

数据结构初步

Claris

Hangzhou Dianzi University

2018 年 2 月 10 日

Overview

- 本节课介绍几种数据结构以及一些技巧：

Overview

- 本节课介绍几种数据结构以及一些技巧：
- 全局最大值的维护。

Overview

- 本节课介绍几种数据结构以及一些技巧：
- 全局最大值的维护。
- 单调栈/单调队列。

Overview

- 本节课介绍几种数据结构以及一些技巧：
- 全局最大值的维护。
- 单调栈/单调队列。
- 链表。

Overview

- 本节课介绍几种数据结构以及一些技巧：
- 全局最大值的维护。
- 单调栈/单调队列。
- 链表。
- 寻找不变量与变化点。

Overview

- 本节课介绍几种数据结构以及一些技巧：
- 全局最大值的维护。
- 单调栈/单调队列。
- 链表。
- 寻找不变量与变化点。
- 并查集。

Overview

- 本节课介绍几种数据结构以及一些技巧：
- 全局最大值的维护。
- 单调栈/单调队列。
- 链表。
- 寻找不变量与变化点。
- 并查集。
- 时间倒流法。

Rage Minimum Query

维护一个序列 a_1, a_2, \dots, a_n , m 次修改。

Rage Minimum Query

维护一个序列 a_1, a_2, \dots, a_n , m 次修改。

每次指定一个 x , 将 a_x 修改为 y 。

Rage Minimum Query

维护一个序列 a_1, a_2, \dots, a_n , m 次修改。

每次指定一个 x , 将 a_x 修改为 y 。

求每次修改之后的全局最小值。

Rage Minimum Query

维护一个序列 a_1, a_2, \dots, a_n , m 次修改。

每次指定一个 x , 将 a_x 修改为 y 。

求每次修改之后的全局最小值。

数据在指定范围内完全随机生成。

Rage Minimum Query

维护一个序列 a_1, a_2, \dots, a_n , m 次修改。

每次指定一个 x , 将 a_x 修改为 y 。

求每次修改之后的全局最小值。

数据在指定范围内完全随机生成。

- $n \leq 10^7$ 。

Rage Minimum Query

维护一个序列 a_1, a_2, \dots, a_n , m 次修改。

每次指定一个 x , 将 a_x 修改为 y 。

求每次修改之后的全局最小值。

数据在指定范围内完全随机生成。

- $n \leq 10^7$ 。
- $m \leq 5 \times 10^7$ 。

Rage Minimum Query

维护一个序列 a_1, a_2, \dots, a_n , m 次修改。

每次指定一个 x , 将 a_x 修改为 y 。

求每次修改之后的全局最小值。

数据在指定范围内完全随机生成。

- $n \leq 10^7$ 。
- $m \leq 5 \times 10^7$ 。
- $0 \leq a_i < 2^{32}$ 。

Rage Minimum Query

维护一个序列 a_1, a_2, \dots, a_n , m 次修改。

每次指定一个 x , 将 a_x 修改为 y 。

求每次修改之后的全局最小值。

数据在指定范围内完全随机生成。

- $n \leq 10^7$ 。
- $m \leq 5 \times 10^7$ 。
- $0 \leq a_i < 2^{32}$ 。
- Source : XVII Open Cup named after E.V. Pankratiev. GP of Moscow Workshops

Solution

- 若 $y \leq a_x$, 那么显然 $ans = \min(ans, y)$ 即可。

Solution

- 若 $y \leq a_x$, 那么显然 $ans = \min(ans, y)$ 即可。
- 否则当且仅当 a_x 之前是最小值时答案才可能会改变。

Solution

- 若 $y \leq a_x$, 那么显然 $ans = \min(ans, y)$ 即可。
- 否则当且仅当 a_x 之前是最小值时答案才可能会改变。
- a_x 是最小值的概率为 $\frac{1}{n}$, 暴力重算最小值即可。

Solution

- 若 $y \leq a_x$, 那么显然 $ans = \min(ans, y)$ 即可。
- 否则当且仅当 a_x 之前是最小值时答案才可能会改变。
- a_x 是最小值的概率为 $\frac{1}{n}$, 暴力重算最小值即可。
- 时间复杂度 $O(n + m)$ 。

+1 Maximum Query

维护一个序列 a_1, a_2, \dots, a_n , m 次修改。

+1 Maximum Query

维护一个序列 a_1, a_2, \dots, a_n , m 次修改。

每次指定一个 x , 将 a_x 增加 1 或者减少 1。

+1 Maximum Query

维护一个序列 a_1, a_2, \dots, a_n , m 次修改。

每次指定一个 x , 将 a_x 增加 1 或者减少 1。

求每次修改之后的全局最大值。

+1 Maximum Query

维护一个序列 a_1, a_2, \dots, a_n , m 次修改。

每次指定一个 x , 将 a_x 增加 1 或者减少 1。

求每次修改之后的全局最大值。

- $n \leq 10^7$ 。

+1 Maximum Query

维护一个序列 a_1, a_2, \dots, a_n , m 次修改。

每次指定一个 x , 将 a_x 增加 1 或者减少 1。

求每次修改之后的全局最大值。

- $n \leq 10^7$ 。
- $m \leq 10^7$ 。

+1 Maximum Query

维护一个序列 a_1, a_2, \dots, a_n , m 次修改。

每次指定一个 x , 将 a_x 增加 1 或者减少 1。

求每次修改之后的全局最大值。

- $n \leq 10^7$ 。
- $m \leq 10^7$ 。
- $0 \leq a_i \leq n$ 。

Solution

- 若是增加，那么还是同理， $ans = \max(ans, a_x + 1)$ 。

Solution

- 若是增加，那么还是同理， $ans = \max(ans, a_x + 1)$ 。
- 否则最大值最多减少 1。

Solution

- 若是增加，那么还是同理， $ans = \max(ans, a_x + 1)$ 。
- 否则最大值最多减少 1。
- 记录每个数字出现次数，判断当前还剩几个最大值即可。

Solution

- 若是增加，那么还是同理， $ans = \max(ans, a_x + 1)$ 。
- 否则最大值最多减少 1。
- 记录每个数字出现次数，判断当前还剩几个最大值即可。
- 时间复杂度 $O(n + m)$ 。

+1 Maximum Query 2

维护一个序列 a_1, a_2, \dots, a_n , m 次修改。

+1 Maximum Query 2

维护一个序列 a_1, a_2, \dots, a_n , m 次修改。

每次指定一个 x , 将 a_x 增加 1 或者减少 1。

+1 Maximum Query 2

维护一个序列 a_1, a_2, \dots, a_n , m 次修改。

每次指定一个 x , 将 a_x 增加 1 或者减少 1。

求每次修改之后的全局最大值。

+1 Maximum Query 2

维护一个序列 a_1, a_2, \dots, a_n , m 次修改。

每次指定一个 x , 将 a_x 增加 1 或者减少 1。

求每次修改之后的全局最大值。

- $n \leq 10^7$ 。

+1 Maximum Query 2

维护一个序列 a_1, a_2, \dots, a_n , m 次修改。

每次指定一个 x , 将 a_x 增加 1 或者减少 1。

求每次修改之后的全局最大值。

- $n \leq 10^7$ 。
- $m \leq 10^7$ 。

+1 Maximum Query 2

维护一个序列 a_1, a_2, \dots, a_n , m 次修改。

每次指定一个 x , 将 a_x 增加 1 或者减少 1。

求每次修改之后的全局最大值。

- $n \leq 10^7$ 。
- $m \leq 10^7$ 。
- $0 \leq a_i \leq 10^9$ 。

Solution

- 数字范围太大，不能直接记录出现次数。

Solution

- 数字范围太大，不能直接记录出现次数。
- 注意到只有与初始最大值差值不超过 m 的数才有用。

Solution

- 数字范围太大，不能直接记录出现次数。
- 注意到只有与初始最大值差值不超过 m 的数才有用。
- 只记录这 $O(m)$ 个数的个数即可。

Solution

- 数字范围太大，不能直接记录出现次数。
- 注意到只有与初始最大值差值不超过 m 的数才有用。
- 只记录这 $O(m)$ 个数的个数即可。
- 时间复杂度 $O(n + m)$ 。

滑窗最大值

给定一个序列 a_1, a_2, \dots, a_n , 求每个长度为 k 的子区间的区间最大值。

滑窗最大值

给定一个序列 a_1, a_2, \dots, a_n , 求每个长度为 k 的子区间的区间最大值。

- $1 \leq k \leq n \leq 10^7$ 。

Solution

- 从左往右考虑每个数。

Solution

- 从左往右考虑每个数。
- 若 $a_i \leq a_j$ 且 $i < j$, 那么 i 永远不会有比 j 优。

Solution

- 从左往右考虑每个数。
- 若 $a_i \leq a_j$ 且 $i < j$, 那么 i 永远不会有比 j 优。
- 可以整理出一个单调递减的队列。

Solution

- 从左往右考虑每个数。
- 若 $a_i \leq a_j$ 且 $i < j$, 那么 i 永远不会比 j 优。
- 可以整理出一个单调递减的队列。
- 若队首不在区间内, 则出队, 否则队首就是最大值。

Solution

- 从左往右考虑每个数。
- 若 $a_i \leq a_j$ 且 $i < j$, 那么 i 永远不会比 j 优。
- 可以整理出一个单调递减的队列。
- 若队首不在区间内, 则出队, 否则队首就是最大值。
- 不断将不满足递减性质的队尾弹出。

Solution

- 从左往右考虑每个数。
- 若 $a_i \leq a_j$ 且 $i < j$, 那么 i 永远不会有比 j 优。
- 可以整理出一个单调递减的队列。
- 若队首不在区间内, 则出队, 否则队首就是最大值。
- 不断将不满足递减性质的队尾弹出。
- 时间复杂度 $O(n)$ 。

Parcel

$n \times n$ 的 01 方阵，从中找到一个面积最大的全 0 矩阵。

Parcel

$n \times n$ 的 01 方阵，从中找到一个面积最大的全 0 矩阵。

- $n \leq 2000$ 。

Parcel

$n \times n$ 的 01 方阵，从中找到一个面积最大的全 0 矩阵。

- $n \leq 2000$ 。
- Source : POI 2002

Kanade's sum

给定一个 1 到 n 的全排列 A_1, A_2, \dots, A_n 以及一个参数 k 。

Kanade's sum

给定一个 1 到 n 的全排列 A_1, A_2, \dots, A_n 以及一个参数 k 。
对于一个区间 $[l, r]$, 设 $f(l, r)$ 为 $[l, r]$ 区间内第 k 大的数。

Kanade's sum

给定一个 1 到 n 的全排列 A_1, A_2, \dots, A_n 以及一个参数 k 。
对于一个区间 $[l, r]$, 设 $f(l, r)$ 为 $[l, r]$ 区间内第 k 大的数。
求所有区间的 f 的和。

Kanade's sum

给定一个 1 到 n 的全排列 A_1, A_2, \dots, A_n 以及一个参数 k 。

对于一个区间 $[l, r]$ ，设 $f(l, r)$ 为 $[l, r]$ 区间内第 k 大的数。

求所有区间的 f 的和。

- $n \leq 500000$ 。

Kanade's sum

给定一个 1 到 n 的全排列 A_1, A_2, \dots, A_n 以及一个参数 k 。
对于一个区间 $[l, r]$, 设 $f(l, r)$ 为 $[l, r]$ 区间内第 k 大的数。
求所有区间的 f 的和。

- $n \leq 500000$ 。
- $k \leq 80$ 。

Kanade's sum

给定一个 1 到 n 的全排列 A_1, A_2, \dots, A_n 以及一个参数 k 。
对于一个区间 $[l, r]$, 设 $f(l, r)$ 为 $[l, r]$ 区间内第 k 大的数。
求所有区间的 f 的和。

- $n \leq 500000$ 。
- $k \leq 80$ 。
- Source : 2017 Multi-University Training Contest 3

寻找不变量与变化点

- 寻找问题中不变的部分，起到优化效果。

寻找不变量与变化点

- 寻找问题中不变的部分，起到优化效果。
- 分析清楚变化点，考虑如何维护变化。

Zapiekanki 2

n 个顾客按时间顺序依次点餐，每个顾客会恰好点一份砂锅。第 i 个顾客点餐时间为 t_i ，而且只接受 $\geq t_i$ 时刻之后出锅的砂锅。

Zapiekaniki 2

n 个顾客按时间顺序依次点餐，每个顾客会恰好点一份砂锅。第 i 个顾客点餐时间为 t_i ，而且只接受 $\geq t_i$ 时刻之后出锅的砂锅。

你只有一个烤箱，做一份砂锅需要 d 时间，且中间不能打开烤箱，也不能同时做多份砂锅，你可以从 0 时刻开始做砂锅。

Zapiekanki 2

n 个顾客按时间顺序依次点餐，每个顾客会恰好点一份砂锅。第 i 个顾客点餐时间为 t_i ，而且只接受 $\geq t_i$ 时刻之后出锅的砂锅。

你只有一个烤箱，做一份砂锅需要 d 时间，且中间不能打开烤箱，也不能同时做多份砂锅，你可以从 0 时刻开始做砂锅。

m 次询问，每次给定一个 d ，求所有顾客等待时间之和的最小值。

Zapiiekanki 2

n 个顾客按时间顺序依次点餐，每个顾客会恰好点一份砂锅。第 i 个顾客点餐时间为 t_i ，而且只接受 $\geq t_i$ 时刻之后出锅的砂锅。

你只有一个烤箱，做一份砂锅需要 d 时间，且中间不能打开烤箱，也不能同时做多份砂锅，你可以从 0 时刻开始做砂锅。

m 次询问，每次给定一个 d ，求所有顾客等待时间之和的最小值。

- $n, m \leq 200000$ 。

Zapiekaniki 2

n 个顾客按时间顺序依次点餐，每个顾客会恰好点一份砂锅。第 i 个顾客点餐时间为 t_i ，而且只接受 $\geq t_i$ 时刻之后出锅的砂锅。

你只有一个烤箱，做一份砂锅需要 d 时间，且中间不能打开烤箱，也不能同时做多份砂锅，你可以从 0 时刻开始做砂锅。

m 次询问，每次给定一个 d ，求所有顾客等待时间之和的最小值。

- $n, m \leq 200000$ 。
- Source : PA 2017

Solution

- 分析问题的本质：最小化所有顾客拿到砂锅的时间之和。

Solution

- 分析问题的本质：最小化所有顾客拿到砂锅的时间之和。
- 设 f_i 表示顾客 i 拿到砂锅的时间。

Solution

- 分析问题的本质：最小化所有顾客拿到砂锅的时间之和。
- 设 f_i 表示顾客 i 拿到砂锅的时间。
- $f_0 = 0$ 。

Solution

- 分析问题的本质：最小化所有顾客拿到砂锅的时间之和。
- 设 f_i 表示顾客 i 拿到砂锅的时间。
- $f_0 = 0$ 。
- $f_i = \max(f_{i-1} + d, t_i)$ 。

Solution

- 分析问题的本质：最小化所有顾客拿到砂锅的时间之和。
- 设 f_i 表示顾客 i 拿到砂锅的时间。
- $f_0 = 0$ 。
- $f_i = \max(f_{i-1} + d, t_i)$ 。
- 时间复杂度 $O(nm)$ ，不能接受。

Greenhouse Growth

维护一个序列 h_1, h_2, \dots, h_n , 两种操作 :

Greenhouse Growth

维护一个序列 h_1, h_2, \dots, h_n , 两种操作 :

A : 从 2 到 n 依次考虑每个位置 i , 若 $h_i < h_{i-1}$, 则将 h_i 增加 1。

Greenhouse Growth

维护一个序列 h_1, h_2, \dots, h_n , 两种操作 :

A : 从 2 到 n 依次考虑每个位置 i , 若 $h_i < h_{i-1}$, 则将 h_i 增加 1。

B : 从 $n-1$ 到 1 依次考虑每个位置 i , 若 $h_i < h_{i+1}$, 则将 h_i 增加 1。

Greenhouse Growth

维护一个序列 h_1, h_2, \dots, h_n , 两种操作 :

A : 从 2 到 n 依次考虑每个位置 i , 若 $h_i < h_{i-1}$, 则将 h_i 增加 1。

B : 从 $n - 1$ 到 1 依次考虑每个位置 i , 若 $h_i < h_{i+1}$, 则将 h_i 增加 1。

求 m 次操作结束后序列每一项的值。

Greenhouse Growth

维护一个序列 h_1, h_2, \dots, h_n , 两种操作 :

A : 从 2 到 n 依次考虑每个位置 i , 若 $h_i < h_{i-1}$, 则将 h_i 增加 1。

B : 从 $n-1$ 到 1 依次考虑每个位置 i , 若 $h_i < h_{i+1}$, 则将 h_i 增加 1。

求 m 次操作结束后序列每一项的值。

- $n, m \leq 300000$ 。

Greenhouse Growth

维护一个序列 h_1, h_2, \dots, h_n , 两种操作 :

A : 从 2 到 n 依次考虑每个位置 i , 若 $h_i < h_{i-1}$, 则将 h_i 增加 1。

B : 从 $n - 1$ 到 1 依次考虑每个位置 i , 若 $h_i < h_{i+1}$, 则将 h_i 增加 1。

求 m 次操作结束后序列每一项的值。

- $n, m \leq 300000$ 。
- Source : CERC 2015

Range Maximum Query

给定一个序列 a_1, a_2, \dots, a_n , m 次询问一个区间 $[l, r]$ 的区间最大值。

Range Maximum Query

给定一个序列 a_1, a_2, \dots, a_n , m 次询问一个区间 $[l, r]$ 的区间最大值。

- $n, m \leq 10^7$ 。

Solution

- 从 1 到 n 考虑每个右端点为 r 的询问。

Solution

- 从 1 到 n 考虑每个右端点为 r 的询问。
- 初始时 n 个位置独立。

Solution

- 从 1 到 n 考虑每个右端点为 r 的询问。
- 初始时 n 个位置独立。
- 到 r 时将 r 的父亲指向 $r+1$, 边权为 a_r 。

Solution

- 从 1 到 n 考虑每个右端点为 r 的询问。
- 初始时 n 个位置独立。
- 到 r 时将 r 的父亲指向 $r+1$, 边权为 a_r 。
- 则询问 $[l, r]$ 的答案为 l 到根路径上的边权最大值。

Solution

- 从 1 到 n 考虑每个右端点为 r 的询问。
- 初始时 n 个位置独立。
- 到 r 时将 r 的父亲指向 $r+1$, 边权为 a_r 。
- 则询问 $[l, r]$ 的答案为 l 到根路径上的边权最大值。
- 并查集路径压缩即可。

Solution

- 从 1 到 n 考虑每个右端点为 r 的询问。
- 初始时 n 个位置独立。
- 到 r 时将 r 的父亲指向 $r+1$, 边权为 a_r 。
- 则询问 $[l, r]$ 的答案为 l 到根路径上的边权最大值。
- 并查集路径压缩即可。
- 时间复杂度 $O(n\alpha(n))$ 。

区间赋值

给定一个序列 a_1, a_2, \dots, a_n , m 次操作。

区间赋值

给定一个序列 a_1, a_2, \dots, a_n , m 次操作。

每次指定 l, r, k , 将 $[l, r]$ 每个数都赋值为 k 。

区间赋值

给定一个序列 a_1, a_2, \dots, a_n , m 次操作。

每次指定 l, r, k , 将 $[l, r]$ 每个数都赋值为 k 。

输出最终的序列。

区间赋值

给定一个序列 a_1, a_2, \dots, a_n , m 次操作。

每次指定 l, r, k , 将 $[l, r]$ 每个数都赋值为 k 。

输出最终的序列。

- $n, m \leq 10^7$ 。

Solution

- 时间倒流法。

Solution

- 时间倒流法。
- 倒着处理每个操作，那么之前赋值过的位置不能再被赋值。

Solution

- 时间倒流法。
- 倒着处理每个操作，那么之前赋值过的位置不能再被赋值。
- 设 f_i 表示 i 右侧第一个未被赋值的位置。

Solution

- 时间倒流法。
- 倒着处理每个操作，那么之前赋值过的位置不能再被赋值。
- 设 f_i 表示 i 右侧第一个未被赋值的位置。
- 每次沿着 f 遍历 $[l, r]$ 中所有未赋值的位置，同时将 f_i 指向 f_{i+1} 。

Solution

- 时间倒流法。
- 倒着处理每个操作，那么之前赋值过的位置不能再被赋值。
- 设 f_i 表示 i 右侧第一个未被赋值的位置。
- 每次沿着 f 遍历 $[l, r]$ 中所有未赋值的位置，同时将 f_i 指向 f_{i+1} 。
- 并查集路径压缩即可。

Solution

- 时间倒流法。
- 倒着处理每个操作，那么之前赋值过的位置不能再被赋值。
- 设 f_i 表示 i 右侧第一个未被赋值的位置。
- 每次沿着 f 遍历 $[l, r]$ 中所有未赋值的位置，同时将 f_i 指向 f_{i+1} 。
- 并查集路径压缩即可。
- 时间复杂度 $O(n\alpha(n))$ 。

Buffalo Barricades

第一象限中有 n 个点，第 i 个点位于 $(x_i - 0.5, y_i - 0.5)$ 。

Buffalo Barricades

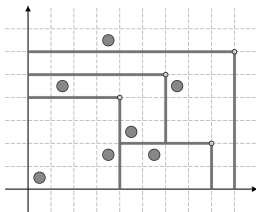
第一象限中有 n 个点，第 i 个点位于 $(x_i - 0.5, y_i - 0.5)$ 。

依次放入 m 个篱笆，每个篱笆是从 $P_i(x_i, y_i)$ 点往左往下发射两条射线，直到碰到另一个篱笆或者坐标轴。

Buffalo Barricades

第一象限中有 n 个点, 第 i 个点位于 $(x_i - 0.5, y_i - 0.5)$ 。

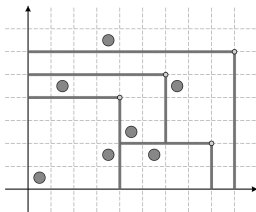
依次放入 m 个篱笆，每个篱笆是从 $P_i(x_i, y_i)$ 点往左往下发射两条射线，直到碰到另一个篱笆或者坐标轴。



Buffalo Barricades

第一象限中有 n 个点，第 i 个点位于 $(x_i - 0.5, y_i - 0.5)$ 。

依次放入 m 个篱笆，每个篱笆是从 $P_i(x_i, y_i)$ 点往左往下发射两条射线，直到碰到另一个篱笆或者坐标轴。

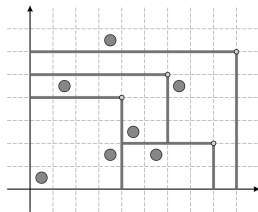


问每次新加入的篱笆划分出的区域内部有多少个点。

Buffalo Barricades

第一象限中有 n 个点，第 i 个点位于 $(x_i - 0.5, y_i - 0.5)$ 。

依次放入 m 个篱笆，每个篱笆是从 $P_i(x_i, y_i)$ 点往左往下发射两条射线，直到碰到另一个篱笆或者坐标轴。



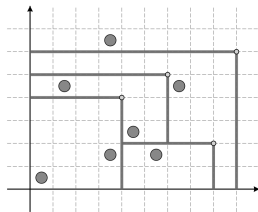
问每次新加入的篱笆划分出的区域内部有多少个点。

- $n, m \leq 300000$ 。

Buffalo Barricades

第一象限中有 n 个点，第 i 个点位于 $(x_i - 0.5, y_i - 0.5)$ 。

依次放入 m 个篱笆，每个篱笆是从 $P_i(x_i, y_i)$ 点往左往下发射两条射线，直到碰到另一个篱笆或者坐标轴。



问每次新加入的篱笆划分出的区域内部有多少个点。

- $n, m \leq 300000$ 。
- Source : CERC 2017

题目提交

课上例题：

http://acm.hdu.edu.cn/diy/contest_show.php?cid=33143

课后习题：

http://acm.hdu.edu.cn/diy/contest_show.php?cid=33144

密码：

G*&GSF&*t387tr

Thank you!