团队竞技

我们必然是选择 X,Y,Z 最大的三个人最优，但是如果其中有一个人在两项中作为的最大值，则它必然不可能作为合法的答案出现，我们就可以将其删除，这样就变成了一个子问题。

不断重复删除的过程，使用优先队列维护当前最大值即可，时间复杂度 $O(n\log n)$。

甲虫

记 $val(l,r)=\max\limits_{i=l}^{r}S_i$。我们有 $f_i=\max\limits_{j=i-k}^{i-1}(f_j+val(j+1,i))$。

根据贪心，只有 $j=i-k$ 和满足 $val(j+1,i)\neq val(j,i)$ 的 j 才有可能去进行转移，而这些点可以使用单调栈维护。

发现就是需要一个支持如下操作的数据结构：维护一个序列，支持从开头和末尾删除，末尾加入，查询序列内的最大值。

使用可删堆可以做到 $O(n\log n)$。

但这个东西是可以做到 $O(n)$ 的。

考虑维护两个栈，相当于将这个双端队列分成两个部分。同时每一个元素同时维护前缀最大值，此时维护头尾的插入和删除以及查询最大值都是容易的。

但是如果有一个队列被删空了，考虑直接 $O(siz)$ 暴力重构，由于在进行这次重构之前，至少会进行 $O(siz)$ 次加入或者删除，因此总时间复杂度为 $O(n)$。

两种货币

假设需要使用金币，根据贪心，那么必然是在需要银币数量最多的那些检查站中使用，因此用银币的将会是路径上银币代价前 k 小的检查站。

如果只有单个询问，考虑二分答案 k，只有在全局 c_j 前 k 小的那些检查站才使用银币，可以通过二分来得到 k 至多为多少的时候，银币的数量还是足够的。此时可以计算还剩多少个站点需要用金币支付，依次来得到答案。

对于这个值的查询，可以使用重链剖分加线段树维护，时间复杂度 $O(q\log^2 n\log m)$。

长途巴士

将所有人按照 D_i 排序之后重标号，则每一次下车的人必然是编号的连续区间，而且这个区间内的最后一个人 i 必须要满足存在一个服务站 S 使得 $D_i<(S\bmod T)<D_{i+1}$。

同时由于司机不能下车，所以对应的区间必然是 $[l,r]$，而不会出现 $[1,r]\cup[l,m]$ 的结构。

由此，我们可以设计 DP f_i 表示处理了前 i 个人的最小代价。

我们考虑每一个人是被赶下车还是留在车上。

如果留在车上就有 $f_i\leftarrow f_{i-1}+W(\left\lfloor\dfrac{X}{T}\right\rfloor+[X\bmod T>D_i])$。

否则，我们枚举这一次下车的区间 $[j+1,i]$，$f_i \leftarrow \min\limits_{j<i}(f_j+(sum_i-sum_j)+W(i-j)sd_i)$。其中 sum_i 为前 i 个人的 C_i 的和，sd_i 为 $\min\limits_{D_i<S_j \bmod T<D_{i+1}} \left\lfloor \dfrac{S}{T} \right\rfloor$，表示将 i 赶下车之前需要喝水的次数。

对于前者，可以直接暴力转移，对于后者，其等于 $\min\limits_{j<i}(f_j-sum_j-Wjsd_i)+sum_i+Wisd_i$，可以使用斜率优化，具体的就是二分栈。