The 2022 Hangzhou Normal U Summer Trials

Tutorial

Hangzhou Normal U ACM/ICPC Team

2022 年 05 月 14 日

1. Solution

Problem A Hello, ACMer!

- 签到题，输出字符串中有几个"hznu"

Problem A Hello, ACMer!

- 签到题，输出字符串中有几个"hznu"
- 注意 string.size() 为无符号整数

Problem B New String

- 签到题，重写一个 *cmp* 函数传入 *sort* 就可以了

Problem C Check Problems

- 首先题目保证 $a_i < a_{i+1}, a_{i+1} - a_i < a_{i+2} - a_{i+1}$

Problem C Check Problems

- 首先题目保证 $a_i < a_{i+1}, a_{i+1} - a_i < a_{i+2} - a_{i+1}$
- 考虑 t 分钟每个人的实际贡献，若 $a_{i+1} - a_i > t$ 那么第 i 个人的贡献为 t, 否则第 i 个人的贡献为 $a_{i+1} - a_i$.

Problem C Check Problems

- 首先题目保证 $a_i < a_{i+1}, a_{i+1} - a_i < a_{i+2} - a_{i+1}$
- 考虑 t 分钟每个人的实际贡献，若 $a_{i+1} - a_i > t$ 那么第 i 个人的贡献为 t，否则第 i 个人的贡献为 $a_{i+1} - a_i$.
- 然后题目保证了 $a_{i+1} - a_i < a_{i+2} - a_{i+1}$，这个条件告诉我们两个人之间的差是单调的，那么我们就可以二分出有 x 个人的贡献 $< t$

Problem C Check Problems

- 首先题目保证 $a_i < a_{i+1}, a_{i+1} - a_i < a_{i+2} - a_{i+1}$
- 考虑 t 分钟每个人的实际贡献，若 $a_{i+1} - a_i > t$ 那么第 i 个人的贡献为 t, 否则第 i 个人的贡献为 $a_{i+1} - a_i$.
- 然后题目保证了 $a_{i+1} - a_i < a_{i+2} - a_{i+1}$, 这个条件告诉我们两个人之间的差是单调的，那么我们就可以二分出有 x 个人的贡献 $< t$
- 答案就为 $a[x+1] - a[1] + (n-x) * t$

Problem D Tree Problem

- 经典题目。一种典型的做法是在 *dfs* 中预处理出所有答案，然后直接根据询问输出

Problem D Tree Problem

- 经典题目。一种典型的做法是在 *dfs* 中预处理出所有答案，然后直接根据询问输出
- 遍历完一条边后，固定路径中的一个端点在当前边上，另一个点在之前遍历过的所有点里选

Problem D Tree Problem

- 经典题目。一种典型的做法是在 *dfs* 中预处理出所有答案，然后直接根据询问输出
- 遍历完一条边后，固定路径中的一个端点在当前边上，另一个点在之前遍历过的所有点里选
- 实际上就是在 *dfs* 子树大小的过程中多一步计算，时间复杂度 $O(n)$

Problem D Tree Problem

- 经典题目。一种典型的做法是在 *dfs* 中预处理出所有答案，然后直接根据询问输出
- 遍历完一条边后，固定路径中的一个端点在当前边上，另一个点在之前遍历过的所有点里选
- 实际上就是在 *dfs* 子树大小的过程中多一步计算，时间复杂度 $O(n)$
- 注意算上根到当前节点的那条边的贡献

Problem D Tree Problem

- 经典题目。一种典型的做法是在 *dfs* 中预处理出所有答案，然后直接根据询问输出
- 遍历完一条边后，固定路径中的一个端点在当前边上，另一个点在之前遍历过的所有点里选
- 实际上就是在 *dfs* 子树大小的过程中多一步计算，时间复杂度 $O(n)$
- 注意算上根到当前节点的那条边的贡献
- 如果没有提前计算出答案，请注意记录询问过的答案，不然遇到菊花图复杂度就不对了

Problem E Easy Problem

- 这是一个广搜问题

Problem E Easy Problem

- 这是一个广搜问题
- $vis[ax][ay][bx][by]$ 代表一个状态，他的含义为到达玩家 a 在 (ax, ay) 这个点，玩家 b 在 (bx, by) 这个点时的状态的最短步数

Problem E Easy Problem

- 这是一个广搜问题
- $vis[ax][ay][bx][by]$ 代表一个状态，他的含义为到达玩家 a 在 (ax, ay) 这个点，玩家 b 在 (bx, by) 这个点时的状态的最短步数
- 那么对于当前状态，我们可以同时控制玩家 a, b 往上下左右移动，假如移动后的新的状态非零，那代表有更短的路径到达该点

Problem E Easy Problem

- 这是一个广搜问题
- $vis[ax][ay][bx][by]$ 代表一个状态，他的含义为到达玩家 a 在 (ax, ay) 这个点，玩家 b 在 (bx, by) 这个点时的状态的最短步数
- 那么对于当前状态，我们可以同时控制玩家 a, b 往上下左右移动，假如移动后的新的状态非零，那代表有更短的路径到达该点
- 那么我们不移动，否则新的状态从上一个状态步数加一进行转移，当 $(ax, ay) = (bx, by)$ 时代表到达同一点，广搜结束

Problem F Subarrays

- 求满足 $sum[l, r] \equiv 0 \bmod k, 1 \le l \le r \le n$ 的区间和的个数

Problem F Subarrays

- 求满足 $sum[l, r] \equiv 0 \bmod k, 1 \le l \le r \le n$ 的区间和的个数
- 显然 $sum[l, r] \bmod k = (sum[r] - sum[l-1]) \bmod k = 0$

Problem F Subarrays

- 求满足 $sum[l, r] \equiv 0 \bmod k, 1 \leq l \leq r \leq n$ 的区间和的个数
- 显然 $sum[l, r] \bmod k = (sum[r] - sum[l-1]) \bmod k = 0$
- 可以容易推出 $sum[l-1] \equiv sum[r] \bmod k$

Problem F Subarrays

- 求满足 $sum[l,r] \equiv 0 \bmod k, 1 \le l \le r \le n$ 的区间和的个数
- 显然 $sum[l,r] \bmod k = (sum[r] - sum[l-1]) \bmod k = 0$
- 可以容易推出 $sum[l-1] \equiv sum[r] \bmod k$
- 所以题目就变成了，对于每个位置 i 求小于 i 中的前缀和有多少模 k 值相同，这个问题可以用 map 去维护，时间为 $O(nlogn)$

Problem G Ganyu Segment Tree

- 考虑如何判断能否整除

Problem G Ganyu Segment Tree

- 考虑如何判断能否整除
- 对 x 分解质因数，区间内所有元素对应的质因数的最小值应该不小于 x 中的数量

Problem G Ganyu Segment Tree

- 考虑如何判断能否整除
- 对 x 分解质因数，区间内所有元素对应的质因数的最小值应该不小于 x 中的数量
- 30 以内的质因数只有 10 个，用线段树维护区间最小值即可，乘法相当于区间加，除法相当于区间减

Problem G Ganyu Segment Tree

- 考虑锁定和解锁的影响

Problem G Ganyu Segment Tree

- 考虑锁定和解锁的影响
- 维护区间锁定和解锁两种最小值，设解锁初始值为 1，锁定初始值为 inf，区间加减只影响解锁值

Problem G Ganyu Segment Tree

- 考虑锁定和解锁的影响
- 维护区间锁定和解锁两种最小值，设解锁初始值为 1，锁定初始值为 inf，区间加减只影响解锁值
- 维护区间最小值时，对两种最小值取小

Problem G Ganyu Segment Tree

- 考虑锁定和解锁的影响
- 维护区间锁定和解锁两种最小值，设解锁初始值为 1，锁定初始值为 *inf*，区间加减只影响解锁值
- 维护区间最小值时，对两种最小值取小
- 遇到状态转换，则交换两种最小值

Problem G Ganyu Segment Tree

- 考虑锁定和解锁的影响
- 维护区间锁定和解锁两种最小值，设解锁初始值为 1，锁定初始值为 inf，区间加减只影响解锁值
- 维护区间最小值时，对两种最小值取小
- 遇到状态转换，则交换两种最小值
- 我们发现当 inf 和加减的值不在一个数量级上，相当于没有影响

Problem G Ganyu Segment Tree

- 考虑锁定和解锁的影响
- 维护区间锁定和解锁两种最小值，设解锁初始值为 1，锁定初始值为 inf，区间加减只影响解锁值
- 维护区间最小值时，对两种最小值取小
- 遇到状态转换，则交换两种最小值
- 我们发现当 inf 和加减的值不在一个数量级上，相当于没有影响
- 时间复杂度 $O(mlogn)$

Problem G Ganyu Segment Tree

- 彩蛋

Problem G Ganyu Segment Tree

- 彩蛋
- 这题是原来题目的 *validator*，但是发现比原题更有意思，所以直接出了这个题

Problem H Optimal Biking Strategy

- 先讨论一种简单的情况，$k = 1$ 时

Problem H Optimal Biking Strategy

- 先讨论一种简单的情况，$k = 1$ 时
- 题目要求总步行距离最小，等价于总骑车距离最大。那么，如果有许多个点可以转移到 i, 我们总是取最靠前的一个点 j, 从 j 骑到 i, 而对于任何介于 j 和 i 中间的点 x, 从 x 骑到 i 总是不如 j 的，因为这样骑车的距离更小了

Problem H Optimal Biking Strategy

- 先讨论一种简单的情况，$k = 1$ 时
- 题目要求总步行距离最小，等价于总骑车距离最大。那么，如果有许多个点可以转移到 i, 我们总是取最靠前的一个点 j, 从 j 骑到 i, 而对于任何介于 j 和 i 中间的点 x, 从 x 骑到 i 总是不如 j 的，因为这样骑车的距离更小了
- 基于此，我们可以维护一个指针 l, 代表到 i 距离不超过 s 的的点。

Problem H Optimal Biking Strategy

- 设 $f[i,0/1]$ 表示到达第 i 个点的最小步行距离，$j=1$ 表示已经花费了 1 元骑车，$j=0$ 表示没有

Problem H Optimal Biking Strategy

- 设 $f[i,0/1]$ 表示到达第 i 个点的最小步行距离，$j=1$ 表示已经花费了 1 元骑车，$j=0$ 表示没有
- 则有如下状态转移方程

Problem H Optimal Biking Strategy

- 设 $f[i,0/1]$ 表示到达第 i 个点的最小步行距离，$j=1$ 表示已经花费了 1 元骑车，$j=0$ 表示没有
- 则有如下状态转移方程
- $f[i,0]=f[i-1,0]+a[i]-a[i-1]$

Problem H Optimal Biking Strategy

- 设 $f[i,0/1]$ 表示到达第 i 个点的最小步行距离，$j=1$ 表示已经花费了 1 元骑车，$j=0$ 表示没有
- 则有如下状态转移方程
- $f[i,0]=f[i-1,0]+a[i]-a[i-1]$
- $f[i,1]=min(f[i-1,1]+a[i]-a[i-1],f[l,0])$

Problem H Optimal Biking Strategy

- 设 $f[i,0/1]$ 表示到达第 i 个点的最小步行距离，$j=1$ 表示已经花费了 1 元骑车，$j=0$ 表示没有
- 则有如下状态转移方程
- $f[i,0]=f[i-1,0]+a[i]-a[i-1]$
- $f[i,1]=min(f[i-1,1]+a[i]-a[i-1],f[l,0])$
- 对于第二个方程的解释是，到达该点且骑过车有两种可能。1 是最后一段路是骑车的，2 是在 i 点之前就已经骑过车了，最后一段路是步行的

Problem H Optimal Biking Strategy

- 最后

Problem H Optimal Biking Strategy

- 最后
- $ans = \min_{1 \le i \le n}(f[i, 1] + a[p] - a[i])$

Problem H Optimal Biking Strategy

- 最后
- $ans = \min_{1 \le i \le n}(f[i, 1] + a[p] - a[i])$
- 即得到最终答案，其中 p 表示终点

Problem H Optimal Biking Strategy

- 最后
- $ans = \min_{1 \le i \le n}(f[i, 1] + a[p] - a[i])$
- 即得到最终答案，其中 p 表示终点
- 而当 $k > 1$ 时，采用同样的方法，我们只要维护 k 个指针

Problem H Optimal Biking Strategy

- 设 $l[x]$ 表示到 i 距离不超过 $x*s$ 的最靠前的点,$f[i,j]$ 表示到达第 i 个点，花了 j 元钱, 骑行后的最小步行距离

Problem H Optimal Biking Strategy

- 设 $l[x]$ 表示到 i 距离不超过 $x * s$ 的最靠前的点,$f[i,j]$ 表示到达第 i 个点，花了 j 元钱, 骑行后的最小步行距离
- $f[i,j] = \min_{1 \le x \le k}\{f[i-1,j] + a[i] - a[i-1], f[l[x], j-x]\}$

Problem H Optimal Biking Strategy

- 设 $l[x]$ 表示到 i 距离不超过 $x*s$ 的最靠前的点,$f[i,j]$ 表示到达第 i 个点，花了 j 元钱, 骑行后的最小步行距离
- $f[i,j] = \min_{1\le x\le k}\{f[i-1,j] + a[i] - a[i-1], f[l[x], j-x]\}$
- $ans = \min_{1\le i\le n}(f[i,k] + a[p] - a[i])$

Problem H Optimal Biking Strategy

- 设 $l[x]$ 表示到 i 距离不超过 $x*s$ 的最靠前的点,$f[i,j]$ 表示到达第 i 个点，花了 j 元钱, 骑行后的最小步行距离
- $f[i,j] = \min_{1\leq x\leq k}\{f[i-1,j]+a[i]-a[i-1], f[l[x], j-x]\}$
- $ans = \min_{1\leq i\leq n}(f[i,k]+a[p]-a[i])$
- 即为最终答案

Problem H Optimal Biking Strategy

- 设 $l[x]$ 表示到 i 距离不超过 $x*s$ 的最靠前的点,$f[i,j]$ 表示到达第 i 个点，花了 j 元钱, 骑行后的最小步行距离
- $f[i,j] = \min_{1 \le x \le k}\{f[i-1,j] + a[i] - a[i-1], f[l[x], j-x]\}$
- $ans = \min_{1 \le i \le n}(f[i,k] + a[p] - a[i])$
- 即为最终答案
- 数据设置在二分能过的范围

Problem I IHI's Homework

- 对于 $\sum_{i=1}^{n} x_i = s \ (x_i \geq 1)$ 的解的情况数用隔板法:$\binom{s-1}{n-1}$

Problem I IHI's Homework

- 对于 $\sum_{i=1}^{n} x_i = s\ (x_i \geq 1)$ 的解的情况数用隔板法:$\binom{s-1}{n-1}$
- 对于 $\forall x_i \geq a_i$ 这个条件，可以令 $y_i = x_i - a_i$，这样 $\forall y_i \geq 0$

Problem I IHI's Homework

- 对于 $\sum_{i=1}^{n} x_i = s\ (x_i \geq 1)$ 的解的情况数用隔板法: $\binom{s-1}{n-1}$
- 对于 $\forall x_i \geq a_i$ 这个条件, 可以令 $y_i = x_i - a_i$, 这样 $\forall y_i \geq 0$
- 即 $\sum_{i=1}^{n} y_i = \sum_{i=1}^{n} x_i - \sum_{i=1}^{n} a_i = s - \sum_{i=1}^{n} a_i$

Problem I IHI's Homework

- 对于 $\sum_{i=1}^{n} x_i = s \ (x_i \geq 1)$ 的解的情况数用隔板法:$\binom{s-1}{n-1}$
- 对于 $\forall x_i \geq a_i$ 这个条件，可以令 $y_i = x_i - a_i$，这样 $\forall y_i \geq 0$
- 即 $\sum_{i=1}^{n} y_i = \sum_{i=1}^{n} x_i - \sum_{i=1}^{n} a_i = s - \sum_{i=1}^{n} a_i$
- 再考虑 $\leq$ 的影响

Problem I IHI's Homework

- 我们将不等式转换为等式 $\sum_{i=1}^{n} y_i + z = s - \sum_{i=1}^{n} a_i (z \geq 0)$

Problem I IHI's Homework

- 我们将不等式转换为等式 $\sum_{i=1}^{n} y_i + z = s - \sum_{i=1}^{n} a_i (z \geq 0)$
- 左右两边同时加上 $n + 1$

Problem I IHI's Homework

- 我们将不等式转换为等式 $\sum_{i=1}^{n} y_i + z = s - \sum_{i=1}^{n} a_i (z \geq 0)$
- 左右两边同时加上 $n+1$
- $\sum_{i=1}^{n} Y_i + Z = s - \sum_{i=1}^{n} a_i + n + 1$

Problem I IHI's Homework

- 我们将不等式转换为等式 $\sum_{i=1}^{n} y_i + z = s - \sum_{i=1}^{n} a_i (z \geq 0)$
- 左右两边同时加上 $n+1$
- $\sum_{i=1}^{n} Y_i + Z = s - \sum_{i=1}^{n} a_i + n + 1$
- 所以解的数量是 $\binom{s-\sum_{i=1}^{n} a_i + n}{n}$

Problem J IHI's Magic String

- 首先拿到题目之后发现正向做很简单，我们只需要考虑模拟这个过程即可，但是这个时间复杂度是不允许的

Problem J IHI's Magic String

- 首先拿到题目之后发现正向做很简单，我们只需要考虑模拟这个过程即可，但是这个时间复杂度是不允许的
- 我们可以思考一下给定的操作的性质，假设只有步骤 1 和步骤 2 显而易见就是一道很简单的题目，但是对于步骤 3 来说我们思考这样的一个字符转换过程

Problem J IHI's Magic String

- 首先拿到题目之后发现正向做很简单，我们只需要考虑模拟这个过程即可，但是这个时间复杂度是不允许的
- 我们可以思考一下给定的操作的性质，假设只有步骤 1 和步骤 2 显而易见就是一道很简单的题目，但是对于步骤 3 来说我们思考这样的一个字符转换过程
- 假设我有两个 3 操作，先将 x 变为 y，再将 y 变为 z, 那么其实最终我们是让 x 变成了 z, 那么其实我们可以看到后面的操作对于前面的操作是由一定的覆盖性质的。

Problem J IHI's Magic String

- 首先拿到题目之后发现正向做很简单，我们只需要考虑模拟这个过程即可，但是这个时间复杂度是不允许的
- 我们可以思考一下给定的操作的性质，假设只有步骤 1 和步骤 2 显而易见就是一道很简单的题目，但是对于步骤 3 来说我们思考这样的一个字符转换过程
- 假设我有两个 3 操作，先将 x 变为 y，再将 y 变为 z, 那么其实最终我们是让 x 变成了 z, 那么其实我们可以看到后面的操作对于前面的操作是由一定的覆盖性质的。
- 那么我们可以从后面往前面做，我们设定 $f[x]$ 是这个字符最终实际放进字符串的时候的字符，对于每个 $f[x]$ 初始值我们都可以赋值给他的本身。

Problem J IHI's Magic String

- 那么对于一个 3 操作，假设把 x 变成 y, 因为我们从后面往前面做并且要考虑到 y 后面是否还会变化，因此我们可以使得 $f[x] = f[y]$，实际放进这个字符只需要放进去 $f[x]$ 即可

Problem J IHI's Magic String

- 那么对于一个 3 操作，假设把 x 变成 y，因为我们从后面往前面做并且要考虑到 y 后面是否还会变化，因此我们可以使得 $f[x]=f[y]$，实际放进这个字符只需要放进去 $f[x]$ 即可
- 如果我们从后面往前面做，那么对于 2 操作怎么实现呢？

Problem J IHI's Magic String

- 那么对于一个 3 操作，假设把 x 变成 y, 因为我们从后面往前面做并且要考虑到 y 后面是否还会变化，因此我们可以使得 $f[x] = f[y]$，实际放进这个字符只需要放进去 $f[x]$ 即可
- 如果我们从后面往前面做，那么对于 2 操作怎么实现呢？
- 那对于这个问题我们可以开一个 vis 数组来表示这个位置可不可以使用 2 操作，然后我们再从前往后处理每个位置的操作 1 和操作 2，预处理一遍即可

Problem K Klee’s Wonderful Adventure

- 因为点和点之间的距离和速度都已经确定，那么我们可以算出点到点之间的时间。

Problem K Klee's Wonderful Adventure

- 因为点和点之间的距离和速度都已经确定，那么我们可以算出点到点之间的时间。
- 算出时间之后我们思考一下给定起始点和终点，并且给定若干条路，可以想到采用最原始的 *dijkstra* 跑，时间复杂度为 $O(n^2)$

Problem K Klee's Wonderful Adventure

- 因为点和点之间的距离和速度都已经确定，那么我们可以算出点到点之间的时间。
- 算出时间之后我们思考一下给定起始点和终点，并且给定若干条路，可以想到采用最原始的 *dijkstra* 跑，时间复杂度为 $O(n^2)$
- 没有卡建边后跑堆优化的 *dijkstra*

Thank You!