本章主要衔接第二章,题目类型不定

Translate:USACO/agrinet

Agri-Net 最短网络

描述

农民约翰被选为他们镇的镇长！他其中一个竞选承诺就是在镇上建立起互联网，并连接到所有的农场。当然，他需要你的帮助。约翰已经给他的农场安排了一条高速的网络线路，他想把这条线路共享给其他农场。为了使花费最少，他想铺设最短的光纤去连接所有的农场。你将得到一份各农场之间连接费用的列表，你必须找出能连接所有农场并所用光纤最短的方案。每两个农场间的距离不会超过 100000

格式

PROGRAM NAME: agrinet

INPUT FORMAT:

(file agrinet.in)

第一行： 农场的个数，N（3<=N<=100）。

第二行..结尾: 后来的行包含了一个 N*N 的矩阵,表示每个农场之间的距离。理论上，他们是 N 行，每行由 N 个用空格分隔的数组成，实际上，他们限制在 80 个字符，因此，某些行会紧接着另一些行。当然，对角线将会是 0，因为不会有线路从第 i 个农场到它本身。

OUTPUT FORMAT:

(file agrinet.out)

只有一个输出，其中包含连接到每个农场的光纤的最小长度。

SAMPLE INPUT

```
4
0 4 9 21
4 0 8 17
9 8 0 16
21 17 16 0
```

SAMPLE OUTPUT

```
28
```

解题报告:MST

Translate:USACO/inflate

Score Inflation 总分

描述

学生在我们 USACO 的竞赛中的得分越多我们越高兴。

我们试着设计我们的竞赛以便人们能尽可能的多得分,这需要你的帮助。

我们可以从几个种类中选取竞赛的题目,这里的一个"种类"是指一个竞赛题目的集合,解决集合中的题目需要相同多的时间并且能得到相同的分数。

你的任务是写一个程序来告诉 USACO 的职员,应该从每一个种类中选取多少题目,使得解决题目的总耗时在竞赛规定的时间里并且总分最大。输入包括竞赛的时间,M(1 <= M <= 10,000)(不要担心,你要到了训练营中才会有长时间的比赛)和 N,"种类"的数目 1 <= N <= 10,000。后面的每一行将包括两个整数来描述一个"种类":

第一个整数说明解决这种题目能得的分数(1 <= points <= 10000),第二整数说明解决这种题目所需的时间(1 <= minutes <= 10000)。你的程序应该确定我们应该从每个"种类"中选多少道题目使得能在竞赛的时间中得到最大的分数。

来自任意的"种类"的题目数目可能任何非负数(0 或更多)。

计算可能得到的最大分数。

格式

PROGRAM NAME: inflate

INPUT FORMAT:

(file inflate.in)

第 1 行: M, N--竞赛的时间和题目"种类"的数目。

第 2-N+1 行: 两个整数:每个"种类"题目的分数和耗时。

OUTPUT FORMAT:

(file inflate.out)

单独的一行包括那个在给定的限制里可能得到的最大的分数。

SAMPLE INPUT

```
300 4
100 60
250 120
120 100
35 20
```

SAMPLE OUTPUT

```
605
```

{从第 2 个"种类"中选两题第 4 个"种类"中选三题}

解题报告:无限背包

Translate:USACO/humble

Humble Numbers 丑数

描述

对于一给定的素数集合 S = {p1, p2, ..., pK},考虑一个正整数集合，该集合中任一元素的质因数全部属于 S。这个正整数集合包括，p1、p1*p2、p1*p1、p1*p2*p3...(还有其它)。该集合被称为 S 集合的“丑数集合”。

注意：我们不认为 1 是一个丑数。

你的工作是对于输入的集合 S 去寻找“丑数集合”中的第 N 个“丑数”。所有答案可以用 longint(signed 32-bit)存储。

格式

PROGRAM NAME: humble

INPUT FORMAT:

(file humble.in)

第 1 行: 二个被空格分开的整数:K 和 N ， 1<= K<=100 ， 1<= N<=100,000.

第 2 行: K 个被空格分开的整数:集合 S 的元素

OUTPUT FORMAT:

(file humble.out)

单独的一行，输出对于输入的 S 的第 N 个丑数。

SAMPLE INPUT

```
4 19
2 3 5 7
```

SAMPLE OUTPUT

```
27
```

分析

官方题解 by Russ Cox

译 by 逆铭

我们在数组 hum 中计算出前 n 个丑数。为了实现起来更简单，我们把 1 也作为一个丑数，算法也要因此略微调整一下。

当我们已知前 k 个丑数，想得到第 k+1 个，我们可以这样做：

```
对于每个质数 p
        寻找最小的丑数 h
          使得 h * p 比上一个丑数大

取我们找到的 h * p 中最小的一个：它就是下一个丑数
```

为了使搜索更快，我们可以为每个质数维护一个索引“pindex”表示每个质数已经乘到了哪个丑数，每次都从那里开始，而不是再从头再来。

方法二：

这道题可以用 BFS+Treap 来做。但这里的 BFS 不使用队列来扩展，而是用 Treap 来扩展。

建一个 Treap 保存已经得到的数，从小到大每次从堆中取出一个数，用它和集合中的质数相乘，查找判断它是否重复.如果没有重复，那么将它插入到 Treap 中。直到产生了 n 个数，那么再往后扩展一位，得到的第 n 个数既为所求的结果。时间复杂度：BFS 扩展为 O(N), 查第 k 大为 O(logN)判重为 O(logN),插入为 O(logN),因此总的复杂度为 O(NlogN)

方法三：

这道题目可以直接应用 STL 中的 priority_queue 来做，复杂度为 O(NKlogN),对于第 i 小的数，扩充它能得到的所有新元素并插入 priority_queue 中，然后 pop 第 i 小的数。时间没问

题，但是空间会爆，当然只对于最后一个数据，我最后一个数据在自己机子上 RUN 完后，骗过去的，USACO 上分配内存太小!

Translate:USACO/rect1

Shaping Regions 形成的区域

描述

N 个不同的颜色的不透明的长方形(1 <= N <= 1000)被放置在一张宽为 A 长为 B 的白纸上。这些长方形被放置时，保证了它们的边与白纸的边缘平行。所有的长方形都放置在白纸内，所以我们会看到不同形状的各种颜色。坐标系统的原点(0,0)设在这张白纸的左下角，而坐标轴则平行于边缘。

格式

PROGRAM NAME: rect1

INPUT FORMAT:

(file rect1.in)

按顺序输入放置长方形的方法。第一行输入的是那个放在底的长方形(即白纸)。

第 1 行: A ， B 和 N, 由空格分开 (1 <=A, B<=10,000)

第 2 到 N+1 行: 为五个整数 llx, lly, urx, ury, color 这是一个长方形的左下角坐标，右上角坐标和颜色。

颜色 1 和底部白纸的颜色相同。 (1 <= color <= 2500)

OUTPUT FORMAT：

(file rect1.out)

输出文件应该包含一个所有能被看到颜色连同该颜色的总面积的清单(即使颜色的区域不是连续的)，按 color 增序排列。

不要显示没有出现过的颜色。

SAMPLE INPUT

```
20 20 3
2 2 18 18 2
0 8 19 19 3
8 0 10 19 4
```

SAMPLE OUTPUT

```
1 91
2 84
3 187
4 38
```

INPUT EXPLANATION

请注意：被(0,0)和(2,2)所描绘的是 2 个单位宽、2 个单位高的区域

这里有一个示意图输入：

```
11111111111111111111
33333334433333333331
33333334433333333331
33333334433333333331
33333334433333333331
33333334433333333331
33333334433333333331
33333334433333333331
33333334433333333331
33333334433333333331
33333334433333333331
33333334433333333331
11222222442222222211
```

```
11222222442222222211
11222222442222222211
11222222442222222211
11222222442222222211
11222222442222222211
11111111441111111111
11111111441111111111
```

'4'在(8,0)与(10,19)形成的是宽为 2 的区域,而不是 3.（也就是说，4 形成的区域包含(8,0)和(8,1) ，而不是(8,0)和(8,2)） 。

分析

最简单的方法是灌水法了，虽然很可能行不通的，但大多数朋友还是会去试一试的，因为这是第一直觉，也是最容易实现的。

灌水法

```
fillchar (map,sizeof(map),1);
    for l:=1 to n do →放置每一个矩形
      for i:=x1[l] to x2[l]-1 do
        for j:=y1[l] to y2[l] do
          map[i,j]:=color[l];
```

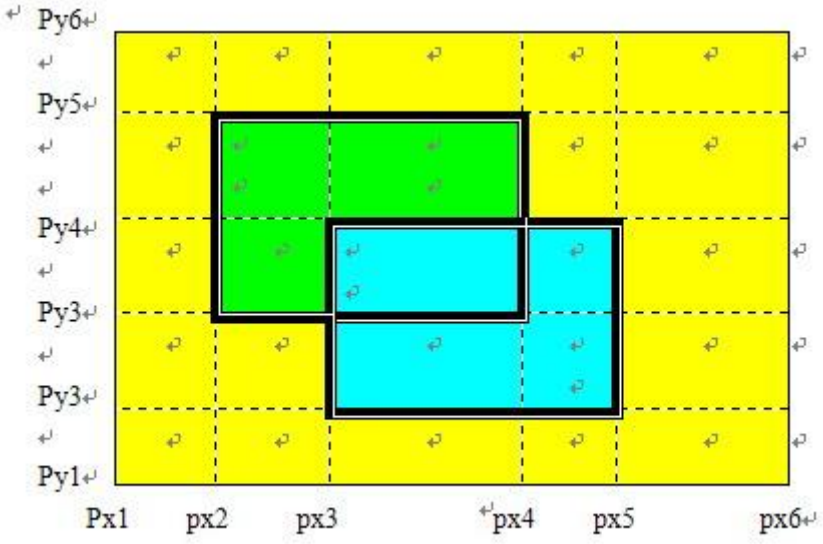
最后对 map 整体扫描一遍便可得到各种颜色的面积。很可惜的是空间运行是远远不够的，只能换个角度了…… 基于对各个格子的考虑。也可以容易的写出下面的代码：

```
for i:=1 to a do
      for j:=1 to b do
        for l:=n downto 0 do  →{第 0 个矩形为矩形（a,b）,color[0]=1}
          if (x1[l]<i) and (x2[l]>=i) and (y1[l]<j) and (y2[l]>=j)
            then begin
                   inc(col[color[l]]);
                   break;
                 end.
```

两个算法只是角度不一样，复杂度都是 O（nab）。但后一个空间比前一个小的多，且速度快的多（多了个 break）。这个算法可以过 10 个数据，一共 11 个数据（usaco 老是出一些 bt 的数据）. O(nab)的复杂度，对付 n=1000，a=b=10000 的数据显然是行不通的。第十一个数据就是这样的一个数据。

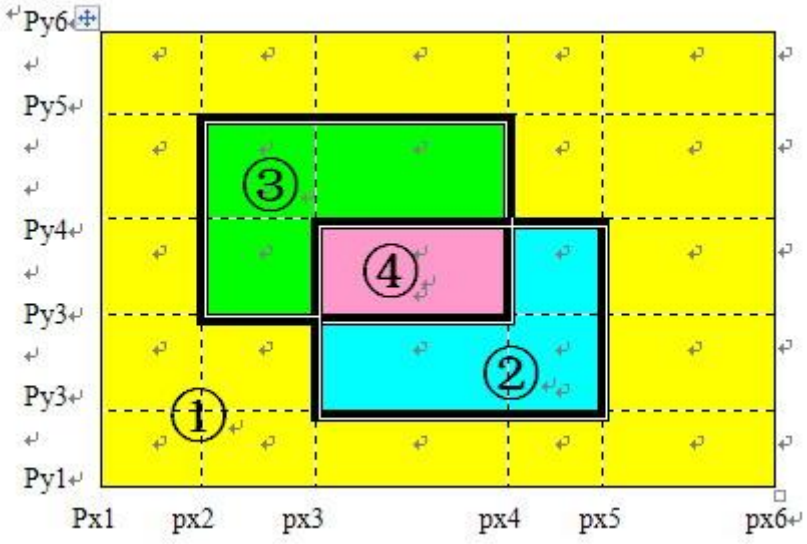
离散化

上面的算法到底是慢在那里呢？基于上述算法，我们对每一个 1*1 的方格都处理了一次。很显然这是不必要的。更多的方格可以连在一起处理。所以也就很自然的想到了离散化：两个数组分别记录放置矩形的横坐标和纵坐标。



很容易发现这些离散化后的小矩形中的各个方格的颜色是相同的。在每一个离散矩形里选出一 1*1 的方格作为代表，计算它的颜色。这样就可以在一次中处理一个离散矩形里的颜色属性。离散化显然是比原来的算法更进一步，离散后复杂度降到 O（4n^3）。对付 n=1000 的数据还是弱了点（还是只能过 10 数据）。

仔细想一想。离散化后还是做了很多的不必要的工作。



如图，用前面的算法，要确定 5*5=25 个矩形的颜色。而实际上只有 3 重颜色，4 个区域（实线围成），按照区域算，也只要确定 4 个区域的颜色属性，而不是 25 个。算法之所以慢的原因也就找到了：做了大量的不必要的工作。现在我们要做的是合并这些相同属性的格子，做最精简的工作。怎么有效的去合并这些相同属性的子问题，是现在面临的首要问题。在原

来算法的基础上，很容易想到的是用‘种子填充法’。我们以深搜时搜到某个区域的第一个格子为代表，记录该区域的面积。本题不仅在时间上很苛刻，在空间上的障碍也是制约算法实现的难点，所以想出的算法必须考虑算法的有效性。在 dfs 前预处理用一 boolean 数组记录相邻格子之间的连通情况（虚线为连通，实线为不连通）。基于空间上的开销太大，还是要采用离散思想，并对每个矩形进行压缩。离散后的空间开销只有 8^6 的 boolean 数组，压缩矩形后处理边的连用情况也快的多了。

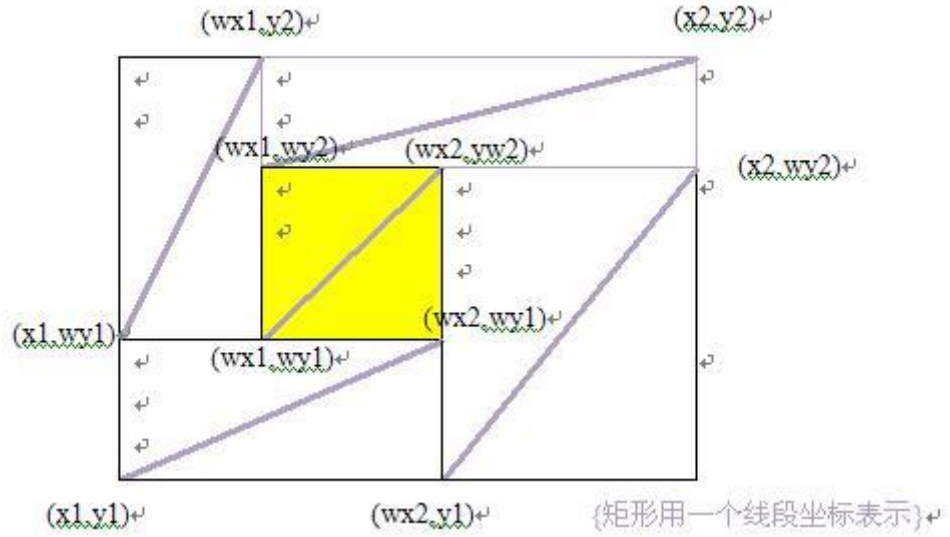
当然也不一定要像上面的算法那样‘把能合并的都合并了’，其实只要不去做太多无谓的工作，一样可以得到很不错的效果。基于此思想的‘矩形切割’和‘线段树’就是很不错的算法。

离散化+倒序染色

现在我们从另一个角度思考“如何减少不必要的工作”。采用离散化+暴力染色法，最坏的情况是 O(n^3)的（即最后一个数据），TLE。我们 TLE 的原因除了把矩形分割的过细，还对同一个格子重复染色多次，如何能做到对一个格子只染色一次呢？倒序染色+封闭染过色的格子！具体可以对平面的每一行开一个链表，存储这一行中的格子，染色后将这个格子删去（当然，你得用一定的手段使得最后我们还能查到这个格子染的颜色，推荐数组模拟链表）。P.S：最后一个数据大概在 1.2S 左右,前面都是秒出.其实可以不用记录格子的颜色,在每次染色的过程中就可以顺便统计一下。不过这个方法会超时，实现的时候需要倒序，对于已经染过色的需要从队列里删除。

矩形切割

此思想把矩形（a,b）看成是最先加入的矩形，颜色为 1。然后每次依次加入矩形，将与其重叠的矩形分割后加入队列之后（去掉重叠的部分）。最后扫描一次队列即可得出各个颜色的面积。具体分割方法如下：



黄色区域是黄色矩形和白色矩形的重叠部分:

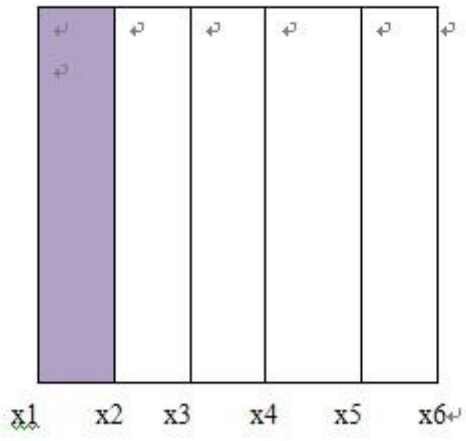
```
wx1=max(x1,px1);
wx2=min(x2,px2);
wy1=max(y1,py1);
wy2=min(y2,py2);
```

以上就是重叠部分的坐标。而原来的白色矩形被除去黄色区域以外，被分割成了四个小矩形。上图只是一般情况，更多的被分割出的矩形面积为 0。

线段树

上面所提到的算法都是动态处理的，还可以写出基于线段树的静态处理的方法。如果采用二维线段树（每次最多有 4 个下降分支），复杂度为 O(4nlogn)。由于空间的限制，必须进行压缩，这一点还不够，还要使用动态线段树处理才能突破空间这个头痛的问题。实现起来太繁琐。

为了解决空间这个难题，我们不得不牺牲时间换空间，我们只对 x 轴进行离散化，对每一列用一维线段树处理。



插入一线段：

```
procedure insert(u:integer);
    begin
      if treec[u]>0 then exit;
      if (treel[u]>=y1[k]) and (treer[u]<=y2[k]) and (treec[u]=0)
        then begin treec[u]:=col[k]; exit; end;
      treec[u]:=-1;
      if (y1[k]<treer[2*u]) then insert(2*u);
      if (y2[k]>treel[2*u+1]) then insert(2*u+1);
    end;
```

统计各种颜色覆盖面积：

```
procedure total(u:integer);
    begin
      if treec[u]=0 then exit;
      if treec[u]<>-1
        then inc(ff[treec[u]],(treer[u]-treel[u])*(pt[w+1]-pt[w]))
        else begin total(2*u); total(2*u+1); end;
      treec[u]:=0;
    end;
```

Translate:USACO/contact

Contact 联系

IOI'98

奶牛们开始对用射电望远镜扫描牧场外的宇宙感兴趣。最近，他们注意到了一种非常奇怪的脉冲调制微波从星系的中央发射出来。他们希望知道电波是否是被某些地外生命发射出来的，还是仅仅是普通的的星星发出的。

帮助奶牛们用一个能够分析他们在文件中记下的记录的工具来找到真相。他们在寻找长度在 A 到 B 之间（含）在每天的数据文件中重复得最多的比特序列 (1 <= A <= B <= 12)。他们在找那些重复得最多的比特序列。一个输入限制告诉你应输出多少频率最多的序列。

符合的序列可能会重叠，并且至少重复一次的序列会被计数。

格式

PROGRAM NAME: contact

INPUT FORMAT:

(file contact.in)

第一行：三个用空格分隔的整数: A, B, N; (1 <= N < 50)

第二行及以后: 一个最多 200，000 字符的序列，全是 0 或 1; 每行有 80 个字符，除了可能的最后一行。

OUTPUT FORMAT:

(file contact.out)

输出 N 个频率最高的序列（按照频率由高到低的次序）。由短到长排列频率相同的这些序列，如果长短相同，按二进制大小排列。如果出现的序列个数小于 N，输出存在的序列。

对于每个存在的频率，先输出单独包含该频率的一行，再输出以空格分隔的这些频率。每行六个（除非少于六个剩下）。

SAMPLE INPUT

```
2 4 10
0101001001000100011110110000101001100111100001001001111001000000
```

在样例里，序列 100 出现了 12 次，而序列 1000 出现了 5 次。次数最多的序列是 00，出现了 23 次。

SAMPLE OUTPUT

```
23
00
15
01 10
12
100
11
11 000 001
10
010
8
0100
```

```
7
0010 1001
6
111 0000
5
011 110 1000
4
0001 0011 1100
```

分析

1 <= A <= B <= 12，由于信号长度不同需要区别（0 与 00 不是同一信号），可以转化为三进制储存，也可以用二维数组 F[i,j]表示值为 i 位数为 j 的信号出现次数。最后看储存方法不同按频率由高到低，由短到长，由小到大输出。

也可直接转成二进制处理,只需要每种字符串加一个前导'1'。

直接暴力枚举！！！！简单 粗暴

TRIE

这道题可以用 TRIE 解决。。

AC 自动机

建立一棵深度为 B 的满而叉 trie 树，然后对于所有深度大于等于 A 的节点标记一个单词。对文本进行匹配，排序输出。

Translate:USACO/stamps

Stamps 邮票

已知一个 N 枚邮票的面值集合（如，{1 分，3 分}）和一个上限 K —— 表示信封上能够贴 K 张邮票。计算从 1 到 M 的最大连续可贴出的邮资。

例如，假设有 1 分和 3 分的邮票；你最多可以贴 5 张邮票。很容易贴出 1 到 5 分的邮资（用 1 分邮票贴就行了），接下来的邮资也不难：

```
6 = 3 + 3
```

```
7 = 3 + 3 + 1
8 = 3 + 3 + 1 + 1
9 = 3 + 3 + 3
10 = 3 + 3 + 3 + 1
11 = 3 + 3 + 3 + 1 + 1
12 = 3 + 3 + 3 + 3
13 = 3 + 3 + 3 + 3 + 1。
```

然而，使用 5 枚 1 分或者 3 分的邮票根本不可能贴出 14 分的邮资。因此，对于这两种邮票的集合和上限 K=5，答案是 M=13。

[规模最大的一个点的时限是 3s]

格式

PROGRAM NAME: stamps

INPUT FORMAT:

(file stamps.in)

第 1 行： 两个整数，K 和 N。K（1 <= K <= 200）是可用的邮票总数。N（1 <= N <= 50）是邮票面值的数量。

第 2 行 .. 文件末：N 个整数，每行 15 个，列出所有的 N 个邮票的面值，每张邮票的面值不超过 10000。

OUTPUT FORMAT:

(file stamps.out)

第 1 行：一个整数，从 1 分开始连续的可用集合中不多于 K 张邮票贴出的邮资数。

SAMPLE INPUT

```
5 2
```

```
1 3
```

SAMPLE OUTPUT

```
13
```

官方题解(by Alex Schwendner)

一个简单的 dp 题目.我们开设数组 minstamps.minstamps[i]表示凑成 i 分游资所需要的最少邮票数.对于每一种邮票,我们试图在每一分我们已经贴出的邮资上再贴一张这种邮票.在我们找到贴出贴出任何钱数所需要的最少邮票数,我们就寻找不能用所给的邮票数所能贴出的最少邮资.输出比它小 1 的数.