CSP2019 模拟赛

Day 1

时间：2019 年我也不知道什么时候

题目名称	雨中的晴天	燃烧的火焰	消失的序列
题目类型	传统型	传统型	传统型
目录	weather	flame	stack
可执行文件名	weather	flame	stack
输入文件名	weather.in	flame.in	stack.in
输出文件名	weather.out	flame.out	stack.out
每个测试点时限	1.0 秒	1.0 秒	1.0 秒
内存限制	512MB	512MB	512MB
子任务数目	10	25	25
测试点是否等分	是	是	是

提交源程序文件名

对于 C++ 语言	weather.cpp	flame.cpp	stack.cpp
对于 C 语言	weather.c	flame.c	stack.c
对于 Pascal 语言	weather.pas	flame.pas	stack.pas

编译选项

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

注意事项：

1. 由于测评机的配置问题，请注意时间、空间限制和环境问题。
2. 测评时栈空间与内存限制相同。
3. 时间限制保证在标程的两倍以上。

雨中的晴天（weather）

【题目描述】

宫水三叶生活的城市是一个一维平面上的城市。三叶喜欢用一个长度为n的线段来表示这座城市。线段上（包含端点）平均分布着$n+1$个点，其中第i个点到第$i+1$个点视为第i个区。

最近，这座城市不断的下雨，一直没有放晴，所有人都在期待的晴天。不同的区对晴天的渴望度不一样。三叶通过统计，将第i个区的人对晴天的渴望度形式化成s_i。

终于，这座城市迎来了久违的晴天。但是晴天的范围没有覆盖整个城市，而是从$n+1$个点中的某一个出发，向往扩散d个区。

在晴天下的人们非常开心。形式化的，如果第i个区在晴天的覆盖范围内，并且和晴天中心还隔着x个区，那么这个区的人的开心值为$(d-x)^2 \cdot s_i$。这个城市的开心值为每一个区的开心值之和。

虽然晴天的地点已经固定了，但是三叶还是想知道如果晴天的地点可以任选，那么最后城市的开心值最大是多少？

【输入格式】

从文件 weather.in 中读入数据。

总共两行，第一行一个整数n，表示这座城市的长度。

第二行n个整数，第i个整数表示s_i。

【输出格式】

输出到文件 weather.out 中。

输出一行，一个整数，表示可能的最大开心值。

【样例 1 输入】

```
6 2
3 2 1 1 2 3
```

【样例 1 输出】

```
21
```

【样例 1 解释】

这座城市共有 6 个区，其中每一个区的渴望度分别为{3,2,1,1,2,3}。

如果晴天中心在 1 号点，那么第一区的人的开心值为$3 \times 2^2 = 12$，第二个区的人的开心值为$2 \times 1^2 = 2$。那么这座城市的总开心值为$12 + 2 = 14$。

如果晴天中心在 2 号点，那么第一区的人的开心值为$3 \times 2^2 = 12$，第二区的人的开心值为$2 \times 2^2 = 8$，第三区的人的开心值为$1 \times 1^2 = 1$。那么这座城市的总开心值为$12 + 8 + 1 = 21$。

如果晴天中心在 3 号点，那么第一区的人的开心值为$3 \times 1^2 = 3$，第二区的人的开心值为$2 \times 2^2 = 8$，第三区的人的开心值为$1 \times 2^2 = 4$，第四区的人的开心值为$2 \times 1^2 = 2$。那么这座城市的总开心值为$3 + 8 + 4 + 2 = 17$。

类似的，如果晴天中心在 4 号点，那么开心值为12；如果晴天中心在 5 号点，那么开心值为17，如果晴天中心在 6 号点，那么开心值为21；如果晴天中心在 7 号点，那么开心值为14。

综上所述，晴天中心在 2 号点或 6 号点是，城市的开心值达到最大值21。

【样例 2 输入】

见选手目录下 `weather\weather2.in`。

【样例 2 输出】

见选手目录下 `weather\weather2.ans`。

【数据范围和提示】

对于所有数据，满足$1 \le n \le 10^6, 1 \le d \le 3 \times 10^4, 1 \le s_i \le 10^3$。

每个测试点具体限制见下表。

测试点编号	n	特殊性质
1	≤ 5	无
2 ~ 4	$\le 2 \times 10^3$	无
5 ~ 6	$\le 10^5$	无
7	$\le 10^6$	$s_i = 1$
8 ~ 10		无

燃烧的火焰（flame）

【题目描述】

宫水三叶擅长手工，她自己编织了一张网。

这张网可以用一个n个点m条边的连通图来表示，每一条边都有长度。

但是这张网毕竟是可燃物。某一天，网上的k个节点在0时刻突然同时被点燃了，火焰以单位速度沿着边向外扩散。具体来说，如果有一条长度为l的边连接着点x,y，假设第i个时刻x节点被点燃了，那么在$i+l$的时刻y节点也会被点燃。反之也是成立的。

如果整张图的n个节点全部被点燃了，那么就认为这张图完全被点燃了。

既然着火了，那么首要任务就是救火。三叶请小 H 来帮忙。在0时刻时，小 H 随机选择了若干个已经被点燃的点，将它们扑灭。但是，小 H 扑灭了那些点后并没有使整张图完全被点燃的时间推晚！

三叶觉得小 H 运气太差了，于是她想知道这个事件的概率。

形式化的说，小 H 有2^k种灭火方案（包含一个都不选）。假设小 H 随机从中选一种，有多少概率选到的灭火方案没能使整张图完全被点燃的时间推晚。

假设在没有灭火时整张图完全被点燃从时刻a开始，灭火后整张图完全被点燃从时刻b开始，而没能使整张图完全被点燃的时间推晚的方案当且仅当$a=b$。

【输入格式】

从文件 `flame.in` 中读入数据。

总共$m+2$，第一行三个整数n,m,k，表示这张网的点数，边数，和在0时刻被点燃的点数。

第二行k个整数，表示和在0时刻被点燃的点。

接下来m行，每一行三个整数x_i,y_i,l_i，表示第i条边连接着x_i,y_i，其长度为l_i。

【输出格式】

输出到文件 `flame.out` 中。

输出一行，一个整数，表示可能的概率。为了避免精度的误差，概率统一输出对998244353取模的结果。

概率取模的定义如下：如果概率是一个有理数，假设它可以表示为$\frac{a}{b}$，如果在

[0,998244352]中能找到整数x，满足$bx \equiv a\ (mod\ 998244353)$。那么我们称$x$为$\frac{a}{b}$对998244353取模的结果。

【样例 1 输入】

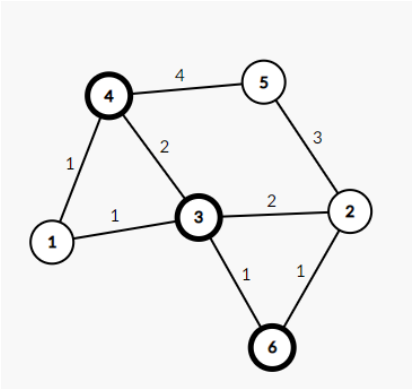
```
6 8 3
3 4 6
1 3 1
1 4 1
3 4 2
2 3 2
2 6 1
3 6 1
2 5 3
4 5 4
```

【样例 1 输出】

```
249561089
```

【样例 1 解释】

这张网有 6 个节点，有 8 条边。其中节点 3,4,6 一开始着火了。



初始时最后一个被点燃的节点为 5 号节点，它在时刻 4 被点燃。

一共有 8 种灭火方案，其中方案{3}{4}{6}{3,4}{3,6}和没有选择任何一个点灭火整张图被点燃的时刻还是 4。

对于方案{4,6}，最后被点燃的节点为 5 号点，在时刻 5 被点燃。

对于方案{3,4,6}，最后图不会被点燃。

因此，小 H 选择的概率为$\frac{6}{8} = \frac{3}{4}$。$\frac{3}{4}$在对998244353取模后为249561089。

【样例 2 输入】

见选手目录下 `flame\flame2.in`。

【样例 2 输出】

见选手目录下 flame\flame2.ans。

【样例 3 输入】

见选手目录下 flame\flame3.in。

【样例 3 输出】

见选手目录下 flame\flame3.ans。

【样例 2、3 解释】

样例二满足$n \leq 10^3, k \leq 8$。样例三满足$n \leq 10^5, k \leq 12$和特殊性质 A。

【数据范围和提示】

对于所有数据，满足$1 \leq n \leq 10^5, 1 \leq m \leq 2 \times 10^5, 1 \leq k \leq 20$。

整张图保证没有重边和自环，并且保证联通。

k个在0时刻被点燃的点的编号互不相同。

对于每条边满足$1 \leq x_i, y_i \leq n, 1 \leq l_i \leq 10^9$。

每个测试点具体限制见下表。

测试点编号	n	k	特殊性质
$1 \sim 2$	≤ 10	≤ 3	无
$3 \sim 6$	$\leq 10^3$	≤ 8	$m \leq 2 \times 10^4$
7	$\leq 10^5$		特殊性质 A
8			特殊性质 B
$9 \sim 10$			无
$11 \sim 12$		≤ 15	特殊性质 A
13			特殊性质 B
$14 \sim 16$			无
$17 \sim 19$		≤ 20	特殊性质 A
20			特殊性质 B
$21 \sim 25$			无

特殊性质 A:满足$m = n - 1$，对于第i条边，满足$x_i = i, y_i = i + 1$。
特殊性质 B:满足$m = n - 1$。

消失的序列（stack）

【题目描述】

宫水三叶曾经有一个栈和序列。如今，栈还存在，序列却随着时间的流逝消失了。

这个序列曾经是三叶最引以为豪的序列，因为这个序列有着特殊的性质。三叶可以在小 H 面前，用手中的栈将这个序列排成升序。

三叶给序列排序是遵循一定的规则的，规则如下：

起初，这个栈是空的，而序列是原本的样子。

接着，三叶每一次可以对序列进行入栈和出栈两种操作。

入栈操作：如果序列非空，那么三叶会取出序列的第一个元素，把它扔进栈的顶部。

出栈操作：如果栈非空，那么三叶可以取出栈的顶部，把这个元素接在结果序列的末尾。

注意结果序列和原序列是不一样的，也就是出栈操作不是把元素接在原序列后。

如果所有元素都经过了一次入栈和出栈操作，得到的结果序列为升序，那么这个序列就被排好序了。

现在这个序列消失了，三叶望着手中的空栈发呆。

在三叶的记忆中，这个序列的长度为n，栈中的元素为$1 \sim n$的排列。

凭着残缺的记忆，三叶还记得这个序列某一位上的数字。

三叶决定找回这个序列，她想知道合法的序列个数。

但是合法的序列太多了，三叶决定求出答案$mod\ 10^9+7$后的结果。

【输入格式】

从文件 `stack.in` 中读入数据。

总共一行，三个正整数n, pos, x，表示在三叶记忆中这个序列长度为n，并且序列的第pos位为x。

【输出格式】

输出到文件 `stack.out` 中。

输入一行，一个整数，表示合法的序列个数$mod\ 10^9+7$后的结果。

【样例 1 输入】

```
4 3 3
```

【样例 1 输出】

```
4
```

【样例 1 解释】

所有的合法序列为$\{1,2,3,4\},\{1,4,3,2\},\{2,1,3,4\},\{4,1,3,2\}$。

序列$\{1,2,4,3\}$是不合法的，因为序列的第三位不为 3。

序列$\{4,2,3,1\}$是不合法的，因为这个序列没有办法利用栈排成升序。

【样例 2 输入】

```
18 6 9
```

【样例 2 输出】

```
14376219
```

【样例 3 输入】

```
2000 1234 666
```

【样例 3 输出】

```
961509521
```

【样例 4 输入】

```
2000 2000 233
```

【样例 4 输出】

```
834559547
```

【数据范围和提示】

对于所有数据，满足$1 \le n \le 10^6, 1 \le pos, x \le n$。

每个测试点具体限制见下表。

测试点编号	n	特殊性质
$1 \sim 2$	≤ 9	无
$3 \sim 4$	≤ 19	无
$5 \sim 6$	≤ 300	无
7	$\leq 2 \times 10^3$	$pos = x = 1$
$8 \sim 9$		$pos = n$
$10 \sim 11$		$pos = 1$
$12 \sim 18$		无
$19 \sim 25$	$\leq 10^6$	无