冲刺 NOIP2024 模拟试题

BSZX

时间：2024 年 10 月 15 日 08:00 ～ 12:30

题目名称	字典树	洄游	城市	多数
题目类型	传统型	传统型	传统型	传统型
目录	`trie`	`migration`	`city`	`majority`
可执行文件名	`trie`	`migration`	`city`	`majority`
输入文件名	`trie.in`	`migration.in`	`city.in`	`majority.in`
输出文件名	`trie.out`	`migration.out`	`city.out`	`majority.out`
每个测试点时限	1.0 秒	1.0 秒	3.0 秒	3.0 秒
内存限制	512 MiB	512 MiB	512 MiB	512 MiB
测试点数目	20	20	7	7
测试点是否等分	是	是	否	否

提交源程序文件名

对于 C++ 语言	`trie.cpp`	`migration.cpp`	`city.cpp`	`majority.cpp`

编译选项

对于 C++ 语言	`-O2 -std=c++14 -static`

注意事项（请仔细阅读）

1. 文件名（程序名和输入输出文件名）必须使用英文小写，不需要建立子文件夹。
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是 0。
3. 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）。
4. 选手提交的程序源文件必须不大于 100KB。
5. 程序可使用的栈空间内存限制与题目的内存限制一致。
6. 若无特殊说明，输入文件与输出文件中同一行的相邻整数均使用一个空格分隔。
7. 直接复制 PDF 题面中的多行样例，数据将带有行号，并且某些字符可能无法正常显示，建议选手直接使用对应目录下的样例文件进行测试。
8. 评测时采用的机器配置为：Intel(R) Core(TM) i7-10700 CPU @ 2.90GHz，内存 32GB。上述时限以此配置为准。

字典树（trie）

【题目描述】

你知道字典树吗？一个字典树是一棵有根树，树上的每条边有一个字符，一个节点所表示的字符串就是根到这个节点路径上边的字符依次拼起来得到的字符串。

我们认为一棵字典树是合法的，当且仅当所有节点表示的字符串互不相同。

小 X 有一棵每条边都有一个小写英文字母的 n 个节点的无根树，小 X 想要知道 $1 \sim n$ 中哪些点作为根时这棵树是一棵合法的字典树。

【输入格式】

从文件 ***trie.in*** 中读入数据。

第一行输入一个整数表示 n。

接下来 $n-1$ 行，每行输入两个整数和一个小写英文字母 u, v, c 表示点 u 和点 v 之间有一条字符为 c 的边。

【输出格式】

输出到文件 ***trie.out*** 中。

第一行输出一个整数表示有多少个点可以被作为根。

第二行输出若干个整数表示哪些节点作为根时字典树时合法的，你需要按照从小到大的顺序输出。

【样例 1 输入】

```
5
1 2 a
1 3 a
3 4 b
4 5 c
```

【样例 1 输出】

```
4
2 3 4 5
```

【样例 2】

见选手目录下的 ***trie/trie2.in*** 与 ***trie/trie2.ans***。

【样例 3】

见选手目录下的 ***trie/trie3.in*** 与 ***trie/trie3.ans***。

【数据范围】

对于 20% 的数据，满足 $n \leq 300$。

对于 50% 的数据，满足 $n \leq 5000$。

对于另外 20% 的数据，满足对于第 i 条边，$u = i, v = i + 1$。

对于全部数据，满足 $1 \leq n \leq 2 \times 10^5$，保证给出的是一棵树，保证 c 是小写英文字母。

洄游（migration）

【题目背景】

渔女格蕾丝体温极低且浸染湿气，走过的地方会留下水渊。水渊连线闭合形成封闭图形，并且其内部区域也变为水洲。已经闭合的封闭图形无法与其他水渊再次形成闭合区域。

对于格蕾丝而言，为了提升追击效率，一个封闭水洲的面积自然是越大越好。但是实际情况往往不尽如鱼意，有些地形是相对难以跨越的，为了形成较大的水洲而穿过这些地形，会浪费大量的时间。

现在，格蕾丝给了你一张网格图，请你计算她从某个点开始并在这个点停止，形成大小不小于 k 的水洲至少需要的时间。

【题目描述】

给定一个长为 n，宽为 m 的网格图，每条边有一个边权，边是双向连接的。现在在网格图中有一个点，初始位于坐标 (x, y)，你控制这个点移动，形成一段连续的轨迹，并且要求最终回到坐标 (x, y)。

令总面积 S 初始为 0。每次移动过后，若轨迹的一部分形成了一个封闭图形，则立即擦除这个部分。同时，设这部分面积是 $area$，则将总面积 S 加上 $area$。不难看出，轨迹在擦除这部分之后依然是连续的。特别地，如果沿着上一步移动过来的轨迹原路返回，那么这一部分也会被擦除，此时形成的面积视作 0。当总面积 S 不小于 k 并且回到起点时，移动停止。

每经过一条边就会耗费一次等同于这条边边权的代价，求在上述过程中所耗费的最小代价。

【输入格式】

从文件 ***migration.in*** 中读入数据。

第一行三个数 n，m 和 k，表示网格图的大小和至少需要的面积。

第二行两个数 x 和 y，表示起点（同时是终点）坐标。坐标从左上角开始编号，左上角为 $(1, 1)$。

下面 n 行，每行 $m-1$ 个数，依次表示水平边的边权；

下面 $n-1$ 行，每行 m 个数，依次表示竖直边的边权。

【输出格式】

输出到文件 ***migration.out*** 中。

输出一行一个数字，表示所耗费的最小代价。

【样例 1 输入】

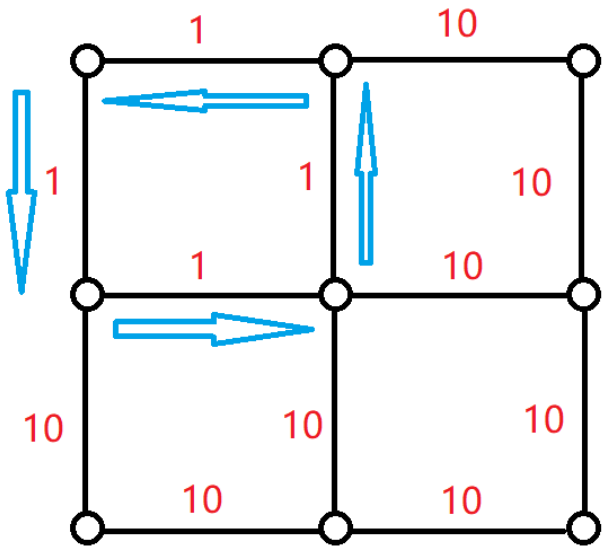
```
3 3 2
2 2
1 10
1 10
10 10
1 1 10
10 10 10
```

【样例 1 输出】

```
8
```

【样例 1 解释】

网格图总共三行三列，左上角的边权都是一，剩下是十。最好的方法是在左上角绕两圈，分别形成面积为 1 的两个封闭正方形，总代价是 8。



【样例 2】

见选手目录下的 *migration/migration2.in* 与 *migration/migration2.ans*。

【样例 3】

见选手目录下的 ***migration/migration3.in*** 与 ***migration/migration3.ans***。

【样例 4】

见选手目录下的 ***migration/migration4.in*** 与 ***migration/migration4.ans***。

【样例 5】

见选手目录下的 ***migration/migration5.in*** 与 ***migration/migration5.ans***。

【数据范围】

对于 10% 的数据，有 $n = m = 2$。
另外对于 5% 的数据，有 $n, m \leq 3$。
另外对于 5% 的数据，有 $n, m \leq 4$。
另外对于 30% 的数据，有 $n, m \leq 10$。
另外对于 20% 的数据，有 $n = 2$。
对于 100% 的数据，$2 \leq n, m \leq 20$，边权均不大于 10^5，$k \leq n \times m$。

城市（city）

【题目描述】

X 国有 n 座城市，这些城市通过 $n-1$ 条道路连接，形成一棵树形网络。每条道路都有一个长度。编号为 1 的城市为 X 国的首都。

X 国的交通规划部门计划在城市 x 和城市 y 之间新建一条长度为 z 的道路。然而，由于预算限制，必须拆除一条现有的道路以筹措资金。规划部门希望在拆除一条道路后，新的道路网络仍然保持连通，并且使**所有城市到首都的距离之和**达到最小。

此外，规划过程中可能会对有些道路进行整修，即道路长度发生变化。总共有 q 个操作，分为两种操作类型：

- 1 x y z：**新建道路**，表示在城市 x 和城市 y 之间新建一条长度为 z 的道路
- 2 x y z：**整修道路**，表示一条连接城市 x 和城市 y 的现有道路被整修，其长度变为 z。

对于每一种新建道路的方案（即每个 $op=1$ 的操作），需计算实施该方案后**所有城市到首都的距离之和**的最小值。需要注意的是，**新建道路的操作是独立的，仅用于查询，不会实际修改道路网络**。

【输入格式】

从文件 ***city.in*** 中读入数据。

第一行包含两个整数 n 和 q，分别表示 X 国的城市数量和操作次数。

接下来的 $n-1$ 行，每行包含三个整数 u, v, w，表示城市 u 和城市 v 之间有一条长度为 w 的道路。

接下来的 q 行，每行包含四个整数 op, x, y, z：

- 如果 $op=1$，表示在城市 x 和城市 y 之间新建一条长度为 z 的道路。
- 如果 $op=2$，表示一条连接城市 x 和城市 y 的现有道路被整修，其长度变为 z。

【输出格式】

输出到文件 ***city.out*** 中。

对于每一个新建道路的方案（即每个 $op=1$ 的操作），输出一个整数，表示实施该方案后**所有城市到首都的距离之和**的最小值。

【样例 1 输入】

```
7 4
1 2 1
```

```
1 3 1
2 4 1
2 5 1
3 6 1
3 7 2
1 4 5 1000
1 6 7 9999
2 1 2 5000
1 3 5 1
```

【样例 1 输出】

```
1011
10009
15
```

【样例 1 解释】

对于第一个方案，拆除 2 号城市和 4 号城市之间的道路。
对于第二个方案，拆除 3 号城市和 6 号城市之间的道路。
对于第三个方案，拆除 1 号城市和 2 号城市之间的道路。

【样例 2】

见选手目录下的 ***city/city2.in*** 与 ***city/city2.ans***。
该样例满足子任务 1 的特殊性质。

【样例 3】

见选手目录下的 ***city/city3.in*** 与 ***city/city3.ans***。
该样例满足子任务 2 的特殊性质。

【样例 4】

见选手目录下的 ***city/city4.in*** 与 ***city/city4.ans***。
该样例满足子任务 3 的特殊性质。

【样例 5】

见选手目录下的 ***city/city5.in*** 与 ***city/city5.ans***。
该样例满足子任务 4 的特殊性质。

【样例 6】

见选手目录下的 ***city/city6.in*** 与 ***city/city6.ans***。
该样例满足子任务 5 的特殊性质。

【样例 7】

见选手目录下的 ***city/city7.in*** 与 ***city/city7.ans***。
该样例满足子任务 6 的特殊性质。

【样例 8】

见选手目录下的 ***city/city8.in*** 与 ***city/city8.ans***。
该样例满足子任务 7 的特殊性质。

【数据范围】

本题采用捆绑测试。
输入保证初始道路构成一棵树，且每次整修的道路均存在。
Subtask 1（10 分）：$n, q \leq 300$。
Subtask 2（10 分）：$n, q \leq 5000$ 且没有 2 操作。
Subtask 3（10 分）：$n, q \leq 5000$。
Subtask 4（10 分）：$n, q \leq 2 \times 10^5$ 且树高不超过 $2 \cdot \log n$ 且没有 2 操作。
Subtask 5（10 分）：$n, q \leq 2 \times 10^5$ 且树高不超过 $2 \cdot \log n$。
Subtask 6（20 分）：$n, q \leq 2 \times 10^5$ 且没有 2 操作。
Subtask 7（30 分）：$n, q \leq 2 \times 10^5$。
对于所有数据，保证所有输入整数在 $[1, 10^6]$ 范围内。

多数（majority）

【题目描述】

多数元素定义为：在一个正整数序列中出现次数不少于序列长度一半的数字。

如果一个序列的所有非空连续子序列都至少包含一个**多数元素**，则称该序列为**良好序列**。例如，序列 $[1,2,1,1,3]$ 是良好的，因为像 $[1,1,3]$、$[1,2]$ 和 $[2,1,1,3]$ 这样的子序列都包含多数元素。但是，序列 $[1,2,1,3]$ 不是良好的，因为子序列 $[2,1,3]$ 没有多数元素。

给定一个由 1、2、3 和 ? 组成的字符串，计算将每个 ? 替换为 1、2 或 3 以形成良好序列的方式总数。输出答案对 998244353 取模的结果。

【输入格式】

从文件 ***majority.in*** 中读入数据。

第一行包含一个整数 n，表示字符串的长度。

第二行包含一个长度为 n 的字符串，由 1、2、3 和 ? 组成。

【输出格式】

输出到文件 ***majority.out*** 中。

输出一行一个整数，表示答案对 998 244 353 取模的结果。

【样例 1 输入】

```
3
???
```

【样例 1 输出】

```
21
```

【样例 1 解释】

对于第一个样例，唯一的不良序列（在 $3^3 = 27$ 个可能的序列中）是 $[1,2,3]$ 及其全排列，因此答案是 $27 - 3! = 21$。

【样例 2 输入】

```
4
1?11
```

【样例 2 输出】

```
3
```

【样例 2 解释】

对于第二个样例，序列如 $[1,1,1,1]$、$[1,2,1,1]$ 和 $[1,3,1,1]$ 都是好的。

【样例 3 输入】

```
5
12213
```

【样例 3 输出】

```
0
```

【样例 3 解释】

对于第三个样例，由于 $[1,2,2,1,3]$ 不是好的，答案是 0。

【样例 4】

见选手目录下的 ***majority/majority4.in*** 与 ***majority/majority4.ans***。
本组样例满足子任务 1 的性质。

【样例 5】

见选手目录下的 ***majority/majority5.in*** 与 ***majority/majority5.ans***。
本组样例满足子任务 2 的性质。

【样例 6】

见选手目录下的 ***majority/majority6.in*** 与 ***majority/majority6.ans***。
本组样例满足子任务 3 的性质。

【样例 7】

见选手目录下的 ***majority/majority7.in*** 与 ***majority/majority7.ans***。
本组样例满足子任务 4 的性质。

【样例 8】

见选手目录下的 ***majority/majority8.in*** 与 ***majority/majority8.ans***。
本组样例满足子任务 5 的性质。

【样例 9】

见选手目录下的 ***majority/majority9.in*** 与 ***majority/majority9.ans***。
本组样例满足子任务 6 的性质。

【样例 10】

见选手目录下的 ***majority/majority10.in*** 与 ***majority/majority10.ans***。
本组样例满足子任务 7 的性质。

【数据范围】

对于所有数据，保证 $3 \le n \le 500$，且字符串由 1、2、3 和 ? 组成。
让 $c_1, c_2, c_3, c_?$ 分别表示字符串中 1、2、3 和 ? 的出现次数。

子任务编号	$n \le$	特殊性质	分值
1	10	无	5
2	60	$2 \cdot \max\{c_1, c_2, c_3\} \ge n$	15
3		无	15
4	25	$c_? = n$	15
5	500		10
6	200	无	30
7	500		10