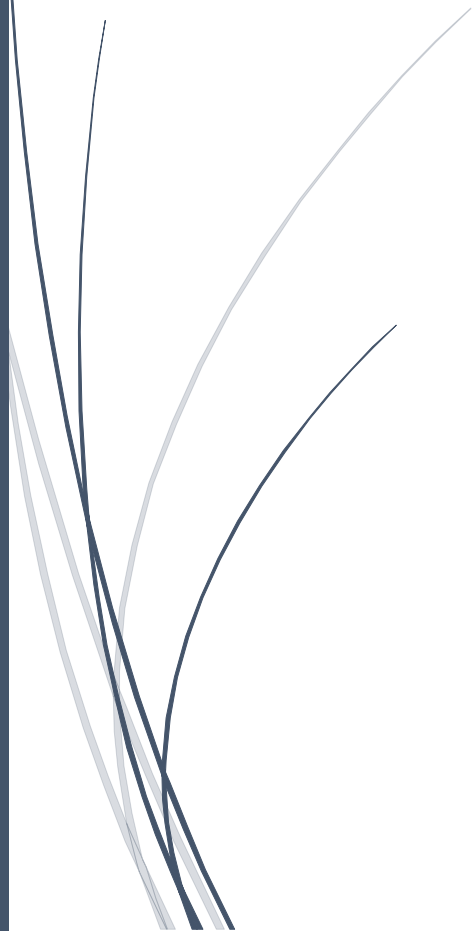
2015-4-18

The 2015 ACM-ICPC

School Contest of Chang'an University

Solutions



By:XorZip

热身赛

A. Magic Prime Number

J_cf

简单的素数判断，数据很弱，试除法都能过，判定一下是否是素数，不是的话一大一小两个方向找最近的即可。

要注意的是 0 和 1 这种边界很容易忽略，尤其是 0，这个是在我打印试题之前突然想到要加上的，果然坑到一群人。

B. Students

J_cf

题意是广场上总共有很多人，其中大于 P%小于 Q%的人是男生。其实当时出这个的时候还是挺担心有人不明白就这点条件能用来干什么？不过想想反正放在热身赛里面也就可以乱搞了。

简单地说就是总人数是一个正整数，男生的人数也是一个正整数。所以找到一个最小的正整数乘上 P%和 Q%之后，夹在中间的能有个正整数即可。数据同样很弱，从 1 开始往上枚举即可。

数据里面有几组出现了 N 乘上 P%或 Q%刚好能得到正整数的情况，但是题目要求必须是大于 P%和小于 Q%，不注意容易坑死在这里，注意了也容易写错坑死在这里。

正式赛

A. YY's string

L_w

这道题目是用来增加 FB 氛围的题目，唯一的亮（坑）点就是题目很长很长，长到我自己都想骂人了，后来发现太长删掉了两大段，不至于赛后被人狂喷不止。

具体做法，以一个标记来记录是否检测到句号。然后进行相应的处理即可。

B. YY's problem

L_w

为了能送出更多的气球，我只选择了一道简单的题目了，此题的正解是裸的矩阵快速幂。

$$f_i = f_{i-1} + f_{i+1} \text{ 移项可得 } f_i = f_{i-1} - f_{i-2}$$

可以得到一个 2*2 的矩阵$\begin{pmatrix} 0 & 1 \\ -1 & 1 \end{pmatrix}$，即$\begin{pmatrix} 0 & 1 \\ -1 & 1 \end{pmatrix}\begin{pmatrix} f_i \\ f_{i+1} \end{pmatrix} = \begin{pmatrix} f_{i+1} \\ f_{i+1} - f_i \end{pmatrix} = \begin{pmatrix} f_{i+1} \\ f_{i+2} \end{pmatrix}$ 然后通过快速幂来解决这个问题了。

当然更简单的解法是，推一下可以发现 fi 会形成一个循环结……

C. YY's sequence

L_w

题目大意是求第 k 个无平方因子数。

无平方因子数(Square-Free Number)，即分解之后所有质因数的次数都为 1 的数。我们考虑二分答案，将问题转化为[1,x]区间中有多少个无平方因子数。

根据**容斥原理**：

对于$\sqrt{x}$内的所有素数，满足

$[1,x]$中的无平方因子数的个数

=

0 个素数乘积的平方的倍数的数的个数 (1 的倍数，也就是所有数)

减去 1 个素数的平方的倍数的数的个数$(4$ 的倍数$,9$ 的倍数$,\dots)$

加上 2 个素数乘积的平方倍数的数的个数$(36$ 的倍数$,100$ 的倍数 $\dots)$

……

每个乘积 a 的前面的符号，恰好是 $\mu(a)$

所以可以直接预处理莫比乌斯函数来用

而x以内i^2的个数有$\left\lfloor\frac{x}{i^2}\right\rfloor$个

所以$[1,x]$中的无平方因子数的个数为

$$Q(x)=\sum_{i=1}^{\lfloor\sqrt{x}\rfloor}\mu(i)\left\lfloor\frac{x}{i^2}\right\rfloor$$

D. BlackBox

J_cf

题目要求维护一个数据结构，一共两种操作：插入和询问第 k 小。第 k 小这个 k 是从 1 开始每询问完一次往上递增 1 的。

动态维护第 k 值这种千古经典问题一向有很多解决办法：线段树、树状数组、平衡树等等均可解，还有个队用了类快排的结构直接在数组中求的第 k 值。对于以上方法，插入和第 k 值都是比较裸的，就没有什么好说的了。出题时候的原意是想让本校同学用堆解决，可惜最后没有出。

由于堆顶元素相对于整个堆中元素的单调性，对于 k 值固定的第 k 小，只要一个容量为 k 的大根堆即可。保证堆中的元素在所有数据中是最小的，那么大根堆中相对于整个堆最大的堆顶元素就是所有数中第 k 小的数。

虽然这里这个 k 是在变的，不是一个固定值，但是幸运的是 k 只是简单的递增。于是可以考虑另外开一个小根堆，把大根堆中多余的数存到小根堆里去，需要 k 增加时，再把小根堆的堆顶弹回大根堆，完成 k++。

则操作是这样的：

插入：先将数据插入大根堆，之后若大根堆的容量大于 k 则弹出堆顶元素，将其插入小根堆。

询问：直接输出大根堆堆顶元素，然后从小根堆中弹出堆顶元素，插入大根堆。

E. Hack

J_cf

求一些数据之中，最多删去 k 组之后的最大加权平均数。首先是一个贪心点，题中说最多删去 k 门课的成绩，但其实加权平均值是数据越多越平均的，就是说为了尽量拉高平均值，删去的课程要尽量多，即删去 k 门课的时候才能达到最大。

然后下面的求解，原来的想法是数据要把贪心卡掉，但是当时出数据的时候忘了在 pc2 上实测了，然后就没卡成……0.0

为了符合题目背景（Orz，下次不这么干了），学分只有 10 个，所以把相同学分的成绩排序，每次取一个成绩时，用每种学分中最大的成绩计算，若取了这个成绩对结果会有多大的影响，把影响最大的那组成绩/学分加入结果。这样每次扫描一遍，由于需要处理的只有 10 组，复杂度高不到哪去，最后的结果也是正确的。再次 Orz，给各位巨巨们跪了，当时出题的时候完全没想过 credit[i] 比较小的时候还能这么解。

正解如下，对于本题，可以抽象成这样：

$$R=\frac{\sum_{i=1}^{n}x[i]*\mathrm{c}[i]*\mathrm{s[i]}}{\sum_{i=1}^{n}x[i]*\mathrm{c}[i]},\sum_{i=1}^{n}x[i]=n-k$$

其中 x[i] 为 0 或者 1，表示取或者不取。然后目标是 R 的最大值。

对上式移项推导如下：

$$R*\sum_{i=1}^{n}x[i]*\mathrm{c}[i]=\sum_{i=1}^{n}x[i]*\mathrm{c}[i]*\mathrm{s[i]}$$

$$R*\sum_{i=1}^{n}x[i]*\mathrm{c}[i]-\sum_{i=1}^{n}x[i]*\mathrm{c}[i]*\mathrm{s[i]}=0$$

$$\sum_{i=1}^{n}x[i]*\left(R*\mathrm{c}[i]-\mathrm{c[i]}*\mathrm{s[i]}\right)=0$$

若令：

$$F(\mathrm{R})=\sum_{i=1}^{n}x[i]*\left(R*\mathrm{c}[i]-\mathrm{c[i]}*\mathrm{s[i]}\right)$$

则对于某一特定的 x[i]序列，能够使得 F(R)等于 0 的 R 就是此时的解。

那么再看，整个$\left(R*\mathrm{c}[i]-\mathrm{c[i]}*\mathrm{s[i]}\right)$ 是对于 R 单调递增的，如果取定一个 R，似乎总能通过调整 x[i] 序列（也就是相当于从原来 n 个数中选取 n-k 个数）来使得 F(R) 等于 0。

然后我们的目标是使得 R 最大，那么对于一个最大的 R 来说，F(R)取到的数应该是所有的$\left(R*\mathrm{c}[i]-\mathrm{c[i]}*\mathrm{s[i]}\right)$里面最小的，即当取到最大的 R 时，所有$\left(R*\mathrm{c}[i]-\mathrm{c[i]}*\mathrm{s[i]}\right)$数中最小的 n-k 个相加的和为零。因为无法直接求解这个 R，所以想到使用二分的方法将 F(R)逼近到 0。

因此最终的解法是，二分答案 R，对于每次的 R，计算所有$\left(R*\mathrm{c}[i]-\mathrm{c}[\mathrm{i}]*\mathrm{s}[\mathrm{i}]\right)$之后对其进行排序，取其中最小的 n-k 个值，判断是否为 0，然后根据判断结果调整二分区间，将它们的总和逼近到 0。当和与 0 的误差小到一定程度之后即可结束，此时的 R 就是要求的最大值。

F. StarCross

J_cf

这道题真的只是稍微麻烦了一点点而已……

题目整体是一个有向图的最短路，走过每条有向边需要消耗 1 单位的时间和一定量的能量。额外的条件是黑洞和白洞的设定。从黑洞走到白洞耗能增加，从白洞走到黑洞耗能减少，从黑洞走到黑洞或者白洞走到白洞耗能不影响。然后题中时间的设定是为了满足黑洞白洞每单位时间转化一次。可以选择在某个点上停留，如果是白洞无影响，如果是黑洞则需要消耗一定量的能量。

看上去比较麻烦（确实很麻烦），其实只要附加一下额外的条件即可：图是每单位时间变化一次，而每次是所有黑白洞都翻转一次，因此从整体上来看，这就相当于是一个分层图。可将原图中的点集拆为颜色相反两部分，分别代表奇时刻和偶时刻，不同时刻之间再相互连边，最后使用任何一种最短路算法都可解决。

当然采用分层图之后，点数翻倍，边数翻倍之后还要加上点与点之间的双向边，复杂度的量级没有变，但是还是翻了一倍，加上各种连边处理啥的，代码写起来还是比较复杂的，因此也不是我想要的正解。

由于题目中，整个图的结构并不会随着时间发生改变，因此若采用 SPFA，则只需要在一般一维的 dist[] 中加上一个奇偶时间维即可，修改一下 BFS 中的动作，用 t[]记录当前状态的时间，d[]表示点的原始颜色状态，则 (t[]+d[]) mod 2 可以很方便地计算出某个点在某时间的黑白状况。整体在普通的 SPFA 上稍加改造即可解决。

G. Partner

J_cf

非常基础的一道并查集，数据也比较裸。并查集处理一下两个班中每个人的分组情况，然后扫描一遍记录下每组的男女生人数，最后把两个班的情况排序之后进行依次对比即可，若有一个不对的即可跳出 Unhappy。因为是两个班，写好一个之后代码复制一遍改改就好。

DFS 找连通块亦可。

H. Dance Party

S_yh

这道题可以有很多种做法，但是最简单的还是贪心。对男生女生的技能值从小到大排序，然后符合条件就结为一组。

I. Paint Round Holes

S_yh

改编自紫书 P336 例题。

给出一个闭区间[a, b]，求这个区间里有多少个 0,6,9，另外 8 要计数两次。

设 f(n)表示 0~n 中所要统计的“圆洞”的个数，则最终答案为 f(b) - f(a-1)。以统计区间[0, 3456]为例，我们可以将区间划分为：

1≤n≤9 *代表 0~9 任意数字
, n, n**　　//这是在统计一位数到三位数的情况
1***, 2***
31**, 32**, 33**
341*, 342*, 343*, 344*
3450, 3451, 3452, 3453, 3454, 3455, 3456

以上只是思路，落实到代码上还有很多细节要考虑。

另外，在赛场上 AC 的代码中还发现一种统计方法：
可以统计每一位上 0~9 分别出现的次数，也许用这种方法编写程序更简单一些。

J. Palindrome Numbers

S_yh

大白书 P171 原题。

好吧，出题人就是太懒了，连改编都懒得改编一下就搬上来了。和上题一样，需要考虑的细节比较多，下面解法并不一定是最好的，仅仅提供一种思路。

可以先找找规律，一位和两位的回文数有 9 个，三位和四位的回文数有 90 个，后面就是 900 个，等等。所以根据 n，我们可以确定下来这是一个多少位的回文数。

五位回文数形如 n***n，六位回文数形如 n****n(n 和*的含义参见上题题解)。因为只要确定了五位数的 n**，根据对称就能确定一个五位回文数。同样，确定了 n**，根据对称能确定一个六位回文数。我们把这样的*叫做自由元，因为五位和六位数都有两个自由元。

所以可以根据输入的 n 来确定所求数字的位数，以及最高位数字。比如，我们确定下来这是一个 7 位最高位是 3 的回文数，那么它有三个自由元 3***(我们只关心这个数的前半部分)。比如，还可以计算出，它是最高位为 3 的第 46 个 7 位回文数。那么我们往三个自由元中填入 46-1=045(不足的用前导 0 补足)，就得到所求为 3045403.