线性类DP

主讲老师：党东

【线型动态规划】

一、线型动态规划

一个问题的状态可以描绘在一条直线上，即状态可以用一维数组来表示，第i个元素的状态与前i-1个元素的状态有关，前i-1个状态组成一个决策序列，它是其它类型动态规划的基础。

典型的线型动态规划有LIS（最长不下降子序列）,LCS（最长公共子序列），部分和问题，区间选择问题等等。

LIS模型

最长上升子序列(Longest Increasing Subsequence)

【例1】最长上升子序列 --1355

Description

一个数的序列bi，当b1< b2 < ... < bS的时候，我们称这个序列是上升的。对于给定的一个序列(a1,a2,...,aN)，我们可以得到一些上升的子序列(ai1, ai2, ..., aiK)，这里1<=i1 你的任务，就是对于给定的序列，求出最长上升子序列的长度。

Input

输入的第一行是序列的长度N(1<=N<=1000)。第二行给出序列中的N个整数，这些整数的取值范围都在0到10000。

Output

最长上升子序列的长度。

Sample Input

6

1 6 2 5 4 7

Sample Output

4

【例1】最长上升子序列 --1355

【问题分析】

1、阶段和状态：

f[i]：表示以a[i]为结尾的最长上升子序列的最大长度；

阶段i表示前i个数，由于每个阶段只有一个状态，所以用一维数组表示；

2、状态转移方程：

- **初始化：f[i]=1;**

每个元素至少都满足以自身作为长度为1的最长上升子序列。

- **状态转移：f[i]=max{f[j]+1,j<i且a[j]<a[i]}**

利用1~(i-1)的f[j]更新当前位置f[i]的最大长度，如果a[i]>a[j]，表明以a[i]可以接在a[j]后得到更长的LIS。

- **遍历答案：answer=max{f[i]};(1<=i<=n)**

遍历每个元素a[i]结尾得到的最大上升子序列的长度，得到最大长度。

【例1】最长上升子序列 --1355

【图表演示】

初始化：

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14
a[i]	13	7	9	16	38	24	37	18	4	19	21	22	63	15
f[i]	1	1	1	1	1	1	1	1	1	1	1	1	1	1

计算过程：

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14
a[i]	13	7	9	16	38	24	37	18	4	19	21	22	63	15
f[i]	1	1	2	3	4	4	5	4	1	5	6	7	8	3

【例1】最长上升子序列 --1355

【核心代码】

```
cin>>n;
for(i=1;i<=n;i++)
   {cin>>a[i];f[i]=1;}//初始化
for(i=2;i<=n;i++)
  for(j=1;j<i;j++)
    if(a[j]<a[i]&&f[j]+1>f[i]){f[i]=f[j]+1;}
max=1;
for(i=1;i<=n;i++)
   if(f[i]>f[max])max=i;
cout<<f[max]<<endl;
```

【扩展1】最长不下降子序列 --1347

Description

设有整数序列b1,b2,b3,……,bm, 若存在i1< i2 <i3 <…… <in,且bi1<bi2<bi3……<bin, 则称b1,b2,b3,……,bn,中有长度为N的不下降序列bi1，bi2，bi3，……，bin。求序列b1,b2,b3,……,bm中最大不下降序列的长度。

Input

第一行为n,第二行为用空格隔开的n个整数。

Output

第一行为输出最大个数max。

Sample Input

14

13 7 9 16 38 24 37 18 44 19 21 22 63 15

Sample Output

8

【扩展1】最长不下降子序列 --1347

【题目提示】

最长不上子序列，a[i]与a[j]之间不相等

最长不下降子序列，a[i]与a[j]可以相等

【扩展2】最长不下降子序列

Description

设有整数序列b1,b2,b3,……,bm, 若存在i1< i2 <i3 <…… <in,且bi1<bi2<bi3……<bin,则称b1,b2,b3,……,bn,中有长度为N的不下降序列bi1，bi2，bi3，……，bin。求序列b1,b2,b3,……,bm中最大不下降序列的长度。

Input

第一行为一个数n，表示有n个数，第二行为n个整数序列；

Output

第一行为最大长度，第二行为满足长度的序列

Sample Input

14

13 7 9 16 38 24 37 18 4 19 21 22 63 15

Sample Output

8

7 9 16 18 19 21 22 63

【扩展2】最长不下降子序列

【问题分析】

本题需输出具体的子序列，故需记录前后结点信息。

数组pre[i]记录第i个数前一个结点下标。

```
for(i=1;i<=n;i++)
    {cin>>a[i];f[i]=1;pre[i]=i;}//初始化
  for(i=2;i<=n;i++)
    for(j=1;j<i;j++)
      if(a[j]<=a[i]&&f[j]+1>f[i]){f[i]=f[j]+1;pre[i]=j;}
```

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14
a[i]	13	7	9	16	38	24	37	18	4	19	21	22	63	15
f[i]	1	1	2	3	4	4	5	4	1	5	6	7	8	3
pre[i]	1	2	2	3	4	4	6	4	9	8	10	11	12	3

【扩展2】最长不下降子序列

【输出结果】

数组pre[i]记录第i个数前一个结点下标。

```
max=1;
 for(i=1;i<=n;i++)
    if(f[i]>f[max])max=i;
 cout<<f[max]<<endl;
 out(pre,a,max);
```

```
void out(int pre[],int a[],int n) //递归输出序列
{  if (pre[n]==n){cout<<a[n]<<" ";return;}
   out(pre,a,pre[n]);
   cout<<a[n]<<" ";
}
```

【例2】渡轮问题 --1501

Description

有一个国家被一条河划分为南北两部分，在南岸和北岸总共有N对城镇，每一城镇在对岸都有唯一的友好城镇。任何两个城镇都没有相同的友好城镇。每一对友好城镇都希望有一条航线来往。于是他们向政府提出了申请。由于河终年有雾。政府决定不允许有任两条航线交叉(如果两条航线交叉，将有很大机会撞船)。

你的任务是写一个程序来帮政府官员决定他们应拨款兴建哪些航线以使得没有出现交叉的航线最多。

Input

第一行一个整数N(1 <= N <= 2000)，表示分布在河两岸的城镇对数。

接下来的N行每行有两个由空格分隔的正数C，D（C、D<=10^9），描述每一对友好城镇沿着河岸与西边境线的距离，C表示北岸城镇的距离而D表示南岸城镇的距离。在河的同一边，任何两个城镇的位置都是不同的。

【例2】渡轮问题 --1501

Output

在安全条件下能够开通的最大航线数目。

Sample Input

```
7
22 4
2 6
10 3
15 12
9 8
17 17
4 2
```

Sample Output

```
4
```

【例2】渡轮问题 --1501

【问题分析】

1、阶段和状态：题目粗略一看，不能够用动态规划求解，因为具有后效性,但是仔细分析如果先按照a[i]从小到大排序后,解除了后效性,问题变成了对于b[i]求最长上升子序列问题

f[i]:表示以b[i]为最后一个数字的最长不下降序列的最大长度

注意：好多题目都要通过排序解除后效性！！

2、状态转移方程：

初始化：f[i]=1;(1<=i<=n)

f[i]=max{f[j]+1,j<i且b[j]<b[i]}

answer=max{f[i]};(1<=i<=n)

【练习1】合唱队形 --1352

Description

N位同学站成一排，音乐老师要请其中的(N-K)位同学出列，使得剩下的K位同学排成合唱队形。

合唱队形是指这样的一种队形：设K位同学从左到右依次编号为1, 2, …, K，他们的身高分别为T[1], T[2], …, T[K]，则他们的身高满足T[1] < T[2] < … < T[i] > T[i+1] > … > T[K] (1≤i≤K)。

你的任务是，已知所有N位同学的身高，计算最少需要几位同学出列，可以使得剩下的同学排成合唱队形。

Input

输入的第一行是一个整数N（2 ≤ N ≤ 100），表示同学的总数。第二行有n个整数，用空格分隔，第i个整数Ti（130 ≤ Ti ≤ 230）是第i位同学的身高（厘米）。

Output

输出包括一行，这一行只包含一个整数，就是最少需要几位同学出列。

Sample Input

8

186 186 150 200 160 130 197 220

Sample Output

4

【练习1】合唱队形 --1352

【问题分析】

分别计算两次正→ 反← 两个方向LIS。

a[i]表示：第i个同学作为→**方向**最后一个元素时的最长上升子序列长度

b[i]表示：第i个同学作为←**方向**最后一个元素时的最长上升子序列长度

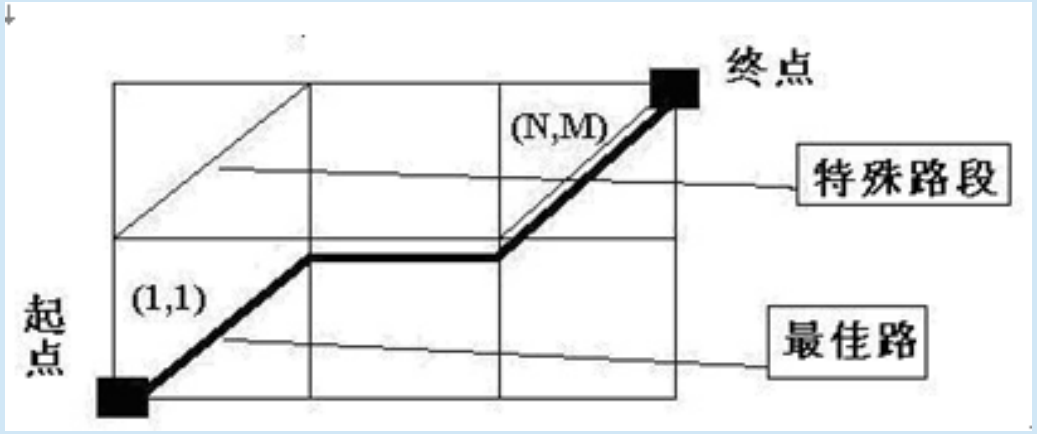
故以第i个同学作为合唱队形最高人时，共能站下**a[i]+b[i]-1**人，即需要**n-(a[i]+b[i]-1)**人出列

【练习2】飞翔 --1502

Description

鹰最骄傲的就是翱翔，但是鹰们互相都很嫉妒别的鹰比自己飞的快，更嫉妒其他的鹰比自己飞行的有技巧。于是，他们决定举办一场比赛，比赛的地方将在一个迷宫之中。

这些鹰的起始点被设在一个N*M矩阵的左下角map[1，1]的左下角。终点被设定在矩阵的右上角map[N,M]的右上角，有些map[i,j]是可以从中间穿越的。每一个方格的边长都是100米。如图所示：



没有障碍，也没有死路。这样设计主要是为了高速飞行的鹰们不要发现死路来不及调整而发生意外。潘帕斯雄鹰冒着减RP的危险从比赛承办方戒备森严的基地中偷来了施工的地图。但是问题也随之而来，他必须在比赛开始之前把地图的每一条路都搞清楚，从中找到一条到达终点最近的路。（哈哈，笨鸟不先飞也要拿冠军）但是此鹰是前无古鹰，后无来鹰的吃菜长大的鹰--菜鸟。他自己没有办法得出最短的路径，于是紧急之下找到了学OI的你，希望找到你的帮助。

【练习2】飞翔 --1502

Input

首行为n,m(0 < n,m<= 100000),第2行为k（0 < k<= 1000）表示有多少个特殊的边。以下k行为两个数，i,j表示map[i,j]是可以直接穿越的。

Output

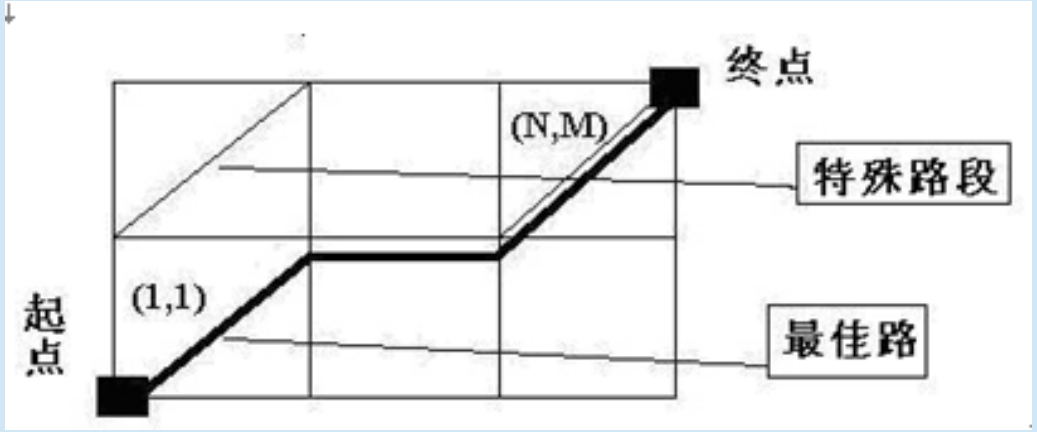
仅一行，1，1-->n,m的最短路径的长度，**四舍五入**保留到整数即可

Sample Input

3 2
3
1 1
3 2
1 2

Sample Output

383



【练习2】飞翔 --1502

【问题分析】

若没有特殊边，则路径长度为m+n（每个格子边长为1）。

每一个特殊边可使路径长度减少2-sqrt(2)，要找到最短路径，只要找出最多的特殊边。

因为鹰是在二维平面上从左下角飞到右上角，所以问题的实质是找二向的最长上升子序列。

先对特殊边坐标排序，然后求最长上升子序列。

【例3】最长上升子序列（二）--1503

Description

LIS 问题是最经典的动态规划基础问题之一。如果要求一个满足一定条件的最长上升子序列，你还能解决吗?

给出一个长度为 N 整数序列，请求出它的**包含第 K个元素**的最长上升子序列。

例如：对于长度为 6的序列< 2,7,3,4,8,5 >，它的最长上升子序列为 <2 ,3,4,5 > ，但如果限制一定要包含第 2个元素，那么满足此要求的最长上升子序列就只能是 <2,7,8 >了。

Input

第一行为两个整数 N,K，如上所述。

接下来是N个整数，描述一个序列。

Output

Sample Input

8 6

65 158 170 299 300 155 207 389

Sample Output

4

【例3】最长上升子序列（二）--1503

【问题1】包含第K个元素。

解决方法：只留下LIS中可以容下第K个元素的数。

```
for(int i=1;i<=n;i++)
{
        if(i<=k&&b[i]<=b[k])  a[++t]=b[i];
        else if(i>=k&&b[i]>=b[k])  a[++t]=b[i];
}
```

【例3】最长上升子序列（二）--1503

【问题2】常规求LIS方法时间复杂度n^2，不够优秀，面对n < = 200000的数据范围，应降低时间复杂度

【例3】最长上升子序列（二）--1503

【优化时间复杂度至nlogn】

举个例子：假设存在一个序列d[1..9] = 2 1 5 3 6 4 8 9 7，可以看出来它的LIS长度为5。

下面一步一步试着找出它。

我们定义一个序列B，然后令 i = 1 to 9 逐个考察这个序列。

此外，我们用一个变量Len来记录现在最长算到多少了

首先，把d[1]有序地放到B里，令B[1] = 2，就是说当只有1一个数字2的时候，长度为1的LIS的最小末尾是2。这时Len=1

然后，把d[2]有序地放到B里，令B[1] = 1，就是说长度为1的LIS的最小末尾是1，d[1]=2已经没用了，很容易理解吧。这时Len=1

接着，d[3] = 5，d[3]>B[1]，所以令B[1+1]=B[2]=d[3]=5，就是说长度为2的LIS的最小末尾是5，很容易理解吧。这时候B[1..2] = 1, 5，Len = 2

再来，d[4] = 3，它正好加在1,5之间，放在1的位置显然不合适，因为1小于3，长度为1的LIS最小末尾应该是1，这样很容易推知，长度为2的LIS最小末尾是3，于是可以把5淘汰掉，这时候B[1..2] = 1, 3，Len = 2

【例3】最长上升子序列（二）--1503

【优化时间复杂度至nlogn】

举个例子：假设存在一个序列d[1..9] = 2 1 5 3 6 4 8 9 7，可以看出来它的LIS长度为5。

继续，d[5] = 6，它在3后面，因为B[2] = 3, 而6在3后面，于是很容易可以推知B[3] = 6, 这时B[1..3] = 1, 3, 6，还是很容易理解吧？ Len = 3 了噢。

第6个, d[6] = 4，你看它在3和6之间，于是我们就可以把6替换掉，得到B[3] = 4。B[1..3] = 1, 3, 4， Len继续等于3

第7个, d[7] = 8，它很大，比4大，嗯。于是B[4] = 8。Len变成4了

第8个, d[8] = 9，得到B[5] = 9，嗯。Len继续增大，到5了。

最后一个, d[9] = 7，它在B[3] = 4和B[4] = 8之间，所以我们知道，最新的B[4] =7，B[1..5] = 1, 3, 4, 7, 9，Len = 5。

于是我们知道了LIS的长度为5。

!!!!! 注意。这个1,3,4,7,9不是LIS，它只是存储的对应长度LIS的最小末尾。有了这个末尾，我们就可以一个一个地插入数据。虽然最后一个d[9] = 7更新进去对于这组数据没有什么意义，但是如果后面再出现两个数字 8 和 9，那么就可以把8更新到d[5], 9更新到d[6]，得出LIS的长度为6。

然后应该发现一件事情了：在B中插入数据是有序的，而且是进行替换而不需要挪动——也就是说，我们可以使用二分查找，将每一个数字的插入时间优化到O(logN)~~~~~于是算法的时间复杂度就降低到了O(NlogN)~！

【例3】最长上升子序列（二）--1503

【问题2】常规求LIS方法时间复杂度n^2，不够优秀，面对n < = 200000的数据范围，应降低时间复杂度

【优化时间复杂度至nlogn】

定义d[k]：长度为k的上升子序列的最末元素，若有**多个长度为k**的上升子序列，则**记录最小的那个最末元素**。

注意d中元素是单调递增的，下面要用到这个性质。

首先len = 1,d[1] = a[1]，然后对a[i]：若a[i]>d[len]，那么len++，d[len] = a[i];

否则，我们要从d[1]到d[len-1]中找到一个j，满足**d[j-1]<a[i]<d[j]**,则根据D的定义，我们需要更新长度为j的上升子序列的最末元素（使之为最小的）即 d[j] = a[i];

最终答案就是len

利用d的单调性，在查找j的时候可以二分查找，从而时间复杂度为nlogn。

【例3】最长上升子序列（二）--1503

【问题2】常规求LIS方法时间复杂度n^2，不够优秀，面对n < = 200000的数据范围，应降低时间复杂度

```
int ERLIS()
{  int i,L,r,mid,len=1;
   b[1]=a[1];
   for(i=2;i<=n;i++)
   {  L=1;r=len;
      if(b[len]<a[i]){len++;b[len]=a[i];continue;} //产生以a[i]结尾（接在b[len]后）更长的子序列
      while(L<=r)      //在b数组中二分，找到a[i]合适的插入（替换）位置
      {  mid=(L+r)/2;
         if(b[mid]<a[i])L=mid+1;else r=mid-1;
      }
      b[L]=a[i];  //L和r一定交叉  r<L，显然替换后者L的元素b[L]
   }
   return len;
}
```

【练习】CMI（2008省选模拟）--1504

Description

给出一个1到n的排列，每次可以移动一个数到任意一个位置（第一个数前，最后一个数后或者相邻两个数之间的位置）。问要达到状态1，2，3，.....，n至少要移动多少次?

Input

第一行一个正整数n，第二行n个整数，表示一个1到n的排列。

Output

在第一行输出最少的移动次数

Sample Input

5

2 1 4 5 3

Sample Output

2

【练习】CMI（2008省选模拟）--1504

【问题分析】

本题数据范围为：1<=n<=200000

显然，采用NlogN的LIS算法即可。

找出 **最长上升子序列** 后，非子序列中的元素都一个一个移动插入的相应位置即可。

【例4】低价购买 --1505

Description

“低价购买”这条建议是在奶牛股票市场取得成功的一半规则。要想被认为是伟大的投资者，你必须遵循以下的问题建议:“低价购买；再低价购买”。每次你购买一支股票,你必须用低于你上次购买它的价格购买它。买的次数越多越好!你的目标是在遵循以上建议的前提下，求你最多能购买股票的次数。你将被给出一段时间内一支股票每天的出售价(2^16范围内的正整数)，你可以选择在哪些天购买这支股票。每次购买都必须遵循“低价购买；再低价购买”的原则。写一个程序计算最大购买次数。

这里是某支股票的价格清单:

日期 1 2 3 4 5 6 7 8 9 10 11 12

价格 68 69 54 64 68 64 70 67 78 62 98 87

最优秀的投资者可以购买最多4次股票，可行方案中的一种是:

日期 2 5 6 10

价格 69 68 64 62

【例4】低价购买 --1505

Input

第1行: N (1 <= N <= 5000)，股票发行天数

第2行: N个数，是每天的股票价格。

Output

输出文件仅一行包含两个数:最大购买次数和拥有最大购买次数的方案数(<=2^31)当二种方案“看起来一样”时（就是说它们构成的价格队列一样的时候）,这2种方案被认为是相同的。

Sample Input

12

68 69 54 64 68 64 70 67 78 62 98 87

Sample Output

4 2

【例5】过河（NOIP2005提高）--1506

Description

在河上有一座独木桥，一只青蛙想沿着独木桥从河的一侧跳到另一侧。在桥上有一些石子，青蛙很讨厌踩在这些石子上。由于桥的长度和青蛙一次跳过的距离都是正整数，我们可以把独木桥上青蛙可能到达的点看成数轴上的一串整点：0，1，……，L（其中L是桥的长度）。坐标为0的点表示桥的起点，坐标为L的点表示桥的终点。青蛙从桥的起点开始，不停的向终点方向跳跃。一次跳跃的距离是S到T之间的任意正整数（包括S,T）。当青蛙跳到或跳过坐标为L的点时，就算青蛙已经跳出了独木桥。

题目给出独木桥的长度L，青蛙跳跃的距离范围S,T，桥上石子的位置。你的任务是确定青蛙要想过河，最少需要踩到的石子数。

【例5】过河（NOIP2005提高）--1506

Input

输入的第一行有一个正整数L（1 <= L <= 10^9），表示独木桥的长度。第二行有三个正整数S，T，M，分别表示青蛙一次跳跃的最小距离，最大距离，及桥上石子的个数，其中1 <= S <= T <= 10，1 <= M <= 100。第三行有M个不同的正整数分别表示这M个石子在数轴上的位置（数据保证桥的起点和终点处没有石子）。所有相邻的整数之间用一个空格隔开。

Output

输出只包括一个整数，表示青蛙过河最少需要踩到的石子数。

Sample Input

10
2 3 5
2 3 5 6 7

Sample Output

2