2013 Multi-University Training Contest 8

Solution

UESTC ACM-ICPC Team

Summary

1006 String

这题是鑫航学姐出的。题意很简单，应该是这场最水的题目。

1003 Mine

一道 SG 函数的水题，也可不用 SG。读懂题应该就很好做了。由于 team034 太快过了 1004，导致这题早期没人关注。

具体代码实现的时候需要注意细节。

1008 The Sad Triangles

大概是几何题，正解枚举。

1004 Terrorist's Destroy

比较烦的树形 DP，整体还是较简单的。也许也可以分治过。

1010 Prince and Princess

这题比较容易想简单。许多队伍对复杂度的估计有一些问题。

此题时限放的很松，或许暴力点也能过。

1005 About Set

启发式合并与平衡树。一点点数学知识。

非常抱歉，这道题目审查上的不严格导致题目输出格式与数据输出格式不同。

1002 Connected Components

这题原题是出成一棵树的，但是和之前一次多校题撞了，于是只好改成一个图。

数据由于是随机的，标程以外的做法也是存在的。

1001 Sum of Gcd

数学加数据结构，数据范围比较容易让人想去暴力，虽然肯定会超时。

“宁波镇海的代码我本机跑不到标程两倍他们怎么 T 了”——出题人

（实测大概 1.5 倍左右）

1007 The Happy Triangles

结论很优美的数学题。

精度卡的稍紧，但是 team034 挂在一个很不应该的地方。

```
du s = (a[i]+b[i]+c[i])/2;
```

`a[i],b[i],c[i]` 均为整数。

1009 The Budget of Traveler

树形 DP 加斜率优化。（斜率用 double 判断可能出现精度误差，判断方法参见标程）

1001 Sum of Gcd

首先 $gcd(a[i], a[j]) = d$ 的倍数的组数是 $\binom{num[d]}{2}$，$num[d]$ 表示 $a[i]$ 中有几个数是 d 的倍数。

那么答案就是 $\sum \binom{num[d]}{2} \times f(d)$，其中 $f(d)$ 是容斥因子，必须满足 $\sum_{d|n} f(d) = n$，可以看出 $f(d) = \phi(d)$（如果看不出来也可以用莫比乌斯反演求出，或者直接暴力求也行 $==$）

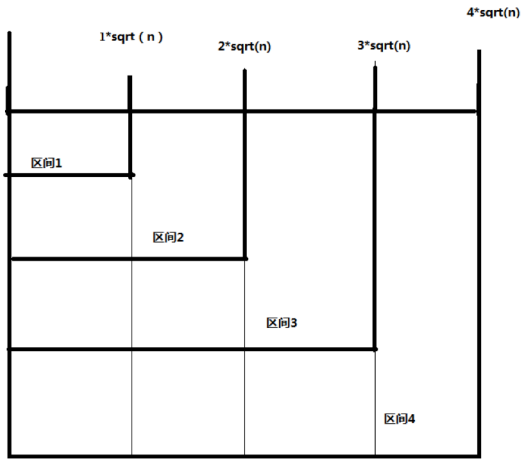
那么我们只要维护 $num[d]$ 就行，我们如果直接暴力的话复杂度是 $\sum (|L[i] - L[i-1]| + |R[i] - R[i-1]|)$。

而这可以通过离线，把询问区间通过排序得到 $n \times \sqrt{n}$ 的复杂度（具体怎么排可以看代码很好懂 $==$）, 也可以通过平面图曼哈顿最小生成树实现 (比较烦不过估计会快一点)。

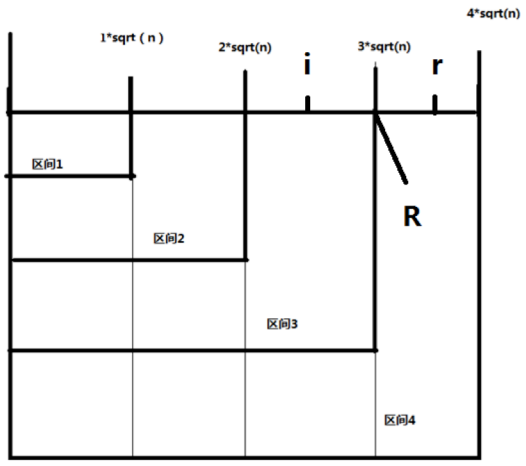
最后总复杂度大概是 $O(n \times \sqrt{n} \times \log n)$，$\log n$ 是约数的原因。

1002 Connected Components

这道题我们可以用分块的思想加并查集解决。
现在将 $1 \to n$ 个点映射在数轴上。
然后将数轴按如下方式分成 $\sqrt{n}$ 块，并且得到如下的 $\sqrt{n}$ 个区间。



现在我们将询问离线，从 n 到 1 倒着来进行处理。
同时对每个区间维护一个并查集。
新进来一个点 i，对于 i，假如某个已经加进来了的点 j（即 $j > i$），与 i 在原图中相连，那么我们就对 i, j 建一条边，并在所有包含这两个点的区间中，将 i, j 相并。
处理完后，我们就可以处理与 i 相关的查询了。
对于一个查询 $l = i$，$r = r$



我们可以发现，r 之前的那个区间（设这个区间的右边界为 R）的连接关系已经确定，连通块数也可以很容易的维护出来。

而 $R+1$ 到 r 最多也只有 $\sqrt{n}$ 个点，所以我们只用对 $R+1$ 到 r 这些点相关的边进行枚举，暴力合并就好。合并过程也要维护一个并查集。

这样就可以根据并查集的性质，得到连通块的数量了。

由于数据的随机性，所以平均每个点有 $\frac{m}{n}$ 的条边。

复杂度是，所有边都会在 $\sqrt{n}$ 个区间中被遍历一次，所以有 $m \times \sqrt{n}$ 的复杂度，每个询问都可能会涉及到需要额外处理的 $\sqrt{n}$ 个点，即 $\frac{\sqrt{n} \times m}{n}$ 条边。

所以总体复杂是 $O(m \times \sqrt{n} + \frac{q \times \sqrt{n} \times m}{n})$ 的。

1003 Mine

把点开空地时会打开的一大片区域看成一块，题目中说到，在一盘游戏中，一个格子不可能被翻开两次，说明任意两块空地不会包含相同的格子。那么就可以看成一个组合游戏。

当空地旁边没连任何数字的时候，$sg = 1$(直接转移到 0)。如果有一个数字，点空地可以转移到 0，点数字可以转移到 1，所以 $sg = 2$。有 2 个数字点空地转移到 0，点数字转移到 2，所以 $sg = 1$。

以此类推，空地旁边有奇数个数字的时候，$sg = 2$，否则 $sg = 1$。

剩下的没与空地相连的数字，每个的 sg 都是 1。

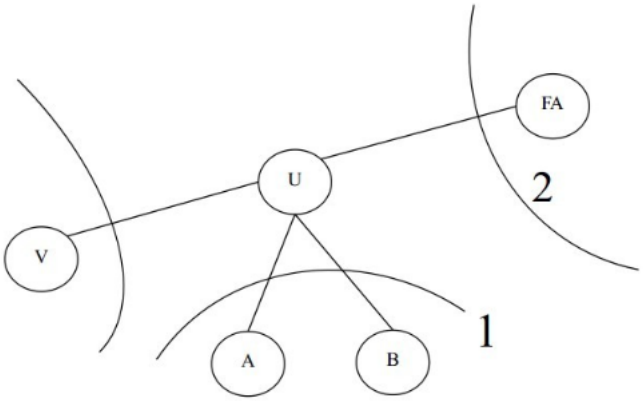
那么将所有空地的 sg 异或起来，再异或 (不与空地相连的数字个数对 2 取模)，等于零输出后手赢，大于 0 输出先手赢即可。

1004 Terrorist's Destroy

可以枚举删掉的边，通过记录一些值，dp 求出删掉此边后剩下两棵树的直径。

首先一遍 dfs 求出以 U 为根的子树直径 $f[U]$，以及子树中距离 U 最长的点和 U 的距离，和它是由 U 的哪个儿子得到; 同时记录一个次大的，再记录一个次次大的。

然后第二遍 dfs，求出除去以 U 为根的子树后，剩下的子树的直径 $h[U]$。



如上图：假设已经得到 $h[U]$ 了 $(h[ROOT] = 0)$，现在要求出除去以 V 为根的子树后，剩下树的直径 $h[V]$，那么 $h[V]$ 可能由四个方面得到：

1. 2 中 $h[U]$。
2. 1 中 U 的儿子的 f 最大值。
3. 1 中从 U 的儿子出发的一条最长的链 $+2$ 中从 FA 出发的一条最长链 $+2\times$(2 中从 FA 出发的一条最长链，可以单独维护得到)。
4. 1 中从 U 的儿子出发的一条最长的链 $+1$ 中从 U 的儿子出发的一条次长链 $+2$。

这样复杂度是 $O(n)$ 的。

1005 About Set

把每个集合用一棵平衡二叉树维护，合并两个集合即为合并两颗平衡树，合并两个平衡树的过程中，可以将较小的一棵平衡树上的点一个个暴力插入点的个数较多的树上。

这样不难发现，每次插入个数若均需要最大，那么每个集合都应该是由两个元素个数相等的集合合并而成，也就是说合并最多有 $n \log n$ 次插入。

对于两种查询操作可以采取贪心的方式进行选点，对于已有两条边 a，b，如果集合中任意三点的权值无法构成三角形，那么第三边满足 $c \geq a+b$，对于任意两条边 $e+d > a+b$，也就是说这时第三边 $c \geq e+d > a+b$。

所以，最优方案是选择集合中权值最小的两个点，第三个点选择集合中权值大于等于这两个数之和的最小数，因为权值为正整数，所以极限情况下所选择的点权值呈斐波那契数列，也就是说每次查找的权值个数不会超过 50。

平衡二叉树的每个节点的左右两个子树的所有节点权值分别小于和大于这个节点的权值，也就是说只要在维护平衡树的过程中维护每棵子树的所有节点的权值的最大公约数，那么查询的时候需要访问的点的个数就是 $\log n$ 级别的。

1006 String

首先枚举 C 在 A 和 B 的起始位置，要使 A 和 B 的公共子序列尽量大，那么 C 在 A 和 B 的占用长度肯定越少越好。所以就分成两个问题：

1. 对 A，B 的任意起始位置，求出最近的包含 C 的结束位置。

   先记录一个数组 $A[i][j](0 \leq i < lenA, 0 \leq j < 26)$ 保存在 i 位置后跟 i 最近的字符 j 的距离。

   这个数组可以 $O(N^2)$ 暴力枚举出来。然后对于每个起始位置 i，按 C 从左往右寻找距离当前位置最近的相应字符即可。

2. 对于任意 A，B 的起始位置，求出 A，B 在起始位置之前的最长公共子序列，用同样的方法求出与起始位置相应的结束位置以后的最长公共子序列。

   只需定义 $dp[i][j]$ 代表 A 的 i 位置下 B 的 j 位置下的最长公共子序列长度。复杂度为 $O(lenA \times lenB)$。

上面两个问题都预处理后每次查询为 $O(1)$。所以总复杂度为 $O(N^2)$.

1007 The Happy Triangles

最佳方案：先让每个三角形的最短边靠着底面，并且在满足最优解的情况下，所有三角形的焦点以及三角形和两竖直杆的交点应该在同意高度。然后再去二分该高度，使得该高度下能够填满 R 区间至少 H 的高度，求得满足条件的最大的 H，最大的面积就应该是 $H \times R+$ 超出高度的部分的面积。

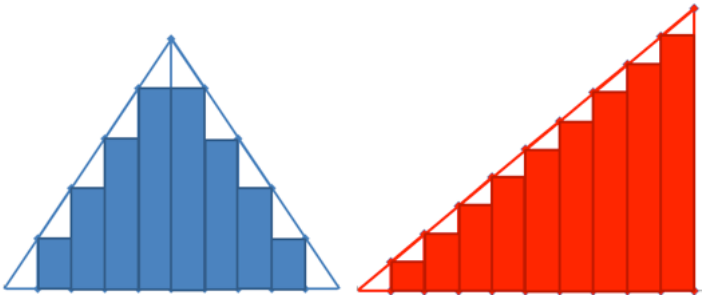
证明

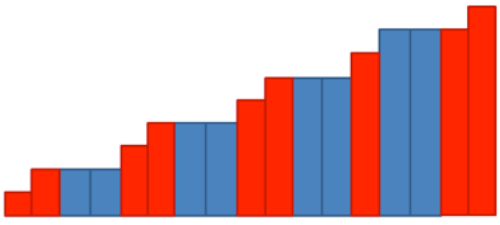
只需证明以下两点即可：

1. 最优策略应该选取最短边靠着底边，也就是说最短边靠着底边不会比其他两种方案差；
2. 当底边放置的方案确定时，二分高度 H，求得最大满足条件的高度 H，此时覆盖区域的面积是最大的。

先证命题 2

首先我们把所有三角形都沿着水平方向分成宽度为 $d(d \to 0)$ 的矩形，然后对这些矩形从小到大排序 (见下图)。 那么如果不考虑三角形的选择，





只考虑矩形的选择，那么面积最大的覆盖应该是选取最大的 $\frac{R}{d}$ 块矩形，那

么显然，选取三角形的覆盖方案肯定不会优于矩形的选择，而且每一种矩形的选择，可以找到对应的三角形区域的选择，用这 $\frac{R}{d}$ 块三角形去覆盖这些三角形，那么对于每个被覆盖的三角形，必然是中间连续的一段，被矩形覆盖，我们把这些被覆盖的区域连着一起，再覆盖区域 R，那么此时，三角形覆盖 = 矩形覆盖，即 2 可证。

再考虑命题 1

不妨假设最优的情况下 N 个三角形有一个三角形不是最短边靠着水平面，那么改局面下的最优策略按命题 2 可得，有 $N-1$ 个三角形靠边的方式不变，而且被选择的高度也确定了，那么现在就相当于是要在剩下未被覆盖的 R' 区域上用这一个三角形去覆盖，先只需证一个三角形的情况下，最短边为底边最优。

设底边的宽度为 L，被覆盖的面积

$$S_{cover} = (1 - (\frac{L - R'}{L})^2) \times S_{\triangle}$$

所以 L 是单调递减的，所以选择最小边是最优的。

PS：当然把每个三角形直接划成多个矩形排序，选择最高的 R 宽度也可以，但这样误差可能会很大。

1008 The Sad Triangles

可以证明，要使覆盖面积最大，矩形起点的横坐标必然是那些能使两边界被三角形斜边所截高度（如图的蓝色部分）一样的点或者端点。

而由于矩形长和三角形的端点都为整数，所以点的横坐标都为 0.25 的倍数（证明略）。把所有边都乘以 4，然后一个单位一个单位枚举面积即可，相当于 0.25 地枚举 (注意用 double 可能导致误差)。

1009 The Budget of Traveler

$dp[i]$ 表示到达 i 节点且送完礼物的最小花费，考虑线性情况下有：

$$dp[i] = min(p[i] \times \sum_{k=j+1}^{i} f[k] + r[i] + dp[j])$$

令 $sum[i] = \sum_{k=1}^{i} f[k]$，那么有：

$$dp[i] = min(p[i] \times (sum[i] - sum[j]) + r[i] + dp[j])$$

$$dp[i] = min(-p[i] \times sum[j] + dp[j]) + sum[i] \times p[i] + r[i]$$

考虑 $-p[i]$ 为斜率，$sum[j]$ 为横坐标，$dp[j]$ 为纵坐标，可以进行斜率优化，将有效点存放在栈中，那么后面的点可能选到的有效点只会在这些点和新产生的点中产生。

现考虑的为一棵树，那么将有效点的父亲节点设定为上一个有效点，那么每个有效点到根节点的路径上的点就是加入该点后的所有有效点。对于这些有效点，$-p[i] \times sum[j] + dp[j]$ 的值先减小后增加，故可以三分找到最小值。

找第 i 级祖先可以通过 rmq 查找。

插入新点也可以类似的通过三分找到插入位置。

1010 Prince and Princess

对于这个题，首先我们考虑一个简化版，即如同样例给出来的有 n 个王子和公主，并且他们能够完全匹配的情况。

我们先求出最大匹配，这时我们如果对于每个王子和他喜欢的每个公主都去枚举他们 link 的时候是否还能找到最大匹配，这样复杂度可能高达 $O(n^4)$，应该是会超时的。

那么我们换一个角度，枚举然后找最大匹配实质上是找到一个人和他互换，并且他们喜欢他们互相配对的公主。那么我们可以试着直接用连边来表示这个过程，对于每个王子，从他目前配对的公主出发连有向边到他喜欢的所有公主，表示他配对的公主单方面可以和这些公主换。若是两者可以互换，那么在图中就表示为了可以互相到达，从而想到对于这样的一个图求出他的强连通分量，对于每个王子，枚举他喜欢的每个公主，如果这个公主和他目前配对的公主在同一个强连通分量里，表示这个王子可以选该公主。

然后回到这个题，这个题变成了 n 个王子和 m 个公主，其实最主要的还是他们的最大匹配不一定是完全匹配。如果直接套用以上的方法就有问题。比如有一个王子，两个公主，这个王子两个公主都喜欢，那么求出的最大匹配只会和其中一个公主相连，通过上面的方法建图，两个公主并不会在一个强连通分量里面，但实际上这个王子选两个中的任意一个都不会使最大匹配减少。我们要怎么解决这个问题，这里又要从最本质的地方来看，就是一个互换。

这里的王子如果求出匹配连的是一号公主，那么二号公主没有和任何王子相连，这个王子就无法和任何人互换从而得到二号公主。考虑到这个问题，很自然的就能想到给二号公主虚拟一个王子和她配对就好了，然后这个虚拟的王子喜欢所有的公主。

因为这个公主本身是没有匹配的，如果通过虚拟一个喜欢所有公主的王子向所有其他公主连边，使得这个公主得到了一个匹配，那么肯定会有另外一边的匹配数减少了，因为最开始就求到的是最大匹配，如果这个公主匹配增加并且没有任何一边的匹配减少，肯定违背了最大匹配的性质。

所以得证这样一种虚拟王子连边的方法不会减少匹配数并且正确。

剩下公主的问题考虑完了，现在考虑剩下王子的问题。同理，我们给每一个没有匹配上的剩下的王子，虚拟一个公主，这个公主被所有王子喜欢，那么和最开始的做法一样求强连通然后考虑是否在一个强连通分量里即可。证明和剩下公主类似。