NOIp2023 模拟赛题解

myee

题目名称	First Snow	Paradise	néo κósmo	Nirv lucE
题目类型	传统型	传统型	传统型	传统型
目录	`firstsnow`	`paradise`	`neokosmo`	`nirvluce`
可执行文件名	`firstsnow`	`paradise`	`neokosmo`	`nirvluce`
输入文件名	`firstsnow.in`	`paradise.in`	`neokosmo.in`	`nirvluce.in`
输出文件名	`firstsnow.out`	`paradise.out`	`neokosmo.out`	`nirvluce.out`
每个测试点时限	5.0 秒	5.0 秒	5.0 秒	5.0 秒
内存限制	1024 MB	512 MB	2048 MB	2048 MB
测试点数目	100	20	50	100
测试点是否等分	是	是	是	是

提交源程序程序名

对于 C++ 语言	`firstsnow.cpp`	`paradise.cpp`	`neokosmo.cpp`	`nirvluce.cpp`

编译选项

对于 C++ 语言	`-lm -O2 -std=c++14`

注意事项

1. 文件名（包括程序名，后缀名和输入输出文件名）必须使用英文小写。
2. 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为 `0`。
3. 提交的程序代码文件的放置位置请参照考场具体要求。
4. 若无特殊说明，输入文件中同一行内的多个整数、浮点数、字符串等均使用一个空格分隔。
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. 评测在~~当前最新公布的 NOI Linux~~ 信友队评测机下进行，**各语言的编译器版本以你提交时选择的语言为准，请勿错选！**
8. 最终评测时所用的编译命令中不含编译选项之外的任何优化开关。
9. 本次比赛各题时限均较长，**请不要卡评测**。
10. 题目背景中的概念可能比较模糊，建议阅读题目描述来获得相对准确的题意。

引言

大家好啊！不知道大家对这场模拟赛观感怎样。

这次模拟赛的四道题均为我本人原创，难度大致接近 CSP-S2022 和 NOIp2021。

四道题目正解均未采用大纲上超出提高级的知识点，而又有一定的难度，非常合适大家练习。

考察内容比较全面，从枚举到位并行，从贪心到容斥，从动态规划到图论，从字符串到数据结构，可以考察你对知识点掌握的熟练程度。

四道题目均给出了大量的部分分，可以模拟真实考试情况。

时限上则为了避免卡常，我放的非常宽松。~~希望没有低素质选手卡评测机。~~

最初的 T3 与 T4 并不是现在这两道，现在这两道是经过慎重考虑而换上的，以力求更好的模拟效果。

题目背景为出题人根据 Arcaea 支线剧情相关内容二次创作而来，文笔不佳请多见谅。

希望大家会喜欢这套题目。

First Snow（firstsnow）

预估难度：NOIp T1。

zak 觉得这题比后面几道难，我不好说。

本题基本没啥好给的部分分，这里直接介绍正解了。

$O(\text{poly}(n))$ 暴力

直接枚举其中几个点的行 / 列坐标，推出剩下的信息。

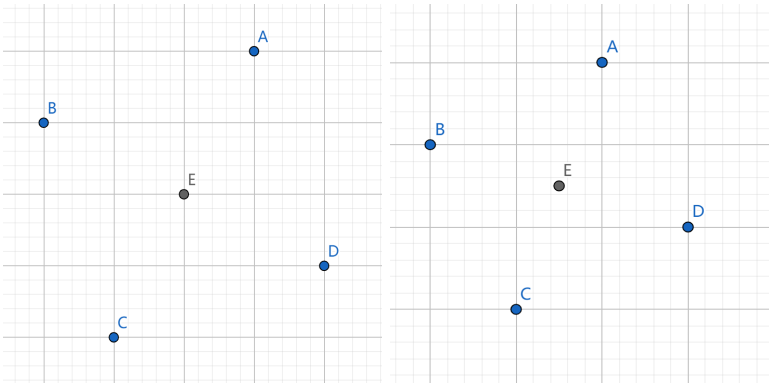
直接这么做起码也要 $O(n^4)$。

注意不要计重。

另一个 $O(n^4)$ 做法

我们观察到，一个正方形存在唯一的**中心**。

我们观察中心的形态。



我们发现，一个格点正方形的中心，一定是形如格点 (x, y)，或者是某个格点两维坐标均加上 0.5。

也就是中心的数目也是 $O(n^2)$ 的。

我们枚举这个中心，统计有多少个以当前点为中心的正方形即可。

我们考虑枚举中心右上方（不包括正上方）的那个点，然后就可以推出剩下三个点。

这样直接做是 $O(n^4)$ 的，根据常数实现可以获得不同分数。

优化

考虑优化。

假设中心是 (x, y)，那么对任一正方形，一定存在唯一的 $a > 0, b \geq 0$ 满足四个点形如

$$(x + a, y + b)(x - b, y + a)(x - a, y - b)(x + b, y - a)$$

我们考虑枚举 a，然后一个 b 合法当且仅当四个对应位置上有点。

容易发现可以直接用 `bitset` 快速统计这份信息。

由于实际就是在枚举当前正方形的外接正方形（外弦图?），枚举的次数大约带个 $1/3$ 左右的的常数。

然后由于每轮有取与和右移操作，以及一次 `.count()` 操作，又带来了 8 的常数，总共对 `bitset` 的操作次数是 $\frac{8}{3}n^3 + O(n^2)$ 次的。

从而这样的总复杂度即为常数不算太大的 $O(n^4/w)$，松一松可以通过。

Paradise（paradise）

预估难度：NOIp T2。

一道相对简单的题目，相信大家都能做出来。

介绍几档主要的部分分做法吧。

朴素算法

显然可以直接做背包。

复杂度 $O(nm+q)$，可以获得 40pts。

优化

注意到虽然 m 很大，但由于数据保证 $a_i|a_{i+1}$，本质不同的 a 其实只有 $O(\log v)$ 个。

对每个相同的 a，我们保留 b 最小的一份即可。

复杂度 $O(n\log v+m+q)$，可以获得 60pts。

正解

考虑在刚刚的基础上继续做。

我们注意到，如果 $a_{i+1}/a_i \le b_{i+1}/b_i$，那我们肯定不必去选择 a_{i+1} 这个元素，而是能替换成若干 a_i，故直接删除 (a_{i+1},b_{i+1}) 即可。

这样，我们最终的序列一定保证有

$$a_i/b_i < a_{i+1}/b_{i+1}$$

或者说性价比递增。

在此基础上我们考虑一个贪心，每次优先选取最大元素，一直取到不能取为止。容易证明这么做是对的。

假设剩下的选法为 $(a_1, b_1)(a_2, b_2)\dots(a_m, b_m)$，则我们也就有

$$f(n) = b_m\lfloor\frac{n}{a_m}\rfloor + \sum_{j=1}^{m-1} b_j\lfloor\frac{n \bmod a_{j+1}}{a_j}\rfloor$$

稍微改写一下

$$f(n) = b_m\lfloor\frac{n}{a_m}\rfloor + \sum_{j=1}^{m-1} b_j\lfloor\frac{n - a_{j+1}\lfloor\frac{n}{a_{j+1}}\rfloor}{a_j}\rfloor = \sum_{j=1}^{m}(b_j - [j>1]\frac{b_{j-1}a_j}{a_{j-1}})\lfloor\frac{n}{a_j}\rfloor$$

设 $c_j = b_j - [j>1]\frac{b_{j-1}a_j}{a_{j-1}}$，即

$$f(n) = \sum_{j=1}^{m} c_j\lfloor\frac{n}{a_j}\rfloor$$

而答案即为

$$\sum_{i=0}^{n-1} f(i) = \sum_{j=1}^{m} c_j \sum_{i=0}^{n-1}\lfloor\frac{i}{a_j}\rfloor$$

而我们有

$$\sum_{i=0}^{n-1}\lfloor\frac{i}{a}\rfloor = \lfloor n/a\rfloor(2n - a\lfloor n/a\rfloor - a)/2$$

（由于对 2^{64} 取模，可能还要对 $\lfloor n/a\rfloor$ 的奇偶性讨论一下）

这样我们就可以做到 $O(\log v)$ 解决单组询问了！

总复杂度 $O(m + q\log v)$，可以通过。

νέο κόsmo（neokosmo）

预估难度：NOIp T2/T3。
比较简单的图论与状压 dp 题？
因为没想好整个数据具体怎么造，数据可能不是很强。QwQ
部分分其实是给那些正解写挂 / 实现的不够优秀的人的。

$c = 1$ 的 4pts 做法

虽然感觉几乎没人会这么干但还是来说一下：
~~直接以关键点为起点，不移动，可以过掉其中两个点~~。
骗分选手素质很差，看出题人不是很会造数据就来骗！来偷袭！
不过起码还是卡了一个点。

$1 \le c \le 3$

$c = 1$ 时容易发现只用最小化走过的边权和 + 起点点权。
注意到环相当于就是重复两遍起点到关键点的最短路，直接跑从关键点开始的最短路即可。
$c = 2/3$ 时类似上面的做法，手玩分类讨论一下就好了。

$c \le 8$，$c \le 12$

给一些 $O(m2^c \log m)$ 之类的做法的……
直接状压 dp 搞一搞就可以了。具体做法类似正解。

$c \leq 15$，$c \leq 18$

给那些复杂度多带个 c 或者 $O^*(3^c)$ 之类的的做法的分……

n 比较小的情况

首先可以邻接矩阵存图，有用的边数也会减少。

同时，把正解的 Dijkstra 换成 SPFA 可以获得约 60pts 的好成绩。

定向卡了几种 SPFA，虽然不太能全卡掉都是了……

~~不会吧不会吧不会真有人写 SPFA 吧。~~

$c \leq 20$

正解分为 3 个部分。

缩图

我们发现，由于整个路径是一个环，我们不妨考虑令起点可以为环上任一结点：容易发现从这种视角来看，我们就不用单独枚举一个起点了。

我们考虑计算出从某一个关键点到达另一个关键点，中间不经过其他关键点时，此间所经过的最小边权总代价，以及中间经过了起点时的最小边权与起点总代价。

容易通过最短路实现：具体的，我们考虑拆点，把每个点拆成「还没经过起点」和「经过了起点」两部分，然后转移时不允许除了起点外的关键点向外松弛即可。

由于要跑 $O(c)$ 轮单源最短路，所以该部分复杂度朴素实现是 $O(cm \log m)$ 的。

这样我们就只用考察一个简化了的只有 c 个点的图了。

状压 dp

我们考虑从环上最大的关键点开始 dp。

状态中记录现在走到了哪个点，是否已经经过起点，当前走过的点集；权值记录当前经过的边和起点带来的代价。

这样的状态数是 $O(c2^c)$ 级别的。

然后转移时，我们按照点集顺序枚举 dp。

然后我们先对未经过起点的状态，按照当前已有的贡献从小到大的顺序来松弛别的路径，然后再对已经经过起点的状态同样松弛。

每次松弛到的关键点一定不能比开始 dp 的点大，由于开始 dp 的点已经记入状态（点集中最大一个）所以容易解决。

容易发现这就是一个 Dijkstra 的贪心过程，直接做即可。

暴力实现即为 $O(c^2 2^c)$ 的。

如果没有做上一步缩图，而是直接 Dijkstra 搞，那可以做到大约 $O(m2^c \log m)$ 左右的复杂度，也就是前面的部分分。

计算答案

我们枚举点集，然后令当前走到的点为当前点集中最大一个，且要求中间经过过起点，直接计入答案即可。

总复杂度 $O(cm\log m + c^2 2^c)$，可以通过。

Nirv lucE（nirvluce）

预估难度：NOIp T4 / NOI T1-T2。

这是一道想法不算很难，但对代码实现能力有一定要求的题目。

本题需要用到失配树的思想，属于 kmp 算法的自然延申；同时也需要使用树状数组和线段树合并快速统计贡献，具有一定的代码实现难度。

整体来说，本题思维层面并不算难，但综合考察了对 kmp 算法的理解，贡献统计的思想，以及简单数据结构维护信息的识别能力。

本题的部分分针对正解设计，一步步引导选手往正解方向思考，具有比较良好的区分效果。

这里介绍几档主要的部分分做法。

$m = 1$

很容易推出 / 大眼瞪出公式：

$$h_l(S) = \frac{(n-l-1)(n-l)(n-l+1)(n-l+2)}{12}$$

总复杂度 $O(n)$，期望得分 10pts。

实际得分可达 12pts。~~出题人下次再也不敢了。~~

$o = 1/2$

我也不知道有啥比较简单的做法。

不过如果你写过暴力的话，就会发现失配树高度和大约是 $O(n)$ 的；对于 $7 \sim 8$ 更是甚至没有除了本身的 Border。

以及对于后面的一些高复杂度做法来说，由于失配树高度毛估估就很矮，可以直接拿个暴力冲上去。

$O(n^3)$ 做法

考虑暴力枚举 $O(n^2)$ 对后缀的贡献。

考虑怎么计算相对代价。

我们考虑对第一个后缀枚举其所有 Border，这个可以用字符串哈希 / kmp 快速找出。

然后对第二个串判断这是否也为其 Border，这个可以用字符串哈希快速判断。

绝对代价的计算就是计算正反两遍相对代价，可以在 $O(n)$ 时间内解决。

最后差分统计贡献即可。

总复杂度 $O(n^3)$，期望得分 30pts。

$O(n^2)$ 做法

之前的做法太暴力了，没有利用 Border 的良好性质。

我们把原串 S 翻转，那么所有 S 的所有后缀也依次翻转成了各个前缀，每个串的所有 Border 也得到了对应翻转。

那么我们就是对原串的每个前缀考虑其 Border 集合的性质。

因此我们引入失配树。

失配树

考虑 kmp 的过程是对原串 S 的每个前缀 $S_{0\sim p}$ 求出其最长的非本身的 Border $S_{0\sim \pi_p}$，其中 $-1 \le \pi_p < p$。π_p 就是 kmp 数组，$\pi_p = -1$ 表示没有别的 Border。

而我们有结论，一个串 $S_{0\sim p}$ 的 Border 为

$$S_{0\sim p}, S_{0\sim \pi_p}, S_{0\sim \pi_{\pi_p}}, S_{0\sim \pi_{\pi_{\pi_p}}}, \dots$$

一直到某个位置 q 使得 $\pi_q = -1$ 为止。

证明是容易的：只用证明原串任何一个前缀的任意两个 Border 一定满足其中较短一个一定是较长一个的 Border 即可。

因此，我们考虑建立一棵树，树的节点编号 $-1 \sim n-1$，其中 -1 为根节点，对 $p = 0, 1, 2, \dots, n-1$ 的每个节点的父亲规定为 π_p，那么一个串的所有 Border 即为**其到根的链上所有的节点所代表的的字符串！**（除了根节点代表空串外）

这颗树我们就称为失配树，而 kmp 算法的过程其实就是在建失配树。

那么，我们发现，对两个前缀 $S_{0\sim a}$，$S_{0\sim b}$，设 a, b 在失配树上的 lca 为 $\mathrm{lca}(a,b)$，那么其共用的 Border 即为 $S_{0\sim \mathrm{lca}(a,b)}$ 的所有 Border（$\mathrm{lca}(a,b) = -1$ 时即为没有）。

因此设 $S_{0\sim p}$ 的 Border 总长为 A_p，那么我们有

$$A_p = \begin{cases} A_{\pi_p} + p + 1 & p \neq -1 \\ 0 & p = -1 \end{cases}$$

而我们有

$$f(S_{0\sim a}, S_{0\sim b}) = \max(A_a, A_b) - A_{\mathrm{lca}(a,b)}$$

因此直接树上前缀和预处理，ST 表加速计算 lca 即可。

总复杂度 $O(n^2)$，期望得分 56pts。

$c = 1$，$O(n \log n)$ 做法

紧接着上述做法，我们同样把原串翻转，考虑到当 $c = 1$ 时，我们只用计算

$$g(S) = \sum_{i=0}^{n-1} \sum_{j=i+1}^{n-1} f(S_{0\sim i}, S_{0\sim j})$$

也就是

$$g(S) = \sum_{i=0}^{n-1} \sum_{j=i+1}^{n-1} (\max(A_i, A_j) - A_{\mathrm{lca}(i,j)})$$

我们考虑拆贡献，得到 $\sum_{i=0}^{n-1} \sum_{j=i+1}^{n-1} \max(A_i, A_j)$ 和 $\sum_{i=0}^{n-1} \sum_{j=i+1}^{n-1} A_{\mathrm{lca}(i,j)}$ 两部分。

对于第一个部分，我们考虑对 A 进行排序，设依次为 $B_0, B_1, \ldots, B_{n-1}$，那么该部分贡献即为

$$\sum_{i=0}^{n-1} iB_i$$

这是简单的，考虑第二部分。

注意到第二部分的组合意义即为被容斥掉的贡献，因此我们考虑每个 Border 会被容斥掉几次：显然，设 p 对应的子树大小为 C_p，那么其会被容斥 $C_p(C_p - 1)/2$ 次，因此即为

$$\sum_{i=0}^{n-1} (i+1) \frac{C_i(C_i - 1)}{2}$$

直接一遍 dp 即可得到所有 C，从而解完。

总复杂度 $O(n \log n)$（使用基数排序可以优化到 $O(n)$，但没有必要），期望得分 42pts，如拼上上一档分则期望得分 70pts。

$O(nc \log n)$ 做法

注意到 $c > 1$ 时，在翻转原串后，我们其实就是求

$$h_l(S) = g(S_{0\sim n-l-1})$$

对每个询问，都跑一遍上面 $c=1$ 的做法。

总复杂度 $O(nc\log n)$，期望得分 $72\sim 86$pts。

$O(n\log n+nc)$ 做法

在刚刚这个做法的基础上，考虑我们按 $n-l-1$ 从小到大枚举，统计贡献。

那么每次往后面增加一个字符，就是给这颗树加一个叶子。

我们提前给整个 A 数组排好序，可以发现每次就是往 B 的中间某个位置插入一个数，可以直接 $O(n)$ 统计贡献。

而对于第二部分，跳父亲更新完 C 之后也可以直接 $O(n)$ 统计贡献。

总复杂度 $O(n\log n+nc)$，期望得分 86pts。

$O(n\log n)$ 做法

接下来介绍正解。

我们注意到上一个做法要往失配树上逐个加入叶子，我们不妨考虑能否直接维护答案的变化量而非答案本身。

对于第一部分贡献，我们考虑到往 B 数组中间插入元素时，除了该元素自身被枚举的次数外，在当前 B 之后的元素带来的贡献也会增加一次。

由于要插入元素，然后计算当前元素当前 rank 和后面元素的和，显然这可以用一个树状数组快速维护，做到 $O(n\log n)$。

对于第二部分贡献，我们注意到 C 所带来的贡献的变化仅会在子树内加入叶子时出现，当 p 子树内第 t 个点加入时会带来 $(p+1)(t-1)$ 的变化量。

不妨考虑线段树合并，在失配树上从下往上对子树进行 dp，每个位置记录当前位置答案的变化量为多少。那我们就只用支持合并操作和全局加变化量（关于当前相对大小为一次函数），在线段树上打 tag 即可直接解决。这样复杂度也是 $O(n\log n)$ 的。

总复杂度 $O(n\log n)$，期望得分 100pts，可以通过此题。

如果把线段树合并换成树剖之类的东西或许也能过，没有特意卡，因为其实也不好卡。