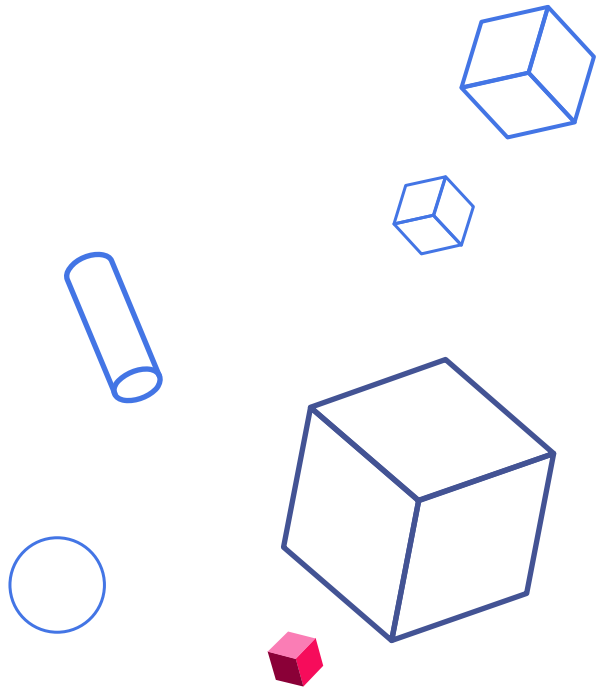
划分dp，顾名思义，就是把需要处理的数据划分成许多小段，让这些小段满足解题条件。

所以做这类题，一般分成两个部分。
第一：数据处理，一般会借助辅助数组，根据题目要求，把数组处理成需要的内容。（要求n个数字的m段划分，使m个部分内部的和相乘，使乘积最大，我们就需要一个数组储存前i个和）。

第二：建立dp方程推出结果。

FZOI
西大附中信息学竞赛
High School Affiliated to Southwest University

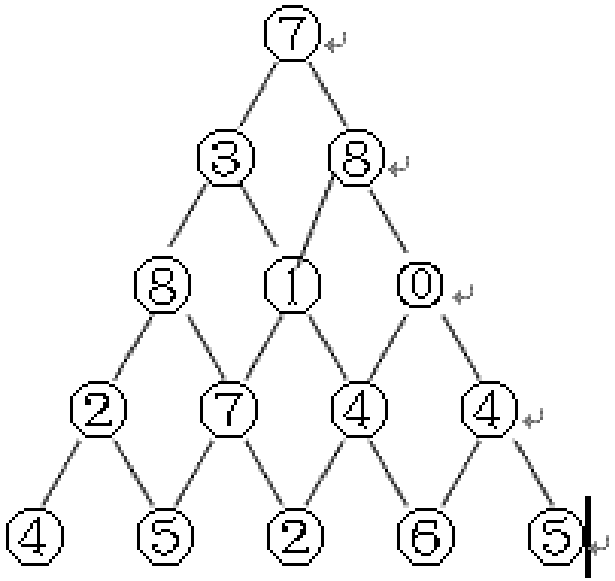
西南大学附属中学
High School Affiliated to Southwest University



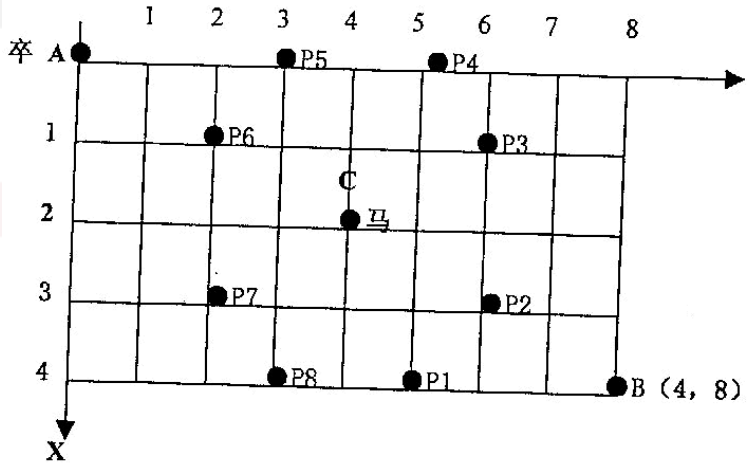
坐标型DP

坐标型DP

其实大家在递推的时候，已经接触这一类问题了



数字三角形



过河卒

他们的状态是？　　dp[i][j]：　到(i,j)点的最优值

方格取数

设有N*N的方格图(N<=10,我们将其中的某些方格中填入正整数,而其他的方格中则放入数字0。如下图所示
某人从图的左上角的A 点出发，可以向下行走，也可以向右走，直到到达右下角的B点。在走过的路上，他可以取走方格中的数（取走后的方格中将变为数字0）。
此人从A点到B 点共走两次，试找出2条这样的路径，使得取得的数之和为最大。
输入
输入的第一行为一个整数N（表示N*N的方格图），接下来的每行有三个整数，前两个表示位置，第三个数为该位置上所放的数。一行单独的0 0 0表示输入结束。
输出
只需输出一个整数，表示2条路径上取得的最大的和。

样例
样例输入1
8
2 3 13
2 6 6
3 5 7
4 4 14
5 2 21
5 6 4
6 3 15
7 2 14
0 0 0
样例输出1
67

A	1	2	3	4	5	6	7	8	
1	0	0	0	0	0	0	0	0	
2	0	0	13	0	0	6	0	0	
3	0	0	0	0	7	0	0	0	
4	0	0	0	14	0	0	0	0	
5	0	21	0	0	0	4	0	0	
6	0	0	15	0	0	0	0	0	
7	0	14	0	0	0	0	0	0	
8	0	0	0	0	0	0	0	0	B

思考一下题目的状态

分析

题意：给出一个方格里面有些数字，请找出两条路径之和最大

本题用dfs是不行的，因为第一条路径的选取会对第二条路径产生影响

标签：坐标型DP，多线程DP，多维DP

分析

问题转化：虽然题目中的2次路径是分开走，但是我们可以转化为两个人分别从A走到B的最优值

状态定义：dp[i,j,h,k]表示第一条道路走到i,j点，第二条走到h,k点时的最优值，

然后判断i,j点，h,k点是由上或左得来最大值

显然i,j,h,k有2*2四种状态：

dp[i-1,j,h-1,k]

dp[i-1,j,h,k-1]

dp[i,j-1,h-1,k]

dp[i,j-1,h,k-1]

之后取这四种最优解的情况下，还要加上a[i,j]和a[h,k]

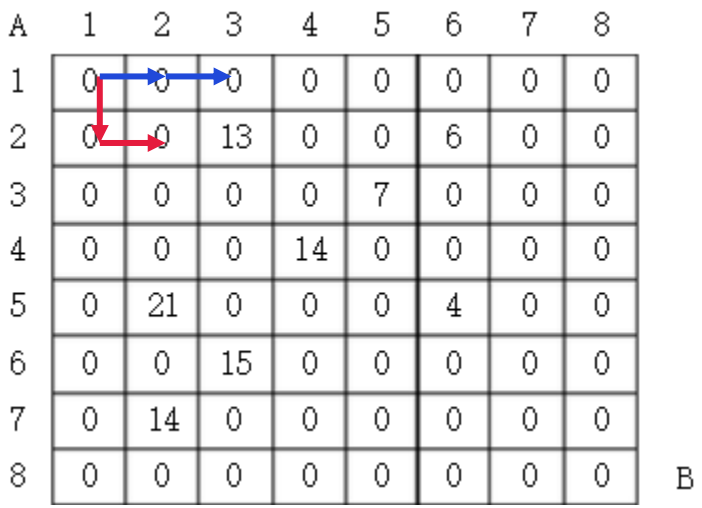
不过这是在i,j与h,k不相等的时候，如果相等则只用加一次即可。

A	1	2	3	4	5	6	7	8	
1	0	0	0	0	0	0	0	0	
2	0	0	13	0	0	6	0	0	
3	0	0	0	0	7	0	0	0	
4	0	0	0	14	0	0	0	0	
5	0	21	0	0	0	4	0	0	
6	0	0	15	0	0	0	0	0	
7	0	14	0	0	0	0	0	0	
8	0	0	0	0	0	0	0	0	B

解法二

另一种状态定义 f[k][x1][y1][x2][y2]:表示走k步，分别到x1，y1,x2,y2的最优值

状态优化:
我们发现，x1+y1==x2+y2==步数k
这样的话，y1、y2就没必要添加到状态中了，我们就可以把五维转三维了：f[k][x1][x2]



A	1	2	3	4	5	6	7	8
1	0	0	0	0	0	0	0	0
2	0	0	13	0	0	6	0	0
3	0	0	0	0	7	0	0	0
4	0	0	0	14	0	0	0	0
5	0	21	0	0	0	4	0	0
6	0	0	15	0	0	0	0	0
7	0	14	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0

B

传纸条

题目描述：
小渊和小轩是好朋友也是同班同学，他们在一起总有谈不完的话题。一次素质拓展活动中，班上同学安排做成一个m行n列的矩阵，而小渊和小轩被安排在矩阵对角线的两端，因此，他们就无法直接交谈了。幸运的是，他们可以通过传纸条来进行交流。纸条要经由许多同学传到对方手里，小渊坐在矩阵的左上角，坐标(1,1)，小轩坐在矩阵的右下角，坐标(m,n)。从小渊传到小轩的纸条只可以向下或者向右传递，从小轩传给小渊的纸条只可以向上或者向左传递。
在活动进行中，小渊希望给小轩传递一张纸条，同时希望小轩给他回复。班里每个同学都可以帮他们传递，但只会帮他们一次，也就是说如果此人在小渊递给小轩纸条的时候帮忙，那么在小轩递给小渊的时候就不会再帮忙。反之亦然。
还有一件事情需要注意，全班每个同学愿意帮忙的好感度有高有低（注意：小渊和小轩的好心程度没有定义，输入时用0表示），可以用一个0-100的自然数来表示，数越大表示越好心。小渊和小轩希望尽可能找好心程度高的同学来帮忙传纸条，即找到来回两条传递路径，使得这两条路径上同学的好心程度只和最大。现在，请你帮助小渊和小轩找到这样的两条路径。
输入格式：
输入第一行有2个用空格隔开的整数m和n，表示班里有m行n列（1<=m,n<=50）。
接下来的m行是一个m*n的矩阵，矩阵中第i行j列的整数表示坐在第i行j列的学生的好心程度。每行的n个整数之间用空格隔开。
输出格式：
输出共一行，包含一个整数，表示来回两条路上参与传递纸条的学生的好心程度之和的最大值。
输入样例：
3 3
0 3 9
2 8 5
5 7 0
输出样例：
34

思考一下题目的状态

分析

状态定义dp[x1][y1][x2][y2]：x1，y1表示第一个纸条；x2，y2表示第二个纸条。
数组里存储的就是两个纸条当前状态下最大好心度。

然后枚举每个可能的x1、y1、x2、y2计算对应的 dp[x1][y1][x2][y2]
状态转移方程就是当前两个位置的好心度之和加上可以到达现在位置的历史最大值
dp[x1][y1][x2][y2]=max(dp[x1-1][y1][x2-1][y2],dp[x1-1][y1][x2][y2-1],
dp[x1][y1-1][x2-1][y2], dp[x1][y1-1][x2][y2-1])+a[x1][y1]+a[x2][y2] 。

枚举时候需要注意两个点的坐标不可以相同，因为这种位置状态不可能出现。

优化：根据方格取数的思想，维度也可以降低
dp[k][i][j]:走了k步，第一个纸条到了横坐标i的位置，第二个纸条到了横坐标j的位置

状态转移方程：dp[k][i][j]=Max(dp[k-1][i][j],dp[k-1][i-1][j],dp[k-1][i][j-1],dp[k-1][i-1][j-1])+a[k-i][i]+a[k-j][j];

坐标型DP

规则型动态规划有一个共性，在一个矩阵中(一般是二维矩阵)给出一些规则，然后按照规则去做某些决策。坐标规则型DP是以坐标为状态进行的转移，比如数字三角形$f(i,j)$表示到达(i,j)的最优值。

当然坐标型DP不仅仅只限于2维状态。一般的状态转移方程可以这样描述：

$f(i,j) = max\{(f(i-1,k))\} +$ 决策，k为题目规则数

总结

入门组所需要掌握的DP模型差不多就是这些
但DP考查得很活，不光要理解模型，还要总结解题思路
达到活学活用

Thanks

For Your Watching