CSP-S 2022 模拟赛

Solution

时间：2022 年 10 月 4 日 08:00 ~ 12:00

题目名称	简单难题练习题	往事成风	Ginger 的无向无环联通图	走廊
题目类型	传统型	传统型	传统型	传统型
目录	easyhard	imperishable	treeq	corridor
可执行文件名	easyhard	imperishable	treeq	corridor
输入文件名	easyhard.in	imperishable.in	treeq.in	corridor.in
输出文件名	easyhard.out	imperishable.out	treeq.out	corridor.out
每个测试点时限	1.0 秒	1.0 秒	1.0 秒	1.5 秒
内存限制	256 MB	512 MB	1024 MB	1024 MB
测试点数目	10	25	20	25
测试点是否等分	是	是	是	是

提交源程序文件名

对于 C++ 语言	easyhard.cpp	imperishable.cpp	treeq.cpp	corridor.cpp

编译选项

对于 C++ 语言	-lm -O2 -std=c++14

注意事项与提醒（请选手务必仔细阅读）

1. 文件名（程序名和输入输出文件名）必须使用英文小写。
2. C/C++ 中函数 main() 的返回值类型必须是 int，程序正常结束时的返回值必须是 0。
3. 提交的程序代码文件的放置位置请参照具体要求。
4. 因违反以上三点而出现的错误或问题，申诉时一律不予受理。
5. 若无特殊说明，结果的比较方式为全文比较（过滤行末空格及文末回车）。
6. 程序可使用的栈内存空间限制与题目的内存限制一致。
7. 只提供 Linux 格式附加样例文件。
8. 评测在 NOI Linux 下进行，各语言的编译器版本以其为准。

简单难题练习题（easyhard）

原题：【AGC013D】

~~为是 at 的题所以出题人想不到什么好的部分分做法~~

首先可以发现，盒子中的总球数是保持不变的

于是可以设 $f_{i,j}$ 表示前 i 次操作执行完，盒子中有 j 个黑球的方案数

那么当 $j>0$ 时，可以转移到 $f_{i+1,j-1}$, $f_{i+1,j}$

当 $j<n$ 时，可以转移到 $f_{i+1,j}$, $f_{i+1,j+1}$

但是这样有一个问题，对于不同的初始黑球数，可能会得到同样的序列

我们得考虑如何去重，这引导我们找出不同序列的特征

假设我们动态写出黑球数：

例如：4 3 4 3 2 3 2 ，把它想象成一条折线在折线往上/下移动的同时（即序列同时加上或减去一个数），我们可以得到相同的序列

而球数有 ≥ 0 的下界，所以计数最小值为 0 的黑球数序列就一定可以不重不漏地计数

随便加一维放进 dp 状态里面就行了

时间复杂度 $O(n^2)$

往事成风（imperishable）

【结论 1】

在最优的划分中，某个众数只会出现至多一次，否则显然可以将相同的众数所在的集合合并来减小答案。

【结论 2】

若不考虑编号最小的限制，那么将某些数字选为最后的众数，这个划分方案合法当且仅当没有选为众数的数，它们的个数的最大值小于等于选出的这些众数的个数之和。而原题目的最优解即为不考虑编号最小限制的解。

这样一来，我们就可以得到一个 $O(nq)$ 的做法：对于每次询问，我们贪心地从高位向低位枚举，如果当前的数字删去后仍满足上述条件，则删去，否则保留。最后保留的数字为选出的众数。

【结论 3】

设 V 为值域。

所有删去的数字形成的连续段数数量级为 $O(\sqrt{V})$。

证明：对于某个删去的数字，若它右边的数字没有被删掉，那么右边的这个数字一定大于前面的删去的数字之和，否则它就会被删去。而未被删去的数字之和大于等于 V 且小于等于 2V（值域决定），而删去的数字在其中共出现了平方次，因此删去的数字形成的连续段只有根号级别。

【算法】

这样，为了优化在序列上暴力找删除的连续段，我们改为在线段树上暴力找删除的连续段：若删去当前节点中的最小值仍不能满足条件，直接返回；若将右节点整个删去仍满足条件，那么将其删去并遍历左节点；否则遍历右节点。

总复杂度 $O(n\sqrt{V}logn)$，实际上很难卡满，期望得分 100。

Ginger 的无向无环联通图（treeq）

题目中对于“优”的定义等价于 $a_{i,j} = \max_{1\le k\le n} a_{k,j}$，假如把 $h_j = \max_{1\le k\le n} a_{k,j}$ 给算出来，那么后面的过程就比较简单了。

【算法 1】

根据题目描述中的定义直接计算 $a_{i,j}$ ，然后计算 h_i ，然后计算答案。

时间复杂度 $O(n^2)$ ，期望得分 10 。

【算法 2】

由于 $w_i = 1$ ，所以一个点的最优边一定出现在与它直接相连的边中，在每个点把边排序计算贡献即可。

时间复杂度 $O(n \log n)$，期望得分 20 。

【算法 3】

对于一条链的情况，分别考虑边在点左侧/右侧的影响形式。

当边在点左侧时，影响就是 w_i 乘边左边的点数；在右边是 w_i 乘边右边的点数。

发现这个边对点的影响形式可以前缀/后缀 max 来计算，于是可以 $O(n)$ 的计算 h_i。

考虑怎么计算边的答案，以计算边右边的点的贡献为例。假设边对右边点的影响为 x ，由于是取 max 所以一定有右边的点 $h \ge x$ 。因为要求有多少 $h = x$ ，所以只需统计右侧的 h 的最小值及其出现次数。计算左边的贡献同理。

时间复杂度 $O(n)$ ，期望得分 30

【算法 4】

对于菊花图的情况，h_1 可以轻松的算出来，那么叶子节点可以考虑维护所有 w 的最大值和次大值（和换根 DP 比较类似），然后简单判断计算。

再考虑计算答案，显然对应的叶子以及节点 1 可以轻松判断，压力主要是其他叶子，也可以类似的维护 h 的最小值和次小值以及对应的出现次数，简单判断计算。

时间复杂度 $O(n)$ ，期望得分 40

【算法 5】

这是一个线性做法。

还是先考虑计算 h_i ，那么可以分为子树内/外边的影响两部分来计算，显然可以换根 DP。

计算边的答案时，可以用类似算法 3 的分析方法和算法 4 的计算方法，只需要维护子树内/外 h 的最小值以及出现次数即可。

这个做法理论时间复杂度为 $O(n)$ ，但是常数非常大，所以实际得分 75 。

【算法 6】

还是一个线性做法，但是常数很小。

把 h 按照 dfn 序排列，发现一条边 (p_i, i, w_i) 对 h 的影响可以表示为让 $[1, dfn_i - 1], [dfn_i + siz_i, n]$ 这两个区间的 h 和 $w_i siz_i$ 取 max ，让 $[dfn_i, dfn_i + siz_i - 1]$ 里的 h 和 $w_i(n - siz_i)$ 取 max 。

第一部分的两个区间可以简单的用前/后缀 max 实现。

第二部分可以根据 dfn 的性质来优化，观察到 $[dfn_i, dfn_i + siz_i - 1]$ 的区间只有包含/不交关系，并且根据树的形态还知道这些区间一定能写成树的关系，那么可以设 f_i 表示考虑了所有包含 $[dfn_i, dfn_i + siz_i - 1]$ 的区间取 max 后 $[dfn_i, dfn_i + siz_i - 1]$ 的 max ，这个转移可以在 dfn 序上从前往后转移。显然，最后 f_i 就是只考虑第二部分的 h_i 。

然后考虑怎么计算答案。对于一条边 i 和它对应的三个区间，我们要找的就是这三个区间中等于 $w_i siz_i$ 或 $w_i(n - siz_i)$ 的 h 的数量。由于前面的操作是区间取 max ，所以 h 一定会大于等于我们要找的值，那么只需要知道这些区间中的最小值和其出现次数即可。

最小值和其出现次数的计算和上面求 h_i 的过程类似，不详细阐述。

时间复杂度 $O(n)$，期望得分 100 。

走廊（corridor）

【算法 1】

按题意模拟，复杂度 $O((n+m)Q)$。显然不能通过所有测试点。

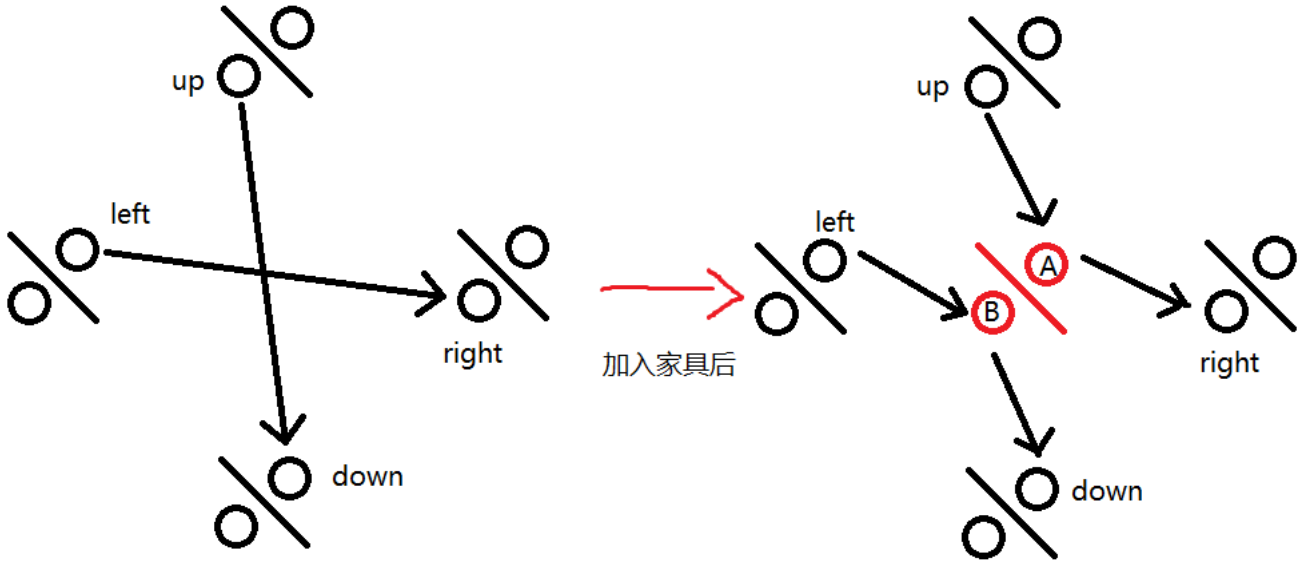
【特殊性质：后续不会加家具】

我们可以发现，RobotDuck 经过的家具形成了一条链。我们一开始将这些链 (注意，一个格子要分上下两个节点) 建出来，倍增之后便可快速回答。

复杂度 $O((k+Q)log(n+m))$，可以通过一部分测试点。

【特殊性质：总反弹次数不会太多】

我们考虑加入一件家具后链会发生什么。我们找到当前格子上面最近的一个节点 up，下面最近的一个节点 down，左边最近的一个节点 left，右边最近的一个节点 right。如果我们称当前格子右上角的节点为 A，左下角的节点为 B，那么 (up− >down)、(left− >right) 的关系会变
为 (up− >A− >right)、(left− >B− >down)。



用链表维护即可，复杂度为 $O(k+Q+N)$，其中 N 为反弹次数。可以通过一部分测试点。

【算法 2】

可以发现，上一个算法虽然修改很快，但询问时很慢。因此我们可以平衡一下复杂度。

将链表改为块状链表后，可以加速过程。对于每次修改或者询问，如果发现访问到的块的个数大于阈值，那么将整条链暴力重构。

合理调整块长后，可以做到 $O((k+Q)\sqrt{k+Q})$ 或 $O((k+Q)\sqrt{k+Q}log(k+Q))$，均可以过题。

【算法 3】

将块状链表改为平衡树，可以稳定做到 $O((k+Q)log(k+Q))$，代码复杂度也可能相应地下降。