

暑期模拟赛

CSPJ 模拟赛 1

时间：2024 年 7 月 29 日 08:15 ~ 11:45

题目名称	天才拆分	自学	打 ACM 最快乐的就是滚榜读队名了	找爸爸
题目类型	传统型	传统型	传统型	传统型
目录	<code>cirno</code>	<code>study</code>	<code>submit</code>	<code>dnaseq</code>
可执行文件名	<code>cirno</code>	<code>study</code>	<code>submit</code>	<code>dnaseq</code>
输入文件名	<code>cirno.in</code>	<code>study.in</code>	<code>submit.in</code>	<code>dnaseq.in</code>
输出文件名	<code>cirno.out</code>	<code>study.out</code>	<code>submit.out</code>	<code>dnaseq.out</code>
每个测试点时限	1.0 秒	1.0 秒	5.0 秒	1.0 秒
内存限制	128 MiB	512 MiB	1024 MiB	512 MB
测试点数目	20	5	20	10
测试点是否等分	是	否	是	是

提交源程序文件名

对于 C++ 语言	<code>cirno.cpp</code>	<code>study.cpp</code>	<code>submit.cpp</code>	<code>dnaseq.cpp</code>
-----------	------------------------	------------------------	-------------------------	-------------------------

编译选项

对于 C++ 语言	<code>-O2 -std=c++14</code>
-----------	-----------------------------

天才拆分 (cirno)

【题目描述】

琪露诺定义一个十进制正整数为「 k 阶天才数」，当且仅当该整数的位数为 k 的倍数，且每一个数位均为 9。例如，9999 是 2 阶天才数，而 999 不是 2 阶天才数，但是它是 1 阶天才数，也是 3 阶天才数。

琪露诺给定你 t 个询问，每个询问有两个整数 n 和 k ，希望你能帮她，告诉她能不能把 n 拆分成若干个 k 阶天才数的和。

【输入格式】

从文件 *cirno.in* 中读入数据。

- 第一行是一个整数 t ，代表询问的个数。
- 接下来 t 行，每行两个整数 k 和 n ，代表一个询问。

【输出格式】

输出到文件 *cirno.out* 中。

输出共 t 行。如果对应的询问答案是可以，输出一行一个字符串 **aya**，否则输出一行一个字符串 **baka**。

【样例 1 输入】

```
1 2
2 1 999
3 2 999
```

【样例 1 输出】

```
1 aya
2 baka
```

【样例 1 解释】

999 不是 2 阶天才数，但是它是 1 阶天才数。

【样例 2】

见选手目录下的 *cirno/cirno2.in* 与 *cirno/cirno2.ans*。

【样例 3】

见选手目录下的 *cirno/cirno3.in* 与 *cirno/cirno3.ans*。

【子任务】

本题共有 20 个测试点，每个测试点 5 分。最终分数为所有测试点分数之和。

测试点	t	n_i	k_i
1	$= 1$	≤ 233	≤ 1
2 ~ 6	$= 1$	$\leq 10^3$	≤ 1
7, 8	$= 10$	$\leq 10^9$	≤ 1
9	$= 10^3$	$\leq 10^9$	≤ 1
10	$= 10^3$	$\leq 10^{18}$	≤ 1
11 ~ 20	$= 10^3$	$\leq 10^{18}$	≤ 10

对于全部数据，满足 $1 \leq t \leq 10^3$ ， $1 \leq n_i \leq 10^{18}$ ， $1 \leq k_i \leq 10$ 。

自学 (study)

【题目描述】

在 JOI 高中高一的第三个学期的 M 个星期的时间内, 有 N 门课, 编号为 $1 \sim N$ 。每个星期有 N 个课时, 第 i 个课时上课程 i 的一节课。

比太郎是一位高一学生。对于 $N \times M$ 个课时中的每一个, 他会选择如下行动中的一个:

- 行动 1: 比太郎去上课。如果他去上了课程 i 的一节课, 那么他对课程 i 的理解程度会增加 A_i 。
- 行动 2: 比太郎不去上课。他转而选择任意一门课, 并且自学选中的那门课。如果他选中了课程 i 进行了时长为一课时的自学, 那么他对课程 i 的理解程度会增加 B_i 。

一开始, 对每门课的理解程度都为 0。由于比太郎想要在课后练习算法竞赛, 他在非上课时间内不会学习。当第三个学期的所有课时结束后, 期末考就会举行。

比太郎不想挂科。所以他想要最大化在期末考时对每门课的理解程度的最小值。

给定学期的长度, 课程的数量, 以及对理解程度的提升数值, 请写一个程序计算在期末考时对每门课的理解程度的最小值的最大可能值。

【输入格式】

从文件 *study.in* 中读入数据。

第一行, 两个正整数 N, M 。

第二行, N 个正整数 A_1, A_2, \dots, A_N 。

第三行, N 个正整数 B_1, B_2, \dots, B_N 。

【输出格式】

输出到文件 *study.out* 中。

输出一行, 一个数, 表示在期末考时对每门课的理解程度的最小值的最大可能值。

【样例 1 输入】

```
1 3 3
2 19 4 5
3 2 6 2
```

【样例 1 输出】

1 18

【样例 1 解释】

举个例子，如果比太郎按如下方式学习，则他对课程 1, 2, 3 的理解程度将分别为 19, 18, 19。

- 第一周课程 1 的课：自学课程 2；
- 第一周课程 2 的课：自学课程 2；
- 第一周课程 3 的课：去上课程 3 的课；
- 第二周课程 1 的课：去上课程 1 的课；
- 第二周课程 2 的课：自学课程 3；
- 第二周课程 3 的课：去上课程 3 的课；
- 第三周课程 1 的课：自学课程 3；
- 第三周课程 2 的课：自学课程 2；
- 第三周课程 3 的课：去上课程 3 的课。

由于对每门课的最小的理解程度不能大于等于 19，输出 18。

这个样例满足子任务 3, 5 的限制。

【样例 2】

见选手目录下的 *study/study2.in* 与 *study/study2.ans*。

【样例 2 解释】

这个样例满足子任务 1, 3, 5 的限制。

【样例 3】

见选手目录下的 *study/study3.in* 与 *study/study3.ans*。

【样例 3 解释】

这个样例满足子任务 3, 5 的限制。

【样例 4】

见选手目录下的 *study/study4.in* 与 *study/study4.ans*。

【样例 4 解释】

这个样例满足子任务 2, 3, 4, 5 的限制。

【子任务】

本题采用捆绑测试。(只有子任务中的测试点全部通过该任务才有分)

对于 100% 的数据, $1 \leq N \leq 3 \times 10^5$, $1 \leq M \leq 10^9$, $1 \leq A_i, B_i \leq 10^9$ 。

- 子任务 1 (10 分): $M = 1$ 。
- 子任务 2 (25 分): $N \cdot M \leq 3 \times 10^5$, $A_i = B_i$ 。
- 子任务 3 (27 分): $N \cdot M \leq 3 \times 10^5$ 。
- 子任务 4 (29 分): $A_i = B_i$ 。
- 子任务 5 (9 分): 无特殊限制。

打 ACM 最快乐的就是滚榜读队名了 (submit)

【题目描述】

一场 ICPC 正式赛共 5 小时。

队伍的排名由通过题数与罚时决定。通过题数更多的队伍排名更靠前，若通过题数相同，则罚时更小的队伍排名更靠前。通过题数与罚时均相同的队伍排名相同。本题中可能出现队伍排名相同的情况，此时，认为先出现在提交记录中的队伍排名靠前。

罚时是由通过题目的时间和未通过提交的次数决定的。罚时为每一道题通过时比赛开始的分钟数之和，加上该题之前未通过提交的次数乘 20 分钟得到的。例如，某队在比赛进行 1:28:35 时通过了 G 题，在此之前共有 3 次未通过的提交，则 G 题对罚时的总贡献为 $88 + 3 \times 20 = 148$ 分钟。

需要注意的是，仅有通过的试题的未通过提交会被计算罚时。例如，某队在 I 题共有 14 次未通过的提交，但到比赛结束，该队都没有通过 I 题，则这 14 次未通过的提交不会被计算罚时。在某一题通过后，该队对这一题的任何提交（无论是否能够通过）都不会影响本题通过的结果和本题的罚时。

选手在比赛过程中可以随时提交某一道试题的代码，代码将被立即评测并返回结果 (*Accepted*, *TimeLimitExceeded*, *MemoryLimitExceeded*, *PresentationError*, *WrongAnswer*, *RuntimeError*)。其中，评测结果 *Accepted* 为通过，其他评测结果均为不通过。

在比赛进行的前四小时 (0:00:00 ~ 4:00:00)，每支队伍的提交均会在排行榜上反映出来。比赛的最后一小时 (4:00:01 ~ 5:00:00)，排行榜将被冻结 (封榜)，所有的提交在排行榜对应队伍对应试题上均显示为待判题 (提交的队伍知道评测结果)。

在比赛结束后，会进行紧张刺激的滚榜环节。滚榜嘉宾将按照封榜时的排行榜，依照从最后一名到第一名，先读出队伍队名，再按照从 A 题依次到最后一题的顺序，公布排行榜上该队“待判题”状态试题最终是否通过。

如果通过，所有队伍的排名将立即重新计算，显然，已经滚榜完成 (被滚榜嘉宾念过队名，且所有待判题状态的结果都已经揭晓) 的队伍排名不会有影响。若该队伍排名上升，则滚榜嘉宾立即开始下一支队伍的滚榜。因此，一支队伍的队名可能被滚榜嘉宾多次读出。

例如，某队队名为“围题”，在前四小时没有通过任何一题，封榜时排在最后一名。在封榜后，该队连续通过全部十三道题目。那么滚榜嘉宾有可能读到该队队名七八次。当然，当该队上升到第一名后，其排名不会再发生变化，即使揭晓的判题结果为通过，但其排名没有发生变化，滚榜嘉宾不会再次读出其队名。

现在给出某场 ICPC 完整的提交记录，请你依次输出滚榜嘉宾念出的队名。

一次提交记录都没有的队伍不会在排行榜上出现，也不会滚榜中被念到队名。

【输入格式】

从文件 `submit.in` 中读入数据。

输入的第一行为三个整数 n, m, K ，依次为该场 ICPC 试题数、该场 ICPC 队伍数、该场 ICPC 提交记录数。

接下来 K 行，每行为四个空格分隔的字符串，表达一条提交记录。第一个形如 $x:yy:zz$ ，代表该记录在比赛开始 x 小时 yy 分钟 zz 秒时提交。第二个字符串为一个 大写英文字母，代表试题的编号 (A, B, \dots)。第三个字符串为队名，保证队名不含空格。第四个字符串（可能含有空格，但为仅出现「题目描述」中的六种评测结果）为该评测记录的评测结果，具体字符串的含义见试题描述部分。

【输出格式】

输出到文件 `submit.out` 中。

输出若干行，为该滚榜嘉宾依次读到的队名。

【样例 1 输入】

```
1 2 2 4
2 0:00:01 A abc Wrong Answer
3 0:00:02 A abc Accepted
4 0:19:38 A bcd Accepted
5 4:18:22 B abc Accepted
```

【样例 1 输出】

```
1 abc
2 bcd
3 abc
```

【样例 1 解释】

在封榜前，队伍 abc 仅通过 A 题，且在第二秒的第一次正确提交之前有一次错误提交，因此罚时为 20 分钟；队伍 bcd 同样仅通过 A 题，且在 $0:19:38$ 的第一次正确提交之前没有错误提交，因此罚时为 19 分钟。

在封榜后，队伍 abc 通过了 B 题。

在滚榜环节开始，由于封榜后的提交未被揭晓，因此暂时认为队伍 abc 与 bcd 均只通过一题，且前者罚时较大，排名靠后。

依照从最后一名到第一名的原则，队伍 abc 的名字先被念到，并揭晓其在封榜后的提交的结果。其通过了 B 题，因此其通过题数被更新为 2，罚时同样被更新。同时，所有队伍的排名立即被重新计算。由于此时 abc 通过题目数量大于 bcd，因此其排名重新计算为第一名，而 bcd 成为最后一名第二名。

这之后，队伍 bcd 的名字被念到，由于其在封榜后没有提交，因此这时所有队伍的排名没有变化，滚榜嘉宾会进行其上一名队伍的滚榜。

最后，队伍 abc 的名字被念到，滚榜结束。

需要注意的是，在滚榜过程中是逐题揭晓提交。也就是说，如果一支队伍封榜后通过了多道题，在其进行滚榜过程中，只要按照从 A 题依次到最后一题的顺序，该队第一个“待判题”状态试题通过，后面的“待判题”同样暂时不会揭晓，而是立刻进行排名更新过程以及可能存在的更换另一支队伍进行滚榜的过程。

【样例 2】

见选手目录下的 *submit/submit2.in* 与 *submit/submit2.ans*。

【样例 3】

见选手目录下的 *submit/submit3.in* 与 *submit/submit3.ans*。

【子任务】

- 对于 30% 的数据， $n = 1$ ；
- 对于另外 10% 的数据， $m = 1$ ；
- 对于 100% 的数据， $1 \leq n \leq 20$ ， $1 \leq m \leq 1000$ ， $1 \leq K \leq 10^4$ ， $0 \leq x \leq 5$ ， $00 \leq yy < 60$ ， $00 \leq zz < 60$ ，且当 $x = 5$ 时保证 $yy = zz = 00$ 。

保证提交记录按照提交时间不降序给出，即先给出的提交记录提交时间不会晚于后给出的提交记录的提交时间，试题名称为大写字母 A ~ Z，队名均为长度不超过 50 的由小写字母组成的字符串，评测状态为试题中所给的 6 种之一。

找爸爸 (dnaseq)

【题目描述】

小 A 最近一直在找自己的爸爸，用什么办法呢，就是 DNA 比对。

小 A 有一套自己的 DNA 序列比较方法，其最终目标是最大化两个 DNA 序列的相似程度，具体步骤如下：

1. 给出两个 DNA 序列，第一个长度为 n ，第二个长度为 m 。
2. 在两个序列的任意位置插入任意多的空格，使得两个字符串长度相同
3. 逐位进行匹配，如果两个序列相同位置上的字符都不是空格，假设第一个是 x ，第二个是 y ，那么他们的相似程度由 $d(x, y)$ 定义。对于两个序列中任意一段极长的长度为 k 的连续空格，我们定义这段空格的相似程度为 $g(k) = -A - B(k - 1)$ 。

那么最终两个序列的相似程度就是所有的 $d(x, y)$ 加上所有的极长空格的相似程度之和。

现在小 A 通过某种奥妙重重的方式得到了小 B 的 DNA 序列中的一段，他想请你帮他算一下小 A 的 DNA 序列和小 B 的 DNA 序列的最大相似程度。

【输入格式】

从文件 *dnaseq.in* 中读入数据。

输入第 1 行一个字符串，表示小 A 的 DNA 序列。

输入第 2 行一个字符串，表示小 B 的 DNA 序列。

接下来 4 行，每行 4 个整数，用空格隔开，表示 d 数组，具体顺序如下所示。

```
1 d(A,A) d(A,T) d(A,G) d(A,C)
2 d(T,A) d(T,T) d(T,G) d(T,C)
3 d(G,A) d(G,T) d(G,G) d(G,C)
4 d(C,A) d(C,T) d(C,G) d(C,C)
```

最后一行两个用空格隔开的正整数 A, B ，意义如题中所述。

【输出格式】

输出到文件 *dnaseq.out* 中。

输出共一行，表示两个序列的最大相似程度。

【样例 1 输入】

```
1 ATGG
2 ATCC
```

```

3 5 -4 -4 -4
4 -4 5 -4 -4
5 -4 -4 5 -4
6 -4 -4 -4 5
7 2 1

```

【样例 1 输出】

```

1 4

```

【样例 1 解释】

首先，将序列补成如下形式（“-” 代表空格）

```

1 ATGG--
2 AT--CC

```

然后所有 $d(x, y)$ 的和为 $d(A, A) + d(T, T) = 10$

所有极长连续空格段的相似程度之和为 $g(2) + g(2) = -6$

总和为 4，可以验证，这是相似程度最大的情况。

【样例 2】

见选手目录下的 *dnaseq/dnaseq2.in* 与 *dnaseq/dnaseq2.ans*。

【样例 2 解释】

不加任何空格，直接两两匹配就是 5

【样例 3】

见选手目录下的 *dnaseq/dnaseq3.in* 与 *dnaseq/dnaseq3.ans*。

【子任务】

对于所有测试点，有 $0 < B < A \leq 1000, -1000 \leq d(x, y) \leq 1000, d(x, y) = d(y, x)$ ，序列只包含 $\{A, T, G, C\}$ 四种字符。

测试点编号	$n + m$ 的范围	特殊约定
1	$n = m = 1$	无特殊要求
2	$n + m \leq 15$	
3	$n + m \leq 300$	
4		
5	$n + m \leq 3000$	序列中只包含一种字符
6		无特殊要求
7		
8		
9		
10		