区间动规 1

区间动规也是较常遇到的动态规划类型。在此，我们再将区间动规简单分为两类：区间分割类型和区间合并类型，本节先探讨区间分割类型。

所谓区间分割类型，是指在长度为 n 的线性序列上，分割为 k 部分，每个部分对应的权值不同，问如何分割才能使总权值最大。

我们定义 f[i][j]表示长度为 i 序列，分割为 j 部分所能获得的最大权值，这是个前 i 前 j 问题，那么我们考虑最后一个分割，即第 j 个分割，肯定存在一个分割点 k，使得 k+1 到 i 构成第 j 部分，获得权值和为 w[k+1][i]，那么问题转化为前面长度为 k 的序列分割成 j-1 个部分获得的最大权值。状态转移方程为 f[i][j]=max(f[k][j-1]+w[k+1][i])|j<=k<=i。

通过以下题目，可以更好的理解区间分割类型动态规划。

摆花

【题目描述】
小明的花店新开张，为了吸引顾客，他想在花店的门口摆上一排花，共 m 盆。通过调查顾客的喜好，小明列出了顾客最喜欢的 n 种花，从 1 到 n 标号。为了在门口展出更多种花，规定第 i 种花不能超过 ai 盆， 摆花时同一种花放在一起，且不同种类的花需按标号的从小到大的顺序依次摆列。
试编程计算，一共有多少种不同的摆花方案。
【输入】
第一行包含两个正整数 n 和 m，中间用一个空格隔开。
第二行有 n 个整数，每两个整数之间用一个空格隔开，依次表示 a1、a2、……an。
【输出】
输出只有一行，一个整数，表示有多少种方案。注意：因为方案数可能很多，请输出方案数对 1000007 取模的结果。
【样例输入】
2 4
3 2
【样例输出】
2
【输入输出样例说明】
有 2 种摆花的方案，分别是(1， 1， 1， 2)， (1， 1， 2， 2)。 括号里的 1 和 2 表示两种花，比如第一个方案是前三个位置摆第一种花，第四个位置摆第二种花。
【数据范围】
对于 20%数据，有 0<n≤8，0<m≤8，0≤ai≤8；
对于 50%数据，有 0<n≤20，0<m≤20，0≤ai≤20；
对于 100%数据，有 0<n≤100，0<m≤100，0≤ ai≤100。
【分析】
因为花是按从小到大的顺序摆放的，无后效性问题，我们可以设 f[i][j]表示前 i 种花摆满 j 盆

的方案数。那么考虑第 i 种花，第 i 种花有 a[i]个，第 i 种花可以不放，放 1 个，放 2 个，最多放 a[i]个，我们假设第 i 种花放 k 个，那么问题转化为前 j-k 盆花用 i-1 种花摆放的方案数。状态转移方程为 f[i][j]=∑f[i-1][j-k]|0<=k<=min(a[i],j)。

【代码】

```
#include<iostream>
using namespace std;
int n,m,a[101],f[101][101];
int main(){
    cin>>n>>m;
    for(int i=1;i<=n;i++)cin>>a[i];
    f[0][0]=1;
    for(int i=1;i<=n;i++){
        f[i][0]=1;
        for(int j=1;j<=m;j++){
            for(int k=0;k<=min(a[i],j);k++){
                f[i][j]=(f[i][j]+f[i-1][j-k])%1000007;
            }
        }
    }
    cout<<f[n][m]%1000007;
}
```

乘积最大

【题目描述】

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】

程序的输入共有两行：

第一行共有 2 个自然数 N，K (6<=N<=40，1<=K<=6)

第二行是一个 K 度为 N 的数字串。

【输出】

屏幕输出(结果显示在屏幕上)，相对于输入，应输出所求得的最大乘积（一个自然数）。

【样例输入】
4 2
1231
【样例输出】
62
【分析】
区间分割问题。解这类问题，难点反而在如何求区间权值上面。设 f[i][j]表示前 i 个数字用了 j 个乘号所获得的最大乘积，设最后一个乘号在 k 和 k+1 之间，问题转化为 f[k][j-1]*w[k+1][i]，f[k][j-1]表示前 k 个数字用了 j-1 个乘号所获得的最大乘积，w[k+1][i]表示第 k+1 个数字到第 i 个数字所组成的整数。
【代码】

```
#include<iostream>
using namespace std;
long long f[41][8],a[41][41],n,L;
char num[41];
int main(){
    cin>>n>>L;
    for(int i=1;i<=n;i++)cin>>num[i];
    for(int i=1;i<=n;i++){  //a[i][j]表示第 i 个字符到第 j 个字符对应的整数
        a[i][i]=num[i]-'0';
        for(int j=i+1;j<=n;j++){
            a[i][j]=a[i][j-1]*10+num[j]-'0';
        }
    }
    for(int i=1;i<=n;i++){  //边界
        f[i][0]=a[1][i];
    }
    for(int i=1;i<=n;i++){
        for(int j=1;j<=L;j++){
            for(int k=j;k<i;k++){
                f[i][j]=max(f[i][j],f[k][j-1]*a[k+1][i]);
            }
        }
    }
    cout<<f[n][L];
}
```

邮局

【题目描述】
一些村庄建在一条笔直的高速公路边上，我们用一条坐标轴来描述这条公路，每个村庄的坐标都是整数，没有两个村庄的坐标相同。两个村庄的距离定义为坐标之差的绝对值。我们需要在某些村庄建立邮局。使每个村庄使用与它距离最近的邮局，建立邮局的原则是：所有村

庄到各自使用的邮局的距离总和最小。
数据规模：1<=村庄数<=300, 1<=邮局数<=30, 1<=村庄坐标<=10000
【输入】
2 行
第一行:n m {表示有 n 个村庄，建立 m 个邮局}
第二行:a1 a2 a3 .. an {表示 n 个村庄的坐标}
【输出】
1 行
第一行:l {l 表示最小距离总和}
【样例输入】
10 5
1 2 3 6 7 9 11 22 44 50
【样例输出】
9
【分析】
区间分割问题。设 f[i][j]表示前 i 个村庄建 j 个邮局的最小距离和，那么我们考虑最后一个邮局，设从第 k+1 个村庄到第 i 个村庄共用第 j 个邮局，那么问题转化为 f[k][j-1]+w[k+1][i]，其中 f[k][j-1]表示前 k 个村庄用 j-1 个邮局的最小距离和，w[k+1][i]表示从第 k+1 到第 i 个村庄建一个邮局所需的最小距离和，其值为 k+1 到 i 每个数到它的中位数的和。
【代码】

```
#include<iostream>
#include<algorithm>
#include<cstring>
using namespace std;
int n,m,a[301],f[301][31],d[301][301];
int main(){
    cin>>n>>m;
    for(int i=1;i<=n;i++)cin>>a[i];
    sort(a+1,a+n+1);  //村庄按坐标从小到大排序
    for(int i=1;i<=n;i++){  //d[i][j]表示从 i 村庄到 j 村庄的最小距离和
        d[i][i]=0;
        for(int j=i+1;j<=n;j++){  //递推求解
            d[i][j]=d[i][j-1]+(a[j]-a[(i+j)/2]);
        }
    }
    memset(f,127,sizeof(f));
    for(int i=1;i<=n;i++){  //边界初始化
        f[i][1]=d[1][i];
    }
    for(int j=2;j<=m;j++){
        for(int i=1;i<=n;i++){
            for(int k=j-1;k<i;k++){
                f[i][j]=min(f[i][j],f[k][j-1]+d[k+1][i]);
            }
```

```
        }
    }
    cout<<f[n][m];
}
```

观光游览

【题目描述】
一条街道被分成 m 格（1<=m<=100），还有 n 个景点（1<=n<=100），分布在街道上。每个景点可以占据连续的若干格，并且有一个美学值 v（0<v<=100）。现要组织 k 个人考察这个街道（1<=k<=m），每个人考察的区域是连续的若干格（不可为 0 格），且任意两个人考察的区域不得相交，也不得有一个格子无人考察。对于任意一个人，如果它考察的区域中有一个风景点（风景点必须完整的位于这个区域），则它就得到了这个风景点的分值（美学值）。
你的任务是将街道的 m 个格子分给 k 个人去考察，使得总的分值最大。
【输入】
第一行一个整数 m，表示街道的长度。
第二行一个整数 n，表示风景点个数。
此后 n 行，每行描述一个风景点，三个整数 x、y 和 v，表示该风景点是从第 x 个格子到第 y 个格子，美学值为 v。
最后一行一个整数 k，表示考察的人数。
【输出】
一个整数，表示最大可以得到的分值。
【样例输入】
3
2
1 2 2
2 3 3
2
【样例输出】
3
【分析】
同样区间分割问题。设 f[i][j]表示长度为 i 的街道给 j 个人考察的最大得分，设第 j 个人考察的区间为 k+1 到 i，得分为 w[k+1][i]，那么问题转化为 f[k][j-1]+w[k+1][i]。难点在于如何预处理每个区间的分值。
【代码】

```
#include<iostream>
using namespace std;
int m,n,k,x,y,val,v[101][101],f[101][101];
int main(){
    cin>>m>>n;
    for(int i=1;i<=n;i++){
        cin>>x>>y>>val;  //以下预处理每段区间的分值
        for(int i=1;i<=x;i++){  //向小于 x 的方向扩展
```

oiClass

```
            for(int j=y;j<=m;j++){  //向大于 y 的方向扩展
                v[i][j]+=val;  //凡包含 x，y 这段区间的分值都加 val
            }
        }
    }
    cin>>k;
    for(int i=1;i<=m;i++){
        for(int j=1;j<=k;j++){
            for(int l=j-1;l<i;l++){
                f[i][j]=max(f[i][j],f[l][j-1]+v[l+1][i]);
            }
        }
    }
    cout<<f[m][k];
}
```

抄书

【题目描述】
现在要把 m 本有顺序的书分给 k 个人复制（抄写），每一个人的抄写速度都一样，一本书不允许给两个（或以上）的人抄写，分给每一个人的书，必须是连续的，比如不能把第一、第三、第四本书给同一个人抄写。
现在请你设计一种方案，使得复制时间最短。复制时间为抄写页数最多的人用去的时间。
【输入】
第一行两个整数 m，k；（k≤m≤500）
第二行 m 个整数，第 i 个整数表示第 i 本书的页数。
【输出】
共 k 行，每行两个整数，第 i 行表示第 i 个人抄写的书的起始编号和终止编号。k 行的起始编号应该从小到大排列，如果有多解，则尽可能让前面的人少抄写。
【样例输入】
9 3
1 2 3 4 5 6 7 8 9
【样例输出】
1 5
6 7
8 9
【分析】
因为抄书是连续的，所以可以区间分割的方式处理。设 f[i][j]表示前 i 本书给前 j 个人抄写，设第 j 个人从第 k+1 本抄写到第 i 本，抄书的时间为 sum[i]-sum[k]（预处理书页的前缀和），那么问题转化为前 k 本书给 j-1 人抄写，即 f[i][j]=max(f[k][j-1],sum[i]-sum[k])，在所有 k 中，选最小的，状态转移方程为 f[i][j]=min(f[i][j], max(f[k][j-1],sum[i]-sum[k]) |0<=k<=i)。本书难点还在于怎样输出每个人抄写的起始区间。设变量 pre[i][j]=k 表示前 i 本书给前 j 个人抄写时，前 j-1 人抄写前 k 本书，输出区间时递归处理即可，详见代码。
【代码】

```cpp
#include<iostream>
#include<cstring>
using namespace std;
int m,n,x,sum[501],f[501][501],pre[501][501];
void print(int x,int y,int k){  //表示第 k 个人抄第 x+1 本书到 y 本书
    if(k<1)return;
    print(pre[x][k-1],x,k-1);  //先输出第 k-1 个人的抄书
    cout<<x+1<<" "<<y<<endl;
}
int main(){
    cin>>m>>n;
    for(int i=1;i<=m;i++){
        cin>>x;
        sum[i]=sum[i-1]+x;
    }
    memset(f,127,sizeof(f));
    for(int i=0;i<=m;i++){
        f[i][1]=sum[i];
        pre[i][1]=0;
    }
    for(int i=1;i<=m;i++){
        for(int j=2;j<=n;j++){
            for(int k=0;k<=i;k++){ //k=0 表示第 j 人全抄，k=i 表示第 j 人一本不抄
                int t=max(f[k][j-1],sum[i]-sum[k]);
                if(t<f[i][j]){
                    f[i][j]=t;
                    pre[i][j]=k; //表示前 j-1 人抄前 k 本书，即第 j 人从 k+1 书开始
                }
            }
        }
    }
    print(pre[m][n],m,n);
}
```

花店橱窗

【题目描述】

某花店现有 F 束花，每一束花的品种都不一样，同时至少有同样数量的花瓶，被按顺序摆成一行，花瓶的位置是固定的，从左到右按 1 到 V 顺序编号，V 是花瓶的数目。花束可以移动，并且每束花用 1 到 F 的整数标识。如果 I<J，则花束 I 必须放在花束 J 左边的花瓶中。例如，假设杜鹃花的标识数为 1，秋海棠的标识数为 2，康乃馨的标识数为 3，所有花束在放入花瓶时必须保持其标识数的顺序，即杜鹃花必须放在秋海棠左边的花瓶中，秋海棠必须放在康乃馨左边的花瓶中。如果花瓶的数目大于花束的数目，则多余的花瓶必须空，即每个花瓶只能放一束花。

每个花瓶的形状和颜色也不相同，因此，当各个花瓶中放入不同的花束时，会产生不同的美学效果，并以美学值（一个整数）来表示，空置花瓶的美学值为 0。在上述的例子中，花瓶与花束的不同搭配所具有的美学值，可以用如下的表格来表示：

	花瓶 1	花瓶 2	花瓶 3	花瓶 4	花瓶 5
杜鹃花	7	23	-5	-24	16
秋海棠	5	21	-4	10	23
康乃馨	-21	5	-4	-20	20

根据表格，杜鹃花放在花瓶 2 中，会显得非常好看，但若放在花瓶 4 中，则显得很难看。

【输入】

输入的第一行是两个整数 F 和 V，分别为花束数和花瓶数（1≤F≤100，F≤V≤100）。接下来是矩阵 Aij,它有 I 行，每行 J 个整数，Aij 表示花束 I 摆放在花瓶 J 中的美学值。

【输出】

输出的第一行是一个整数，为最大的美学值；接下来有 F 行，每行两个数，为那束花放入那个花瓶的编号。

【样例输入】

3 5

7 23 -5 -24 16

5 21 -4 10 23

-21 5 -4 -20 20

【样例输出】

53

2 4 5

【分析】

因为“所有花束在放入花瓶时必须保持其标识数的顺序”，所以本题可以看成是一个区间分割问题。设 f[i][j]表示前 i 束花插在前 j 个花瓶且第 i 束花在第 j 个瓶子所能获得的最大美学值，那么存在 f[i-1][k]，前 i-1 束花插在前 k 个花瓶且第 i-1 束花插在第 k 个花瓶，问题转化为 f[i][j]=max(f[i-1][k]+a[i][j])|i-1<=k<=j-1。同时，我们设变量 pre[i][j]=k 表示第 i-1 束花插在第 k 个花瓶，输出方案时递归处理即可，详见代码。

【代码】

```
#include<iostream>
using namespace std;
int F,V,a[101][101],f[101][101],pre[101][101];
void print(int x,int y){
    if(pre[x][y]==0){
        cout<<y<<" ";
        return;
    }
    print(x-1,pre[x][y]);
    cout<<y<<" ";
}
int main(){
    cin>>F>>V;
    for(int i=1;i<=F;i++){
```

```
        for(int j=1;j<=V;j++){
            cin>>a[i][j];
        }
    }
    for(int i=1;i<=F;i++){
        for(int j=i;j<=V-F+i;j++){
            f[i][j]=-99999999;
            for(int k=i-1;k<j;k++){
                if(f[i-1][k]+a[i][j]>f[i][j]){
                    f[i][j]=f[i-1][k]+a[i][j];
                    pre[i][j]=k;
                }
            }
        }
    }
    int maxn=0,pos=0;
    for(int i=F;i<=V;i++){
        if(f[F][i]>maxn){
            maxn=f[F][i];
            pos=i;
        }
    }
    cout<<maxn<<endl;
    print(F,pos);
}
```