BestCoder Blog

BestCoder 官方博客

2016 Multi-University Training Contest 2 solutions BY zimpha

Acperience

展开式子, $\|W-\alpha B\|^2=\alpha^2\sum_{i=1}^{n}b_i^2-2\alpha\sum_{i=1}^{n}w_ib_i+\sum_{i=1}^{n}w_i^2$.

由于$b_i\in\{+1,-1\}$, 那么$\sum_{i=1}^{n}b_i^2=n$, 显然$c=\sum_{i=1}^{n}w_i^2$也是常数. 转化成求$\alpha^2n-2\alpha\sum_{i=1}^{n}w_ib_i+c$的最小值. 对于固定的$\alpha>0$, 只要$\sum_{i=1}^{n}w_ib_i$最大就好了. 显然$b_i=sign(w_i)$的时候, $\sum_{i=1}^{n}w_ib_i=\sum_{i=1}^{n}|w_i|$最大. 进一步的, 上面显然是一个关于$\alpha$的二次方程, 于是当$\alpha=\frac{1}{n}\sum_{i=1}^{n}w_ib_i=\frac{1}{n}\sum_{i=1}^{n}|w_i|$时, 取到最大值.

化简下, 可以得到最小值是$\sum_{i=1}^{n}w_i^2-\frac{1}{n}(\sum_{i=1}^{n}|w_i|)^2$

Born Slippy

感谢叉姐在ICPCCamp上出的这道题最初的原型 -- Data Structure You've Never Heard Of, 同样感谢Claris老师的教导.

由于and, or和xor方法都差不多, 这里仅考虑and操作. 不妨令$dp(s)=f(s)-w_s$, 我们大概要求的就是$dp(i)=\max_{j\text{ is ancestor of }i}\{dp(j)+w_i\text{ and }w_j\}$. 然后, 显然$dp(j)+w_i\text{ and }w_j$这个式子可以拆成$dp(j)$+[$w_i$后8位] and [$w_j$后8位] + ([$w_i$前8位] and [$w_j$前8位]) << 8.

先考虑序列上应该如何做, 即求$dp(i)=\max_{j<i}\{dp(j)+w_i\text{ and }w_j\}$. 考虑这样一个二维数组$ds(x,y)$, 表示对于某个$w_i$的后8位为$y$, 对于某个$w_j$的前8位为$x$时, $dp(j)$ + [w_i后8位] and [w_j后8位]的最值.

如果知道了上述数组, 那么对于某个i, 计算$dp(i)$的值就十分方便, 不妨令$w_i=(a<<8)|b$, 即a和b分别是w_i前8位和后8位, 那么只需要枚举w_j的前8位x, 用$ds(x,b)+((a\text{ and }x)<<8)$更新$dp(i)$. 把新的$dp$值更新到$ds(x,y)$也是类似的.

上述方法推广到树上也是十分简单, 由于每次更新$ds(x,y)$的时候只有数组的一维会变动(令a=w_i>>8, 那么只有$ds(a,\cdot)$会变化), 那么只要对数组的第一维做一个可持久化就好了(或者说边dfs边备份).

Call It What You Want

这个图大概就是一棵树, 然后最多加了5条边. 首先通过不断删掉度为1的点, 把这个图中属于树的部分全部砍掉, 那么我们会得到一个没有度等于1的点的图. 之后把图中度为2的点都缩掉, 最后得到一个图每个点的度至少为3. 显然最后得到的图最多只有8个点, 12条边(也许是10个点, 14条边, 但是随机出来的数据没有这种情况, 大概8个点12条边就是上界了吧).

考虑最长路的组成, 可以分为2种情况: 1. 在砍掉的树部分上; 2. 树上一条链+最终图上的一条路径+树上另一条链.

对于第一种情况, 在删度为1节点的时候就可以顺便计算出每个点u往下走的最长路f_u和次长路g_u, 显然答案就是$\max\{f_u+g_u\}$.

第二种情况有点复杂, 主要麻烦的地方在于多出来的2条链, 它们的位置有多种情况. 可能是在同一条边中延伸出来; 可能是在同一个环上延伸出来; 可能在两个不同的环上; 可能一个在环上, 另一个在普通边上. 根据这些情况, 大概要预处理出一些东西, 然后考虑枚举12条边的经过顺序, 然后在路径2边接上树上的链. 枚举经过顺序过程可以用状态压缩dp来优化. 需要注意的是最终图上的边也许会有重边.

Nero爷提供了一个比较暴力的方法, 和上面方法类似, 先把树上的一些东西都搞完, 接下来考虑多出的5条非树边. 那么可以5!暴力枚举这5条非树边的经过顺序(可能还要枚举下方向), 显然剩下来一定是要经过树边, 直接用树边把这些边接起来就好了(这里直接暴力bfs或者dfs就好了, 需要注意非树边上的点不要重复经过). 这个方法在测试的时候开长时限给放过了, 不知道比赛时候会不会因为一些不可知的原因而炸掉.

Differencia

感谢Claris老师教我如何卡常数 -- 只要数据范围够大就好了.

这道题$O(n\log^2 n)$的线段树套有序表做法很显然. 线段树每个节点[l,r]维护这个区间内, 数组b排序好的结果. 然后对于修改操作, 只要在这个区间内二分一下就能知道这个区间的答案(往子节点push lazy标记时也同理). 这个做法常数很小, 跑的很快, 但是应该被卡了(没测过zkw写法, 也许能过), 理由参考第一句话.

上面方法稍作修改就可以得到一个$O(n\log n)$的做法, 除了有序表线段树每个节点同时维护有序表第i个数进入左右子树时的位置. 那么只要在线段树根节点做一次二分, 之后就可以$O(1)$查询这个数在左右子树的rank变化. 这个对线段树往下push lazy标记也是适用的.

这个题应该还可以用平衡树+可持久化线段树做到$O(n\log n)$. 平衡树每个点保存a以及这个区间$a_i-b_i\geq 0$的个数, 那么查询就是然后子树和. 考虑修改操作, 暴力从平衡树中拿出这些区间, 然后合并成同一个, 新区间的$a_i-b_i\geq 0$的个数等价于对B的区间查询, 用可持久化线段树维护即可. 出题人没测过这个方法, 大概能过吧.

Eureka

xjb推导一下可以知道best set一定是一些共线的点, 于是问题变成问有多少个子集共线. 首先, 把所有点按照(x,y)双关键字排序, 然后枚举最左边的点i, 那么其他点j一定满足$j>i$. 把在这个点右边的点都做下极角排序(按照

$\frac{1}{gcd(dx,dy)}(dx,dy)$排序), 统计下共线的就好了. 需要注意下对重点的处理.

Fantasia

显然, 只要删掉关键点才会使图不联通. 对于其他点, 权值很容易计算.

首先求出所有的点双联通分量, 对于每一个点双联通分量S, 新建一个节点s, 向S中每个节点v连边. 这样一来, 新增的点和原来图中的点会构成一个森林(据说这个有个名字, block forest data structure). 很容易观察到, 叶子节点肯定都是非关键点, 内部节点要么是关键点, 要么是新增的节点.

对于这个森林F, 删掉一个关键点或者一个叶子i之后, 会得到一个新森林F_i, 这个F_i对应的连通块集合和G_i对应的连通块集合其实是一样的(不考虑那些新增的点). 显然G_i的权值和F_i的权值也是一样的, F_i的权值我们很容易通过树形dp算出来, 那么G_i的权值也随之而出.

Glorious Brilliance

首先对图二分染色, 如果不是二分图, 显然是无解的.

考虑给出图是连通二分图(不连通可以拆成若干个连通块分开搞)的时候, 枚举二分图两边集合的颜色, 观察下0和1的数目对不对. 如果是对的, 接下来考虑如何找到最少步数.

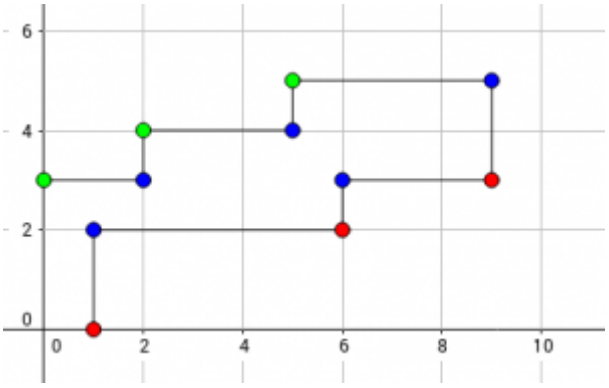
对于两个点u和v, 令它们之间的最短路是$dis(u,v)$, 那么交换它们两个颜色的最少步数是$dis(u,v)$, 且存在一种交换序列不会破坏其它节点的颜色. 证明如下:

不妨设u的颜色是0, v的颜色是1, u到v的最短路是$u \to x_1 \to x_2 \to \cdots \to x_s \to v$. 如果$u$和$x_1$颜色不一样, 直接交换它们即可. 否则找到第一个$i$使得$x_i$和$u$颜色不同, 通过下面交换操作$(x_i,x_{i-1})$, (x_{i-1},x_{i-2}), $\cdots$, (x_1,u)就可以把u的颜色搞到x_i上. 重复上述过程, u和v的颜色就交换了, 而且显然路径上其它点的颜色保持不变.

知道交换次数是最短路之后, 我们只搞清楚枚举谁和谁交换即可. 显然这是一个二分图最小权匹配问题, 可以套用KM或者费用流解决. 至于方案构造, 上面的证明就已经提供了构造方案. 确定匹配之后, 找出最短路, 然后对应地操作即可.

Helter Skelter

可以注意到对于一个固定的a, 可行的b一定是一个区间. 如果我们把所有可行的(a,b)画在二维平面上, 可以观察到一个有趣的现象: 这个可行区域一定是连通的, 且上下界有一些和x轴y轴平行的线段组成. 如下图所示.



显然, 求出这个上下边界这道题目就搞定了. 考虑求下边界, 观察上图可以知道, 求出所有红色的点就可以确定这个下边界. 同样, 所有绿色的点就可以确定上边界. 一个显然的猜想就是这些边界点肯定是由一些连续的run组成的, 红色点的run肯定是从0开始, 以0结尾, 绿色则是从1开始, 以1结尾. 假装这个猜想是对的, 接下来就是枚举这些连续的run, 然后随便排序下这些点对, 利用类似凸包的方法就可以求出这些红色or绿色的点. 确定了上下边界之后, 对于一个询问(a,b), 就可以二分出对应b的上下界.

It's All In The Mind

令$x=a_1+a_2, y=a_3+a_4+\cdots+a_n$, 那么$\frac{a_1+a_2}{a1+a2+\cdots+an}=\frac{x}{x+y}=1-\frac{y}{x+y}$. 对于定值$y$, 显然$x$越大越好, 对于定值$x$, 显然$y$越小越好. 于是按照$a_1$和$a_2$尽量大, 其他元素尽量小的策略填数就好了.

Join The Future

对于题目给出的m个关系, 显然可以确定出一些等价类, 我们删掉只有一个元素的等价类, 那么显然剩下等价类的个数不超过$\frac{n}{2}$, 于是可以暴力$O(2^{\frac{n}{2}})$枚举剩下等价类的值, dp出对应的方案数. 字典序最小也可以在dp的过程中顺便计算出来.

Keep On Movin

如果每个字符出现次数都是偶数, 那么答案显然就是所有数的和. 对于奇数部分, 显然需要把其他字符均匀分配给这写奇数字符. 随便计算下就好了.

La Vie en rose

题目给出的变换规则其实就是交换相邻元素, 并且每个元素最多交换一次. 那么一个$O(nm)$的dp其实十分显然, $dp_{i,j,k}$表示匹配到s的第i个字符, p的第j个字符, j这一位的当前状态是k (0表示和前面交换, 1表示没有交换, 2表示和后面交换). 转移方程如下:

$$dp_{i,j,0}=dp_{i-1,j-1,2}\ and\ s_i=p_{j-1}$$

$$dp_{i,j,1}=(dp_{i-1,j-1,0}\ or\ dp_{i-1,j-1,1})\ and\ s_i=p_j$$

$$dp_{i,j,2}=(dp_{i-1,j-1,0}\ or\ dp_{i-1,j-1,1})\ and\ s_i=t_{j+1}$$

这个dp数组里面存的都是bool值, 可以考虑用bitset压缩这个dp数组中的第一维i, 然后滚动下第二维j, 就得到了到$O(\frac{nm}{w})$的做法, 其中w是机器的字节长.

Memento Mori

虽然这题长着像分类讨论, 但是实际上不需要分类讨论.

显然最终的子矩形的左右边界会被两个1卡住, 不妨考虑枚举这两个1. 枚举完之后, 可以发现根据和排列p的相对位置关系, 事实上其他2个1的位置也是确定了的. 剩下的问题是如何快速定位另外两个1.

先把所有的1按照行优先的顺序排序, 考虑枚举做边界i, 然后维护一个i右边的那些1的列坐标c的一个有序表, 按照j从大到小枚举右边界j, 同时维护这个有序表(j枚举完之后删掉对应的列坐标), 那么显然只要根据i和j上下还需要几个1, 中间还需要几个1, 另外两个1就能够用$O(1)$时间在这个有序表上定位. 因为i和j在这个有序表上的位置我们可以事先维护好.

还需要注意同一行/列内有多个1的处理, 在维护有序表的同时加几个if就好了.

本条目发布于2016年7月21日 [http://bestcoder.hdu.edu.cn/blog/2016-multi-university-training-contest-2-solutions-by-zimpha/] 。属于多校题解分类。作者是wange。