NOIP 模拟赛题解

2018

tkandi

题目名称	naive 的瓶子	naive 的图	naive 的游戏
目录	colour	graph	game
可执行文件名	colour	graph	game
输入文件名	colour.in	graph.in	game.in
输出文件名	colour.out	graph.out	game.out
每个测试点时间限制	2s	2s	5s
每个测试点空间限制	512MB	512MB	512MB
测试点数目	20	20	20
每个测试点分值	5	5	5
是否有部分分	否	否	否
题目类型	传统型	传统型	传统型
是否有附加文件	是	是	是

提交源程序文件名

对于 C++ 语言	colour.cpp	graph.cpp	game.cpp
对于 C 语言	colour.c	graph.c	game.c
对于 Pascal 语言	colour.pas	graph.pas	game.pas

编译选项

对于 C++ 语言	-O2 -lm	-O2 -lm	-O2 -lm
对于 C 语言	-O2 -lm	-O2 -lm	-O2 -lm
对于 Pascal 语言	-O2	-O2	-O2

1 naive 的瓶子 (colour)

1.1 测试点 1-8

记忆化搜索，把当前的状态用 vector<int> 表示，用 map<vetor<int>, long long> 记录到达 vector<int> 的最小总代价，$O(n)$ 枚举转移。设 $m = min(n, max\{a_i\})$，时间复杂度为 $O(m^n \times n^3 \times \log m)$。

1.2 测试点 9-20

首先我们可以枚举最后的颜色 sc。

可以发现，一个瓶子要么直接被染成目标颜色，要么先被染成数值比较小的颜色，再被染成目标颜色。也就是说一个瓶子颜色的数值最多变小一次。这个结论十分显然。首先我们肯定是把一些颜色不为目标颜色的数值比较大的瓶子染成数值比较小颜色，但要确保还存在目标颜色，再最后把所有其他颜色染成目标颜色。在第一步中，一个瓶子颜色的数值肯定不会变大，此外，也不会变小两次，假设他变小了两次，那么我们完全可以直接变小成第二次，这样肯定不会变劣。

所以我们就可以动态规划了，设 f_i 为把前 i 个瓶子染成目标颜色的最小代价，每次转移选择接下来的一个区间，先把他们染成这个区间中数值最小的颜色，再染成目标颜色。

但需要注意的是，在第一步中不能把所有颜色都染成非目标颜色，那么我们可以类似 f_i 的再做一遍 g_i，为把 $[i, n]$ 染成目标颜色的最小代价。那么最后的答案为 $\min\{f_{i-1} + g_{i+1} | c_i = sc\}$。

总的时间复杂度为 $O(T \times n^3)$。

感觉这题的复杂度还可以优化，但是出题人太菜了。

2 naive 的序列 (sequence)

2.1 测试点 1-6

模拟题意跑 n 遍 Dijkstra 即可，时间复杂度为 $O(nm\log m)$。

2.2 测试点 7-10

可以发现原图的 $d(s,t)$ 等于最小生成树上的 $d(s,t)$。

因为 $L=0$，所以答案为所有满足 $u<v$ 的点对 (u,v) 的 $d(u,v)$ 之和。

只有在最小生成树的边才有贡献，一条边的贡献次数为在做 Kruskal 时它加入时连接的两个联通块的大小的乘积。

总的时间复杂度为 $O(m\log m)$。

2.3 测试点 11-14

由于点的颜色种数非常少，类比 $L=0$ 的做法，对于每个联通块维护每种颜色的点的个数。在合并两个联通块时暴力枚举配对的颜色，然后统计答案即可。

2.4 测试点 15-16

我也不知道怎么做。

2.5 测试点 17-20

先建出 Kruskal 重构树，每条边的贡献次数为它连接的两个子树之间的颜色之差大于等于 L 的点对数，可以发现 $\sum \min(size(leftchild_i), size(rightchild_i)) = O(n\log n)$。

对于每条边我们枚举 $size$ 较小的那棵子树内的点，算出在另一棵子树中能与它组成点对的点的个数。这个问题实际上就是询问在 dfs 序的一段区间上并且颜色不在一段区间内的点数，二维数点问题可以离线树状数组完成。

总的时间复杂度为 $O(m\log m + n\log^2 n)$。

3 naive 的游戏 (game)

本题由两部分组成。

3.1 测试点 1-2

直接建图跑最短路，时间复杂度为 $O(n \times range)$。

3.2 测试点 3-4,7-8

因为图的边权为 0 或 1，可以跑 01BFS，时间复杂度为 $O(range)$。

3.3 type = 0

把模 L 意义下相同的点缩成一个点，那么就成了一个环。把线段对应到环上，两点之间的最短路径只有两条，只需要判断这两条路径是否存在，若存在就更新答案。时间复杂度为 $O(n \log n)$。

3.4 type = 1

把模 L 意义下相同的并且可达的点缩成一个点。有一个显然的结论，只有包含起点或终点或某条线段的端点的点在途中是具有决策意义的，也就是只保留这些点在图中即可。

那么点数就是 $O(n)$ 了，如果再直接建图跑最短路，时间复杂度为 n^2，可以通过测试点 1-14 的 type = 1。（不确定能否卡掉测试点 15-20 的 type = 1）

图的边数当然也是可以优化成 $O(n)$ 的。我们把数轴每长度为 L 划为一行，把线段也对应其中。对于出现了两行或两行以上的满行或空行，我们把它们缩成一行。这样，行数就在 $3n$ 之内（见图 1 到图 2）。

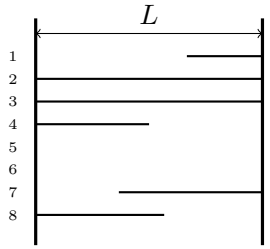

图 1

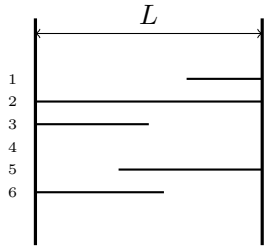

图 2

对于任意的关键点，让它一直向上扩展，一直向下扩展，得到了一段竖的线段，把它当成图中的一个点（见图 3 到图 4）。

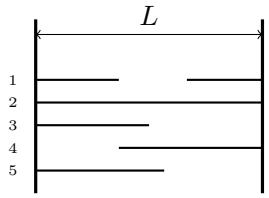

图 3

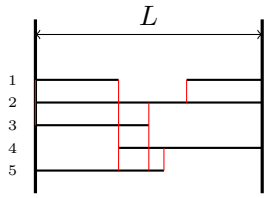

图 4

一共有 3 种边。第一种加入在第 i 行和第 $i+1$ 行之间的边（图 5 中绿色的边）。对于两个点，如果它们之间应该连边，即存在在这两个点（两段竖的线段）之间的横的线段（我们可以假设这些线段没有被其他点经过，如果存在这样的点，那么它必然连接了这两个点，并且距离也是正确的）那么把连接了这两个点的线段一直向上扩展，一直向下扩展，扩展到它们不再被线段连接了为止。此时，要么是因为一个点所代表的线段结束了，要么是因为出现了一段空段。所以第二种边是每个点所代表的线段的端点向左右连边（图 5 中蓝色的边）。第三种边是枚举每一段空段，假如这段空段的上面或者下面是联通的，那么把这段空段两端的点连接起来（图 5 中黄色的边）。

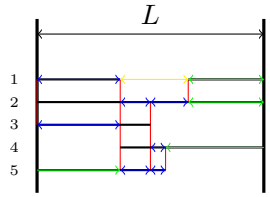

图 5

点数和边数都是 $O(n)$ 的，再跑 Dijkstra 即可，总的时间复杂度为 $O(n \log n)$。

~~因为出题人代码能力较差，代码量和常数较大。~~