# T1

## 测试点 1~2

对于 $n$ 较小的情况，可以选择暴力枚举区间 $l, r$，然后暴力计算 $[l, r]$ 之间的 $a_i$ 的和。

时间复杂度为 $O(n^3)$。

## 测试点3~4

我们不用每次暴力统计 $[l, r]$ 内的 $a_i$ 和，而是可以记前缀和数组 $s_i = \sum_{i=1}^{i} a_i$，这样区间 $[l, r]$ 内 $a_i$ 的和就是 $s_r - s_{l-1}$。

时间复杂度为 $O(n^2)$。

## 测试点 5~6

因为 $a_i \geq 1$，所以满足条件的区间的长度一定小于等于 $m$，所以我们只用枚举长度小于等于 $m$ 的区间就可以。

时间复杂度为 $O(nm)$ 。

## 测试点 7~10

我们枚举区间的右端点 $r$，我们实际上就是在 $s_0, s_1, ..., s_{r-1}$ 中寻找是否存在一个数等于 $s_r - m$ 。

因为 $a_i > 0$，所以 $s_i$ 一定是递增的，所以可以通过二分查找的方式判断。

时间复杂度是 $O(n \log n)$ 。

更进一步，还可以考虑双指针。

固定一个左端点时，右端点不断右移直到区间和不小于 $m$。若此时区间和等于 $m$，就找到一个符合条件的区间。

由于继续让右端点右移是无意义的，所以此时让左端点右移。但右端点无需重置，继续从当前位置开始判断即可。

时间复杂度 $O(n)$。

# T2

## 测试点1~3

留给枚举所有情况的暴力做法。

## 测试点4~7

直接按 $a_i$ 从大到小排序，$a_i$ 最大的选手就是就是第 $1$ 名 ，最二大的选手就是就是第 $2$ 名 。。。

## 测试点8~10

锦标赛排序法就可以。

## 测试点11~13

只需要在上一档做法的基础上修改一下，维护最大值的时候多维护一个次大值。

然后也是锦标赛排序法。

## 测试点 14~16

其实题目中的每一轮比赛就是一个类似分治的过程，那么我们也可以通过分治来处理。

我们维护 `merge(int l,int r)` 表示对编号在 $l \sim r$ 的选手的决出组内的排名，并去除被淘汰的选手。

记 $mid = \frac{1}{2}(l+r)$，我们先要得到 $l \sim mid$ 和 $mid+1 \sim r$ 组内剩下的选手，然后合并起来，用 `sort` 排序，再去除掉第 $k$ 名之后的。

单次`sort`排序的时间复杂度是 $O(n \log n)$，分治一共有 $O(\log n)$ 层，所以总复杂度是 $O(n \log^2 n)$ 。

## 测试点 17~20

合并 $l \sim mid$ 和 $mid+1 \sim r$ 组内剩下的选手时我们使用了`sort`，事实上，我们可以通过归并排序的做法得到单次 $O(n)$ 的排序。

所以整体的复杂度就是 $O(n \log n)$ 。

# T3

## 测试点 1~4

对于 $b_i = 1$ 的情况，加乘运算就变成了加法运算，所以你只需要找到每个 $a_i$ 是什么，然后对于每一次修改，带来的影响就是 $y - a_i$ 。

## 测试点 5~10

每一次修改之后，我们模拟一次计算得到表达式的值。

计算一个后缀表达式的值的过程可以参考 [CSP-J2020] 表达式 ，应该是比较简单的。

## 测试点 11~14

应该注意到，这个表达式可以转化成一个树的结构，而且小 C 每次修改只会修改一个叶子节点的值，所以我们即可以暴力的从根向 $x$ 节点去遍历，然后对这一条路径上的点的值进行修改。

因为表达式随机生成，所以单次修改的时间复杂度的期望是 $O(\log n)$ 的，总复杂度就是 $O(n \log n)$ 。

## 测试点 15~20

实际上 $a *^b c = (a + c) \times b = ab + cb$。

所以我们可以在表达式建成的树上，把每一个 $b_i$ 往其子树推，最后就能将式子化成 $ans = d_1a_1 + d_2a_2 + \ldots + d_na_n$。

例如样例中的：

$$
\begin{aligned}
&(a_1 *^4 a_2) *^2 (a_3 *^3 (a_4 *^1 a_5)) \\
=&((a_1 + a_2) \times 4 + (a_3 + (a_4 + a_5) \times 1) \times 3) \times 2 \\
=&(a_1 + a_2) \times 8 + (a_3 + (a_4 + a_5) \times 1) \times 6 \\
=&8a_1 + 8a_2 + 6a_3 + 6a_4 + 6a_5
\end{aligned}
$$

所以我们就可以很简单的计算 $a_x$ 变成 $y$ 带来的改变。

时间复杂度是 $O(n + m)$。

# T4

## 测试点 1~3

给暴力搜索留的分。

## 测试点4~7

留给暴力讨论的部分分。

因为 $m=3$，情况很少，对于某一天你可能有这几种情况：

- 对于`001`，`111` 等情况，可以退 $1$ 门课，学习时间少 $1$ 小时。
- 而对于 `101` 的情况，可以选择退 $1$ 节课，学习时间少 $2$ 小时。

贪心的先选择少 $2$ 小时的情况，剩下的只能选少 $1$ 小时的情况。

## 测试点14~16

其实就是对上一个做法的扩展。

对于每一天，可以选择退 $1$ 门课，少学习 $r-l$ 个小时。

同样贪心的选择减少时间最多的课，最后如果还能退课，那么都只能少 $1$ 个小时。

## 测试点8~11

进一步扩展上面的做法。

对于一天，我们可以考虑每一节课选择退或者不退，暴力得到 $c_{i,j}$ ，表示第 $i$ 天退 $j$ 门课的情况下的学习时间。

然后就是一个分组背包。

时间复杂度为 $O(n2^m+nmk)$ 。

## 测试点12~13 17~20

我们不用暴力的考虑每一节课选择退或者不退，而只需要考虑这一天中最早的课和最晚的课是什么就可以（中间的课退了也不会减少学习的时间）。

所以就可以枚举最早的课 $j$，最晚的课 $k$，计算出一天退多少课，学习时间为 $k-j+1$ 小时。

这样就可以 $O(nm^2)$ 的时间复杂度求出 $c_{i,j}$ 。