常见的数据结构维护技巧

Claris

Hangzhou Dianzi University

2017 年 7 月 25 日

Overview

- 本节讨论一些常见的数据结构维护技巧。

Overview

- 本节讨论一些常见的数据结构维护技巧。
- 例题涉及少量树链剖分或平衡树。

Overview

- 本节讨论一些常见的数据结构维护技巧。
- 例题涉及少量树链剖分或平衡树。
- 例题涉及大量线段树。

Overview

- 本节讨论一些常见的数据结构维护技巧。
- 例题涉及少量树链剖分或平衡树。
- 例题涉及大量线段树。
- 建议学会线段树后再来学习。

The Best Teams

给定 n 个球员，第 i 个球员年龄为 age_i，水平为 $skill_i$。没有任何两个球员的水平相同。

The Best Teams

给定 n 个球员，第 i 个球员年龄为 age_i，水平为 $skill_i$。没有任何两个球员的水平相同。

将这些球员按水平排序，对于一次比赛，你需要选择若干个球员去比赛，但不能同时选择两个水平相邻的球员。

The Best Teams

给定 n 个球员，第 i 个球员年龄为 age_i，水平为 $skill_i$。没有任何两个球员的水平相同。

将这些球员按水平排序，对于一次比赛，你需要选择若干个球员去比赛，但不能同时选择两个水平相邻的球员。

m 次询问，每次给定 a 和 k，表示要在年龄不超过 a 的球员中选择不超过 k 个球员，请计算 $skill$ 和的最大值。

The Best Teams

给定 n 个球员，第 i 个球员年龄为 age_i，水平为 $skill_i$。没有任何两个球员的水平相同。

将这些球员按水平排序，对于一次比赛，你需要选择若干个球员去比赛，但不能同时选择两个水平相邻的球员。

m 次询问，每次给定 a 和 k，表示要在年龄不超过 a 的球员中选择不超过 k 个球员，请计算 $skill$ 和的最大值。

- $1 \leq n, m \leq 300000$。

The Best Teams

给定 n 个球员，第 i 个球员年龄为 age_i，水平为 $skill_i$。没有任何两个球员的水平相同。

将这些球员按水平排序，对于一次比赛，你需要选择若干个球员去比赛，但不能同时选择两个水平相邻的球员。

m 次询问，每次给定 a 和 k，表示要在年龄不超过 a 的球员中选择不超过 k 个球员，请计算 $skill$ 和的最大值。

- $1 \leq n, m \leq 300000$。
- $1 \leq age_i, skill_i \leq 10^9$。

The Best Teams

给定 n 个球员，第 i 个球员年龄为 age_i，水平为 $skill_i$。没有任何两个球员的水平相同。

将这些球员按水平排序，对于一次比赛，你需要选择若干个球员去比赛，但不能同时选择两个水平相邻的球员。

m 次询问，每次给定 a 和 k，表示要在年龄不超过 a 的球员中选择不超过 k 个球员，请计算 $skill$ 和的最大值。

- $1 \leq n, m \leq 300000$。
- $1 \leq age_i, skill_i \leq 10^9$。
- Source：Balkan Olympiad In Informatics 2012

Solution

- 将球员和询问按照年龄排序，即可去掉年龄的限制。

Solution

- 将球员和询问按照年龄排序，即可去掉年龄的限制。
- 将球员按水平排序后维护线段树，显然最优解一定是从大到小贪心选择。

Solution

- 将球员和询问按照年龄排序，即可去掉年龄的限制。
- 将球员按水平排序后维护线段树，显然最优解一定是从大到小贪心选择。
- 线段树上每个节点维护：
  $g[0/1]$: $r+1$ 不选/选的时候，l 选不选。
  $c[0/1]$: $r+1$ 不选/选的时候，中间选了几个。
  $s[0/1]$: $r+1$ 不选/选的时候，中间选的和。

Solution

- 将球员和询问按照年龄排序，即可去掉年龄的限制。
- 将球员按水平排序后维护线段树，显然最优解一定是从大到小贪心选择。
- 线段树上每个节点维护：
  $g[0/1]$: $r+1$ 不选/选的时候，l 选不选。
  $c[0/1]$: $r+1$ 不选/选的时候，中间选了几个。
  $s[0/1]$: $r+1$ 不选/选的时候，中间选的和。
- 查询的时候在线段树上二分即可。

Solution

- 将球员和询问按照年龄排序，即可去掉年龄的限制。
- 将球员按水平排序后维护线段树，显然最优解一定是从大到小贪心选择。
- 线段树上每个节点维护：
  $g[0/1]$: $r+1$ 不选/选的时候，l 选不选。
  $c[0/1]$: $r+1$ 不选/选的时候，中间选了几个。
  $s[0/1]$: $r+1$ 不选/选的时候，中间选的和。
- 查询的时候在线段树上二分即可。
- 时间复杂度 $O((n+m)\log n)$。

Handsome

一个 n 位数是 Handsome 的，当且仅当其中只有 $0/1/2$，而且相邻的两个数构成的数对不在危险集合中。

Handsome

一个 n 位数是 Handsome 的，当且仅当其中只有 $0/1/2$，而且相邻的两个数构成的数对不在危险集合中。

n 位数 A 在排列 P 的意义下字典序小于 B，则存在一个 $k(1 \leq k < n)$ 使得：

Handsome

一个 n 位数是 Handsome 的，当且仅当其中只有 $0/1/2$，而且相邻的两个数构成的数对不在危险集合中。

n 位数 A 在排列 P 的意义下字典序小于 B，则存在一个 $k(1 \le k < n)$ 使得：

$$A_{P_1} = B_{P_1}, A_{P_2} = B_{P_2} \ldots A_{P_k} = B_{P_k}, A_{P_{k+1}} < B_{P_{k+1}}$$

Handsome

一个 n 位数是 Handsome 的，当且仅当其中只有 $0/1/2$，而且相邻的两个数构成的数对不在危险集合中。

n 位数 A 在排列 P 的意义下字典序小于 B，则存在一个 $k(1 \leq k < n)$ 使得：

$$A_{P_1} = B_{P_1}, A_{P_2} = B_{P_2} \dots A_{P_k} = B_{P_k}, A_{P_{k+1}} < B_{P_{k+1}}$$

给定 n，危险集合，排列 P，以及一个 Handsome 数 T，统计在排列 P 意义下小于 T 的 Handsome 数的个数。答案模 10^9+7。

Handsome

一个 n 位数是 Handsome 的，当且仅当其中只有 $0/1/2$，而且相邻的两个数构成的数对不在危险集合中。

n 位数 A 在排列 P 的意义下字典序小于 B，则存在一个 $k(1 \le k < n)$ 使得：

$A_{P_1} = B_{P_1}, A_{P_2} = B_{P_2} \dots A_{P_k} = B_{P_k}, A_{P_{k+1}} < B_{P_{k+1}}$

给定 n，危险集合，排列 P，以及一个 Handsome 数 T，统计在排列 P 意义下小于 T 的 Handsome 数的个数。答案模 $10^9 + 7$。

- $1 \le n \le 400000$。

Handsome

一个 n 位数是 Handsome 的，当且仅当其中只有 $0/1/2$，而且相邻的两个数构成的数对不在危险集合中。

n 位数 A 在排列 P 的意义下字典序小于 B，则存在一个 $k(1 \le k < n)$ 使得：

$A_{P_1} = B_{P_1}, A_{P_2} = B_{P_2} ... A_{P_k} = B_{P_k}, A_{P_{k+1}} < B_{P_{k+1}}$

给定 n，危险集合，排列 P，以及一个 Handsome 数 T，统计在排列 P 意义下小于 T 的 Handsome 数的个数。答案模 $10^9 + 7$。

- $1 \le n \le 400000$。
- Source：Balkan Olympiad In Informatics 2012

Solution

- DP 求出 $f_{i,j,k}$ 表示长度为 i 的序列，第一个位置是 j，最后一个位置是 k 时合法的方案数。

Solution

- DP 求出 $f_{i,j,k}$ 表示长度为 i 的序列，第一个位置是 j，最后一个位置是 k 时合法的方案数。
- 从后往前枚举与 T 的 LCP 以及那个位置应该改成什么。

Solution

- DP 求出 $f_{i,j,k}$ 表示长度为 i 的序列，第一个位置是 j，最后一个位置是 k 时合法的方案数。
- 从后往前枚举与 T 的 LCP 以及那个位置应该改成什么。
- 用线段树维护区间内最左最右的已经确定的位置，以及区间内的合法方案数。

Solution

- DP 求出 $f_{i,j,k}$ 表示长度为 i 的序列，第一个位置是 j，最后一个位置是 k 时合法的方案数。
- 从后往前枚举与 T 的 LCP 以及那个位置应该改成什么。
- 用线段树维护区间内最左最右的已经确定的位置，以及区间内的合法方案数。
- 合并的时候只需要将左右儿子的答案乘起来，然后再乘以左儿子最右到右儿子最左这一段区间的方案数即可。

Solution

- DP 求出 $f_{i,j,k}$ 表示长度为 i 的序列，第一个位置是 j，最后一个位置是 k 时合法的方案数。
- 从后往前枚举与 T 的 LCP 以及那个位置应该改成什么。
- 用线段树维护区间内最左最右的已经确定的位置，以及区间内的合法方案数。
- 合并的时候只需要将左右儿子的答案乘起来，然后再乘以左儿子最右到右儿子最左这一段区间的方案数即可。
- 时间复杂度 $O(n\log n)$。

动态图

维护一张 n 个点，m 条边的无向图，q 次操作：

动态图

维护一张 n 个点，m 条边的无向图，q 次操作：

(1) 添加一条边 (u, v)。

动态图

维护一张 n 个点，m 条边的无向图，q 次操作：

(1) 添加一条边 (u, v)。

(2) 删除一条边 (u, v)。

动态图

维护一张 n 个点，m 条边的无向图，q 次操作：

(1) 添加一条边 (u, v)。

(2) 删除一条边 (u, v)。

(3) 询问 u 与 v 是否连通。

动态图

维护一张 n 个点，m 条边的无向图，q 次操作：

(1) 添加一条边 (u, v)。

(2) 删除一条边 (u, v)。

(3) 询问 u 与 v 是否连通。

- $1 \leq n \leq 100000$。

动态图

维护一张 n 个点，m 条边的无向图，q 次操作：

(1) 添加一条边 (u, v)。

(2) 删除一条边 (u, v)。

(3) 询问 u 与 v 是否连通。

- $1 \le n \le 100000$。
- $1 \le m \le 100000$。

动态图

维护一张 n 个点，m 条边的无向图，q 次操作：
(1) 添加一条边 (u, v)。
(2) 删除一条边 (u, v)。
(3) 询问 u 与 v 是否连通。

- $1 \leq n \leq 100000$。
- $1 \leq m \leq 100000$。
- $1 \leq q \leq 100000$。

Solution

- 每条边 (u, v) 存在的时间为一个区间 $[l, r]$。

Solution

- 每条边 (u, v) 存在的时间为一个区间 $[l, r]$。
- 对时间建立线段树，每条边可以看成向区间 $[l, r]$ 打一个 “插入一条边” 的标记。

Solution

- 每条边 (u, v) 存在的时间为一个区间 $[l, r]$。
- 对时间建立线段树，每条边可以看成向区间 $[l, r]$ 打一个 “插入一条边” 的标记。
- 找到线段树上对应的 $O(\log q)$ 个节点，在标记集合中加入这条边。

Solution

- 每条边 (u, v) 存在的时间为一个区间 $[l, r]$。
- 对时间建立线段树，每条边可以看成向区间 $[l, r]$ 打一个 “插入一条边” 的标记。
- 找到线段树上对应的 $O(\log q)$ 个节点，在标记集合中加入这条边。
- 这样 q 次加边删边操作就转化为了 $O(q \log q)$ 次加边操作。

Solution

- DFS 整棵线段树，到叶子时回答询问。

Solution

- DFS 整棵线段树，到叶子时回答询问。
- 因为只有加边和撤销操作，因此只需要维护一个带撤销的并查集。

Solution

- DFS 整棵线段树，到叶子时回答询问。
- 因为只有加边和撤销操作，因此只需要维护一个带撤销的并查集。
- 按秩合并的并查集单次操作复杂度为严格 $O(\log n)$，撤销只需要将修改操作按时间倒回去。

Solution

- DFS 整棵线段树，到叶子时回答询问。
- 因为只有加边和撤销操作，因此只需要维护一个带撤销的并查集。
- 按秩合并的并查集单次操作复杂度为严格 $O(\log n)$，撤销只需要将修改操作按时间倒回去。
- 时间复杂度 $O(q \log q \log n)$。

Card Game

平面上有 n 条直线 $y = kx + b$，q 次操作：

Card Game

平面上有 n 条直线 $y = kx + b$，q 次操作：

(1) 添加一条直线 $y = kx + b$。

Card Game

平面上有 n 条直线 $y = kx + b$，q 次操作：

(1) 添加一条直线 $y = kx + b$。

(2) 删除一条直线 $y = kx + b$。

Card Game

平面上有 n 条直线 $y = kx + b$，q 次操作：

(1) 添加一条直线 $y = kx + b$。

(2) 删除一条直线 $y = kx + b$。

(3) 给定 $[L, R]$，在 $[L, R]$ 选择一个整数 x，然后计算 $y = kx + b$ 的最小值。请挑选最优的 x，使得计算结果 y 最大。

Card Game

平面上有 n 条直线 $y = kx + b$，q 次操作：

(1) 添加一条直线 $y = kx + b$。

(2) 删除一条直线 $y = kx + b$。

(3) 给定 $[L, R]$，在 $[L, R]$ 选择一个整数 x，然后计算 $y = kx + b$ 的最小值。请挑选最优的 x，使得计算结果 y 最大。

- $1 \leq n, q \leq 50000$。

Card Game

平面上有 n 条直线 $y = kx + b$，q 次操作：

(1) 添加一条直线 $y = kx + b$。

(2) 删除一条直线 $y = kx + b$。

(3) 给定 $[L, R]$，在 $[L, R]$ 选择一个整数 x，然后计算 $y = kx + b$ 的最小值。请挑选最优的 x，使得计算结果 y 最大。

- $1 \le n, q \le 50000$。
- $-10^9 \le k, b \le 10^9$。

Card Game

平面上有 n 条直线 $y = kx + b$，q 次操作：

(1) 添加一条直线 $y = kx + b$。

(2) 删除一条直线 $y = kx + b$。

(3) 给定 $[L, R]$，在 $[L, R]$ 选择一个整数 x，然后计算 $y = kx + b$ 的最小值。请挑选最优的 x，使得计算结果 y 最大。

- $1 \leq n, q \leq 50000$。
- $-10^9 \leq k, b \leq 10^9$。
- $-10^9 \leq L \leq R \leq 10^9$。

Card Game

平面上有 n 条直线 $y = kx + b$，q 次操作：

(1) 添加一条直线 $y = kx + b$。

(2) 删除一条直线 $y = kx + b$。

(3) 给定 $[L, R]$，在 $[L, R]$ 选择一个整数 x，然后计算 $y = kx + b$ 的最小值。请挑选最优的 x，使得计算结果 y 最大。

- $1 \le n, q \le 50000$。
- $-10^9 \le k, b \le 10^9$。
- $-10^9 \le L \le R \le 10^9$。
- Source：ZOJ 3967

Solution

- 问题等价于求这些直线形成的上凸壳中 $[L, R]$ 的最大值。

Solution

- 问题等价于求这些直线形成的上凸壳中 $[L, R]$ 的最大值。
- 在凸壳上二分导数，找到最大值即可。

Solution

- 问题等价于求这些直线形成的上凸壳中 $[L, R]$ 的最大值。
- 在凸壳上二分导数，找到最大值即可。
- 动态半平面交？

Solution

- 离线求出每条直线存在的时间区间，在时间线段树上打标记，转化成 $O(q \log q)$ 次插入。

Solution

- 离线求出每条直线存在的时间区间，在时间线段树上打标记，转化成 $O(q\log q)$ 次插入。
- DFS 整棵线段树，到叶子时回答询问。

Solution

- 离线求出每条直线存在的时间区间，在时间线段树上打标记，转化成 $O(q \log q)$ 次插入。
- DFS 整棵线段树，到叶子时回答询问。
- 现在需要维护一个数据结构，支持插入直线，询问单点最值。

Solution

- 离线求出每条直线存在的时间区间，在时间线段树上打标记，转化成 $O(q \log q)$ 次插入。
- DFS 整棵线段树，到叶子时回答询问。
- 现在需要维护一个数据结构，支持插入直线，询问单点最值。
- 利用线段树就可以解决这个问题。

Solution

- 在线段树上每个节点维护最优直线。

Solution

- 在线段树上每个节点维护最优直线。
- 插入直线时，若它完全被当前区间最优直线 p 替代，或者它可以完全替代 p，那么显然取最优的那个然后结束插入即可。

Solution

- 在线段树上每个节点维护最优直线。
- 插入直线时，若它完全被当前区间最优直线 p 替代，或者它可以完全替代 p，那么显然取最优的那个然后结束插入即可。
- 否则，插入直线必然与 p 在该区间内存在交点。

Solution

- 在线段树上每个节点维护最优直线。
- 插入直线时，若它完全被当前区间最优直线 p 替代，或者它可以完全替代 p，那么显然取最优的那个然后结束插入即可。
- 否则，插入直线必然与 p 在该区间内存在交点。
- 保留优势区间较大的直线，将另一条直线递归下传到那一侧。

Solution

- 在线段树上每个节点维护最优直线。
- 插入直线时，若它完全被当前区间最优直线 p 替代，或者它可以完全替代 p，那么显然取最优的那个然后结束插入即可。
- 否则，插入直线必然与 p 在该区间内存在交点。
- 保留优势区间较大的直线，将另一条直线递归下传到那一侧。
- 单次插入时间复杂度 $O(\log n)$。

Solution

- 如何查询 x 点的最值？

Solution

- 如何查询 x 点的最值？
- 在线段树上找到 x 点，一路向上走到根，用 $O(\log n)$ 条直线更新答案。

Solution

- 如何查询 x 点的最值？
- 在线段树上找到 x 点，一路向上走到根，用 $O(\log n)$ 条直线更新答案。
- 单次查询时间复杂度 $O(\log n)$。

Solution

- 如何查询 x 点的最值？
- 在线段树上找到 x 点，一路向上走到根，用 $O(\log n)$ 条直线更新答案。
- 单次查询时间复杂度 $O(\log n)$。
- 沿着线段树 DFS，用栈按时间记录所有修改，那么可以很方便地实现还原。

Solution

- 如何查询 x 点的最值？
- 在线段树上找到 x 点，一路向上走到根，用 $O(\log n)$ 条直线更新答案。
- 单次查询时间复杂度 $O(\log n)$。
- 沿着线段树 DFS，用栈按时间记录所有修改，那么可以很方便地实现还原。
- 总时间复杂度 $O(n \log^2 n)$。

Pandaria

给定一张 n 个点，m 条边的无向图，点 i 的颜色为 c_i。

Pandaria

给定一张 n 个点，m 条边的无向图，点 i 的颜色为 c_i。

q 次询问，每次给定 x, k，问从 x 点出发，经过所有边权不超过 k 的边走能到达的所有点中，哪种颜色出现次数最多。

Pandaria

给定一张 n 个点，m 条边的无向图，点 i 的颜色为 c_i。

q 次询问，每次给定 x, k，问从 x 点出发，经过所有边权不超过 k 的边走能到达的所有点中，哪种颜色出现次数最多。

强制在线。

Pandaria

给定一张 n 个点，m 条边的无向图，点 i 的颜色为 c_i。

q 次询问，每次给定 x, k，问从 x 点出发，经过所有边权不超过 k 的边走能到达的所有点中，哪种颜色出现次数最多。

强制在线。

- $1 \leq c_i \leq n \leq 100000$。

Pandaria

给定一张 n 个点，m 条边的无向图，点 i 的颜色为 c_i。

q 次询问，每次给定 x, k，问从 x 点出发，经过所有边权不超过 k 的边走能到达的所有点中，哪种颜色出现次数最多。

强制在线。

- $1 \le c_i \le n \le 100000$。
- $1 \le m, q \le 200000$。

Pandaria

给定一张 n 个点，m 条边的无向图，点 i 的颜色为 c_i。

q 次询问，每次给定 x, k，问从 x 点出发，经过所有边权不超过 k 的边走能到达的所有点中，哪种颜色出现次数最多。

强制在线。

- $1 \leq c_i \leq n \leq 100000$。
- $1 \leq m, q \leq 200000$。
- $1 \leq k \leq 10^6$。

Pandaria

给定一张 n 个点，m 条边的无向图，点 i 的颜色为 c_i。

q 次询问，每次给定 x, k，问从 x 点出发，经过所有边权不超过 k 的边走能到达的所有点中，哪种颜色出现次数最多。

强制在线。

- $1 \le c_i \le n \le 100000$。
- $1 \le m, q \le 200000$。
- $1 \le k \le 10^6$。
- Source：2016-2017 ACM-ICPC China-Final

Solution

- 将边按权值从小到大排序，求出最小生成树。

Solution

- 将边按权值从小到大排序，求出最小生成树。
- 当以边权 w 合并 u, v 时，新建一个节点 x，左右儿子分别为 u, v，x 的权值设为 w。

Solution

- 将边按权值从小到大排序，求出最小生成树。
- 当以边权 w 合并 u, v 时，新建一个节点 x，左右儿子分别为 u, v，x 的权值设为 w。
- 如此一来最终会得到一个森林，方便起见我们可以添加 inf 边将森林连成树。

Solution

- 将边按权值从小到大排序，求出最小生成树。
- 当以边权 w 合并 u, v 时，新建一个节点 x，左右儿子分别为 u, v，x 的权值设为 w。
- 如此一来最终会得到一个森林，方便起见我们可以添加 inf 边将森林连成树。
- 联想 Kruskal 的过程，这棵树越高层的地方权值越大。

Solution

- 对于询问 x, k，从 x 点开始向上走，找到权值不超过 k 的最高的点 y。

Solution

- 对于询问 x, k，从 x 点开始向上走，找到权值不超过 k 的最高的点 y。
- 那么答案就是 y 子树内的颜色众数。

Solution

- 对于询问 x, k，从 x 点开始向上走，找到权值不超过 k 的最高的点 y。
- 那么答案就是 y 子树内的颜色众数。
- 找 y 可以使用树链剖分或者树上倍增。

Solution

- 对于询问 x, k，从 x 点开始向上走，找到权值不超过 k 的最高的点 y。
- 那么答案就是 y 子树内的颜色众数。
- 找 y 可以使用树链剖分或者树上倍增。
- 如何求子树内颜色众数？

Solution

- 对于每个叶子，建立线段树，维护颜色区间内出现次数最多的次数。

Solution

- 对于每个叶子，建立线段树，维护颜色区间内出现次数最多的次数。
- 对于每个非叶子，需要将它的左右儿子的线段树合并起来。

Solution

- 对于每个叶子，建立线段树，维护颜色区间内出现次数最多的次数。
- 对于每个非叶子，需要将它的左右儿子的线段树合并起来。
- 合并方法：暴力递归合并，直到发现一方为空。

Solution

- 对于每个叶子，建立线段树，维护颜色区间内出现次数最多的次数。
- 对于每个非叶子，需要将它的左右儿子的线段树合并起来。
- 合并方法：暴力递归合并，直到发现一方为空。
- 考虑线段树上两个点只会在一个位置被暴力合并掉，故时间复杂度为 $O(n \log n)$。

K 个串

给定一个长度为 n 的序列 $a_1, a_2, \dots, a_n$ ，定义一个区间的价值为区间内所有数字的和 (重复数字只计算一次)。

K 个串

给定一个长度为 n 的序列 $a_1, a_2, \ldots, a_n$，定义一个区间的价值为区间内所有数字的和 (重复数字只计算一次)。

给定 k，将所有区间按价值从大到小排序，求第 k 大的价值。

K 个串

给定一个长度为 n 的序列 $a_1, a_2, \dots, a_n$ ，定义一个区间的价值为区间内所有数字的和 (重复数字只计算一次)。

给定 k ，将所有区间按价值从大到小排序，求第 k 大的价值。

- $1 \le n \le 100000$。

K 个串

给定一个长度为 n 的序列 $a_1, a_2, \ldots, a_n$，定义一个区间的价值为区间内所有数字的和 (重复数字只计算一次)。

给定 k，将所有区间按价值从大到小排序，求第 k 大的价值。

- $1 \leq n \leq 100000$。
- $1 \leq k \leq 200000$。

K 个串

给定一个长度为 n 的序列 $a_1, a_2, \ldots, a_n$ ，定义一个区间的价值为区间内所有数字的和 (重复数字只计算一次)。

给定 k ，将所有区间按价值从大到小排序，求第 k 大的价值。

- $1 \le n \le 100000$。
- $1 \le k \le 200000$。
- $|a_i| \le 10^9$。

K 个串

给定一个长度为 n 的序列 $a_1, a_2, \ldots, a_n$，定义一个区间的价值为区间内所有数字的和 (重复数字只计算一次)。

给定 k，将所有区间按价值从大到小排序，求第 k 大的价值。

- $1 \leq n \leq 100000$。
- $1 \leq k \leq 200000$。
- $|a_i| \leq 10^9$。
- Source： HihoCoder 1046

Solution

- 如何求一个区间的价值？

Solution

- 如何求一个区间的价值？
- 从左往右枚举右端点，用线段树维护每个左端点的去重后的区间和。

Solution

- 如何求一个区间的价值？
- 从左往右枚举右端点，用线段树维护每个左端点的去重后的区间和。
- 对于 a_r，需要在 $[pre_{a_r}+1, r]$ 里区间加上 a_r。

Solution

- 如何求一个区间的价值？
- 从左往右枚举右端点，用线段树维护每个左端点的去重后的区间和。
- 对于 a_r，需要在 $[pre_{a_r}+1, r]$ 里区间加上 a_r。
- 将线段树可持久化，并维护区间最大值，就可以在线回答形如“给定 r 以及 a, b”，问 l 在 $[a, b]$ 里 $[l, r]$ 的区间和的最大值的问题。

Solution

- 用一个大根堆维护五元组 (v, x, l, r, m)，表示区间和为 v，所在线段树根节点为 x，所选左端点范围为 $[l, r]$，选了 m。

Solution

- 用一个大根堆维护五元组 (v, x, l, r, m)，表示区间和为 v，所在线段树根节点为 x，所选左端点范围为 $[l, r]$，选了 m。
- 重复 k 次，每次取出堆顶，扩展出 $[l, m-1]$ 以及 $[m+1, r]$ 两个新状态。

Solution

- 用一个大根堆维护五元组 (v, x, l, r, m)，表示区间和为 v，所在线段树根节点为 x，所选左端点范围为 $[l, r]$，选了 m。
- 重复 k 次，每次取出堆顶，扩展出 $[l, m-1]$ 以及 $[m+1, r]$ 两个新状态。
- 时间复杂度 $O((n+k)\log n)$。

区间最大值

给定一个长度为 n 的序列 $a_1, a_2, \ldots, a_n$ ，m 次操作：

区间最大值

给定一个长度为 n 的序列 $a_1, a_2, ..., a_n$ ，m 次操作：

(1) 修改 a_x 的值。

区间最大值

给定一个长度为 n 的序列 $a_1, a_2, \dots, a_n$ ，m 次操作：

(1) 修改 a_x 的值。

(2) 给定区间 $[L, R]$ ，令 f_i 表示区间 $[L, i]$ 的最大值，求 $f_L, f_{L+1}, \dots, f_R$ 中不同元素的个数。

区间最大值

给定一个长度为 n 的序列 $a_1, a_2, \ldots, a_n$，m 次操作：

(1) 修改 a_x 的值。

(2) 给定区间 $[L, R]$，令 f_i 表示区间 $[L, i]$ 的最大值，求 $f_L, f_{L+1}, \ldots, f_R$ 中不同元素的个数。

- $1 \leq n \leq 100000$。

区间最大值

给定一个长度为 n 的序列 $a_1, a_2, \ldots, a_n$ ，m 次操作：

(1) 修改 a_x 的值。

(2) 给定区间 $[L, R]$ ，令 f_i 表示区间 $[L, i]$ 的最大值，求 $f_L, f_{L+1}, \ldots, f_R$ 中不同元素的个数。

- $1 \leq n \leq 100000$。
- $1 \leq m \leq 100000$。

区间最大值

给定一个长度为 n 的序列 $a_1, a_2, ..., a_n$ ，m 次操作：

(1) 修改 a_x 的值。

(2) 给定区间 $[L, R]$ ，令 f_i 表示区间 $[L, i]$ 的最大值，求 $f_L, f_{L+1}, ..., f_R$ 中不同元素的个数。

- $1 \leq n \leq 100000$。
- $1 \leq m \leq 100000$。
- $1 \leq a_i \leq 10^9$。

Solution

- 对序列 a 建立线段树，设 $query(x, t)$ 表示在节点 x 询问，之前的最大值为 t 的答案。

Solution

- 对序列 a 建立线段树，设 $query(x, t)$ 表示在节点 x 询问，之前的最大值为 t 的答案。
- 设 ans_x 表示 $query(x, 0)$ 的结果，val_x 表示 x 的区间最大值。

Solution

- 对序列 a 建立线段树，设 $query(x,t)$ 表示在节点 x 询问，之前的最大值为 t 的答案。
- 设 ans_x 表示 $query(x,0)$ 的结果，val_x 表示 x 的区间最大值。
- 若 x 为叶子节点，那么 $ans_x=1$。

Solution

- 对序列 a 建立线段树，设 $query(x,t)$ 表示在节点 x 询问，之前的最大值为 t 的答案。
- 设 ans_x 表示 $query(x,0)$ 的结果，val_x 表示 x 的区间最大值。
- 若 x 为叶子节点，那么 $ans_x = 1$。
- 若 $val_{left_x} \geq val_{right_x}$，那么右儿子不会新增答案，故 $ans_x = ans_{left_x}$。

Solution

- 对序列 a 建立线段树，设 $query(x, t)$ 表示在节点 x 询问，之前的最大值为 t 的答案。
- 设 ans_x 表示 $query(x, 0)$ 的结果，val_x 表示 x 的区间最大值。
- 若 x 为叶子节点，那么 $ans_x = 1$。
- 若 $val_{left_x} \geq val_{right_x}$，那么右儿子不会新增答案，故 $ans_x = ans_{left_x}$。
- 否则，经过左儿子的询问之后，t 一定会变成 val_{left_x}，故 $ans_x = ans_{left_x} + query(right_x, val_{left_x})$。

Solution

- 对序列 a 建立线段树，设 $query(x,t)$ 表示在节点 x 询问，之前的最大值为 t 的答案。
- 设 ans_x 表示 $query(x,0)$ 的结果，val_x 表示 x 的区间最大值。
- 若 x 为叶子节点，那么 $ans_x = 1$。
- 若 $val_{left_x} \geq val_{right_x}$，那么右儿子不会新增答案，故 $ans_x = ans_{left_x}$。
- 否则，经过左儿子的询问之后，t 一定会变成 val_{left_x}，故 $ans_x = ans_{left_x} + query(right_x, val_{left_x})$。
- 计算单个 val_x 的时间复杂度为 $O(\log n)$。

Solution

- 对序列 a 建立线段树，设 $query(x,t)$ 表示在节点 x 询问，之前的最大值为 t 的答案。
- 设 ans_x 表示 $query(x,0)$ 的结果，val_x 表示 x 的区间最大值。
- 若 x 为叶子节点，那么 $ans_x = 1$。
- 若 $val_{left_x} \geq val_{right_x}$，那么右儿子不会新增答案，故 $ans_x = ans_{left_x}$。
- 否则，经过左儿子的询问之后，t 一定会变成 val_{left_x}，故 $ans_x = ans_{left_x} + query(right_x, val_{left_x})$。
- 计算单个 val_x 的时间复杂度为 $O(\log n)$。
- 单点修改 a_x 的时间复杂度为 $O(\log^2 n)$。

Solution

- 对于询问 $[L, R]$，可以在线段树上分裂成 $O(\log n)$ 个 $query(x, t)$。

Solution

- 对于询问 $[L, R]$，可以在线段树上分裂成 $O(\log n)$ 个 $query(x, t)$。
- 若 x 为叶子节点，那么做法显然。

Solution

- 对于询问 $[L, R]$，可以在线段树上分裂成 $O(\log n)$ 个 $query(x, t)$。
- 若 x 为叶子节点，那么做法显然。
- 若 $t \geq val_{left_x}$，那么显然在左儿子不会新增答案，直接返回 $query(right_x, t)$。

Solution

- 对于询问 $[L, R]$，可以在线段树上分裂成 $O(\log n)$ 个 $query(x, t)$。
- 若 x 为叶子节点，那么做法显然。
- 若 $t \geq val_{left_x}$，那么显然在左儿子不会新增答案，直接返回 $query(right_x, t)$。
- 否则当 $t < val_{left_x}$ 时，经过左儿子的更新，t 一定会变成左儿子的最大值，故 $query(x, t) = query(left_x, t) + val_x - val_{left_x}$。

Solution

- 对于询问 $[L, R]$，可以在线段树上分裂成 $O(\log n)$ 个 $query(x, t)$。
- 若 x 为叶子节点，那么做法显然。
- 若 $t \geq val_{left_x}$，那么显然在左儿子不会新增答案，直接返回 $query(right_x, t)$。
- 否则当 $t < val_{left_x}$ 时，经过左儿子的更新，t 一定会变成左儿子的最大值，故 $query(x, t) = query(left_x, t) + val_x - val_{left_x}$。
- 计算单个 $query(x, t)$ 的时间复杂度为 $O(\log n)$。

Solution

- 对于询问 $[L, R]$，可以在线段树上分裂成 $O(\log n)$ 个 $query(x, t)$。
- 若 x 为叶子节点，那么做法显然。
- 若 $t \geq val_{left_x}$，那么显然在左儿子不会新增答案，直接返回 $query(right_x, t)$。
- 否则当 $t < val_{left_x}$ 时，经过左儿子的更新，t 一定会变成左儿子的最大值，故 $query(x, t) = query(left_x, t) + val_x - val_{left_x}$。
- 计算单个 $query(x, t)$ 的时间复杂度为 $O(\log n)$。
- 区间询问的时间复杂度为 $O(\log^2 n)$。

Skyscrapers

n 个建筑物从左往右排成一排，第 i 个建筑物的高度为 a_i。

Skyscrapers

n 个建筑物从左往右排成一排，第 i 个建筑物的高度为 a_i。

有一只怪兽会依次发动 m 次攻击，每次攻击它会指定一个未被摧毁的建筑物 i，然后摧毁它，并向两边释放冲击波。

Skyscrapers

n 个建筑物从左往右排成一排，第 i 个建筑物的高度为 a_i。

有一只怪兽会依次发动 m 次攻击，每次攻击它会指定一个未被摧毁的建筑物 i，然后摧毁它，并向两边释放冲击波。

往左的冲击波会从 $i-1$ 开始一直往前扩散，若碰到已被摧毁的建筑物，则冲击波会消散。否则，假设到达了 k，若 $a_i - a_k \geq |i-k|$，则建筑物 k 被摧毁。无论有没有摧毁 k，冲击波都会继续扩散。

Skyscrapers

n 个建筑物从左往右排成一排，第 i 个建筑物的高度为 a_i。

有一只怪兽会依次发动 m 次攻击，每次攻击它会指定一个未被摧毁的建筑物 i，然后摧毁它，并向两边释放冲击波。

往左的冲击波会从 $i-1$ 开始一直往前扩散，若碰到已被摧毁的建筑物，则冲击波会消散。否则，假设到达了 k，若 $a_i - a_k \geq |i-k|$，则建筑物 k 被摧毁。无论有没有摧毁 k，冲击波都会继续扩散。

向右的冲击波效果同理。输出每次攻击摧毁的建筑物个数。

Skyscrapers

n 个建筑物从左往右排成一排，第 i 个建筑物的高度为 a_i。

有一只怪兽会依次发动 m 次攻击，每次攻击它会指定一个未被摧毁的建筑物 i，然后摧毁它，并向两边释放冲击波。

往左的冲击波会从 $i-1$ 开始一直往前扩散，若碰到已被摧毁的建筑物，则冲击波会消散。否则，假设到达了 k，若 $a_i - a_k \geq |i-k|$，则建筑物 k 被摧毁。无论有没有摧毁 k，冲击波都会继续扩散。

向右的冲击波效果同理。输出每次攻击摧毁的建筑物个数。

- $1 \leq m \leq n \leq 100000, 1 \leq a_i \leq 10^9$。

Skyscrapers

n 个建筑物从左往右排成一排，第 i 个建筑物的高度为 a_i。

有一只怪兽会依次发动 m 次攻击，每次攻击它会指定一个未被摧毁的建筑物 i，然后摧毁它，并向两边释放冲击波。

往左的冲击波会从 $i-1$ 开始一直往前扩散，若碰到已被摧毁的建筑物，则冲击波会消散。否则，假设到达了 k，若 $a_i - a_k \geq |i-k|$，则建筑物 k 被摧毁。无论有没有摧毁 k，冲击波都会继续扩散。

向右的冲击波效果同理。输出每次攻击摧毁的建筑物个数。

- $1 \leq m \leq n \leq 100000, 1 \leq a_i \leq 10^9$。
- Source：XVII Open Cup named after E.V. Pankratiev. Eastern GP

Solution

- 用 set 维护所有被摧毁的建筑物，那么可以很容易地求出冲击波的作用范围 $[L, R]$。

Solution

- 用 set 维护所有被摧毁的建筑物，那么可以很容易地求出冲击波的作用范围 $[L, R]$。
- 考虑向左的冲击波，向右的同理。

Solution

- 用 set 维护所有被摧毁的建筑物，那么可以很容易地求出冲击波的作用范围 $[L, R]$。
- 考虑向左的冲击波，向右的同理。
- 若 k 会被摧毁，那么有 $a_i - a_k \geq i - k$，即 $a_k - k \leq a_i - i$。

Solution

- 用 set 维护所有被摧毁的建筑物，那么可以很容易地求出冲击波的作用范围 $[L, R]$。
- 考虑向左的冲击波，向右的同理。
- 若 k 会被摧毁，那么有 $a_i - a_k \geq i - k$，即 $a_k - k \leq a_i - i$。
- 显然区间 $[L, R]$ 内 $a_k - k$ 最小的那个建筑最容易被摧毁。

Solution

- 用 set 维护所有被摧毁的建筑物，那么可以很容易地求出冲击波的作用范围 $[L, R]$。
- 考虑向左的冲击波，向右的同理。
- 若 k 会被摧毁，那么有 $a_i - a_k \geq i - k$，即 $a_k - k \leq a_i - i$。
- 显然区间 $[L, R]$ 内 $a_k - k$ 最小的那个建筑最容易被摧毁。
- 用线段树维护区间最小值，不断取出最小值，看看是否会被摧毁。

Solution

- 用 set 维护所有被摧毁的建筑物，那么可以很容易地求出冲击波的作用范围 $[L, R]$。
- 考虑向左的冲击波，向右的同理。
- 若 k 会被摧毁，那么有 $a_i - a_k \geq i - k$，即 $a_k - k \leq a_i - i$。
- 显然区间 $[L, R]$ 内 $a_k - k$ 最小的那个建筑最容易被摧毁。
- 用线段树维护区间最小值，不断取出最小值，看看是否会被摧毁。
- 因为每个位置只会被摧毁一次，故时间复杂度为 $O(n \log n)$。

Himalayas

维护一个序列 $h_1, h_2, \dots, h_n$ ， m 次修改。

Himalayas

维护一个序列 $h_1, h_2, ..., h_n$ ，m 次修改。

每次指定区间 $[l, r]$ ，将该区间加上一个首项为 A ，公差为 $B(B > 0)$ 的等差数列。

Himalayas

维护一个序列 $h_1, h_2, ..., h_n$ ，m 次修改。

每次指定区间 $[l, r]$ ，将该区间加上一个首项为 A ，公差为 $B(B > 0)$ 的等差数列。

在每次操作之后输出所有比两侧都严格大的位置的个数。

Himalayas

维护一个序列 $h_1, h_2, ..., h_n$ ，m 次修改。

每次指定区间 $[l, r]$ ，将该区间加上一个首项为 A ，公差为 $B(B > 0)$ 的等差数列。

在每次操作之后输出所有比两侧都严格大的位置的个数。

- $1 \leq n \leq 100000$。

Himalayas

维护一个序列 $h_1, h_2, \ldots, h_n$ ，m 次修改。

每次指定区间 $[l, r]$ ，将该区间加上一个首项为 A ，公差为 $B(B > 0)$ 的等差数列。

在每次操作之后输出所有比两侧都严格大的位置的个数。

- $1 \le n \le 100000$。
- $1 \le m \le 100000$。

Himalayas

维护一个序列 $h_1, h_2, ..., h_n$ ，m 次修改。

每次指定区间 $[l, r]$ ，将该区间加上一个首项为 A ，公差为 $B(B > 0)$ 的等差数列。

在每次操作之后输出所有比两侧都严格大的位置的个数。

- $1 \le n \le 100000$。
- $1 \le m \le 100000$。
- $-100000 \le A \le 100000$。

Himalayas

维护一个序列 $h_1, h_2, ..., h_n$ ，m 次修改。

每次指定区间 $[l, r]$ ，将该区间加上一个首项为 A ，公差为 $B(B > 0)$ 的等差数列。

在每次操作之后输出所有比两侧都严格大的位置的个数。

- $1 \le n \le 100000$。
- $1 \le m \le 100000$。
- $-100000 \le A \le 100000$。
- $1 \le h_i, B \le 100000$。

Himalayas

维护一个序列 $h_1, h_2, ..., h_n$ ，m 次修改。

每次指定区间 $[l, r]$ ，将该区间加上一个首项为 A ，公差为 $B(B > 0)$ 的等差数列。

在每次操作之后输出所有比两侧都严格大的位置的个数。

- $1 \leq n \leq 100000$。
- $1 \leq m \leq 100000$。
- $-100000 \leq A \leq 100000$。
- $1 \leq h_i, B \leq 100000$。
- Source： ZOJ 3943

Solution

- 将序列进行差分，即 $a_i = h_i - h_{i-1}$，若 $a_i > 0$ 且 $a_{i+1} < 0$ 则合法。

Solution

- 将序列进行差分，即 $a_i = h_i - h_{i-1}$ ，若 $a_i > 0$ 且 $a_{i+1} < 0$ 则合法。
- 差分的好处是，区间加等差数列会变成端点单点修改，剩余部分区间加上公差。

Solution

- 将序列进行差分，即 $a_i = h_i - h_{i-1}$，若 $a_i > 0$ 且 $a_{i+1} < 0$ 则合法。
- 差分的好处是，区间加等差数列会变成端点单点修改，剩余部分区间加上公差。
- 显然只有当 a 的正负号 (包括 0) 变化时，才会对答案造成影响。

Solution

- 线段树维护区间内 a 的最大负数、0 的个数以及答案。

Solution

- 线段树维护区间内 a 的最大负数、0 的个数以及答案。
- 区间加的时候，若最大负数或者 0 的符号变化了，那么暴力递归修改，否则直接打标记。

Solution

- 线段树维护区间内 a 的最大负数、0 的个数以及答案。
- 区间加的时候，若最大负数或者 0 的符号变化了，那么暴力递归修改，否则直接打标记。
- 因为公差为正数，因此每个数的状态只会变化 $O(1)$ 次。

Solution

- 线段树维护区间内 a 的最大负数、0 的个数以及答案。
- 区间加的时候，若最大负数或者 0 的符号变化了，那么暴力递归修改，否则直接打标记。
- 因为公差为正数，因此每个数的状态只会变化 $O(1)$ 次。
- 时间复杂度 $O((n+m)\log n)$。

魔塔

有三座 n 层魔塔 A, B, C，第一次进入每层塔都需要消耗一把类型分别为 A_i, B_i, C_i 的钥匙。每层塔的经验值为 EA_i, EB_i, EC_i。

魔塔

有三座 n 层魔塔 A, B, C，第一次进入每层塔都需要消耗一把类型分别为 A_i, B_i, C_i 的钥匙。每层塔的经验值为 EA_i, EB_i, EC_i。

一共有 n 种钥匙，你有 k_i 把第 i 种钥匙，且每座魔塔不会有两层需要的钥匙种类相同。

魔塔

有三座 n 层魔塔 A, B, C，第一次进入每层塔都需要消耗一把类型分别为 A_i, B_i, C_i 的钥匙。每层塔的经验值为 EA_i, EB_i, EC_i。

一共有 n 种钥匙，你有 k_i 把第 i 种钥匙，且每座魔塔不会有两层需要的钥匙种类相同。

每座塔都必须从底向上打，但不要求打完。

魔塔

有三座 n 层魔塔 A, B, C，第一次进入每层塔都需要消耗一把类型分别为 A_i, B_i, C_i 的钥匙。每层塔的经验值为 EA_i, EB_i, EC_i。

一共有 n 种钥匙，你有 k_i 把第 i 种钥匙，且每座魔塔不会有两层需要的钥匙种类相同。

每座塔都必须从底向上打，但不要求打完。

求最多可以得到的经验值。

魔塔

有三座 n 层魔塔 A, B, C，第一次进入每层塔都需要消耗一把类型分别为 A_i, B_i, C_i 的钥匙。每层塔的经验值为 EA_i, EB_i, EC_i。

一共有 n 种钥匙，你有 k_i 把第 i 种钥匙，且每座魔塔不会有两层需要的钥匙种类相同。

每座塔都必须从底向上打，但不要求打完。

求最多可以得到的经验值。

- $1 \leq n \leq 100000$。

魔塔

有三座 n 层魔塔 A, B, C，第一次进入每层塔都需要消耗一把类型分别为 A_i, B_i, C_i 的钥匙。每层塔的经验值为 EA_i, EB_i, EC_i。

一共有 n 种钥匙，你有 k_i 把第 i 种钥匙，且每座魔塔不会有两层需要的钥匙种类相同。

每座塔都必须从底向上打，但不要求打完。

求最多可以得到的经验值。

- $1 \le n \le 100000$。
- $1 \le k_i \le 2$。

魔塔

有三座 n 层魔塔 A, B, C，第一次进入每层塔都需要消耗一把类型分别为 A_i, B_i, C_i 的钥匙。每层塔的经验值为 EA_i, EB_i, EC_i。

一共有 n 种钥匙，你有 k_i 把第 i 种钥匙，且每座魔塔不会有两层需要的钥匙种类相同。

每座塔都必须从底向上打，但不要求打完。

求最多可以得到的经验值。

- $1 \le n \le 100000$。
- $1 \le k_i \le 2$。
- $0 \le E \le 2000$。

魔塔

有三座 n 层魔塔 A, B, C，第一次进入每层塔都需要消耗一把类型分别为 A_i, B_i, C_i 的钥匙。每层塔的经验值为 EA_i, EB_i, EC_i。

一共有 n 种钥匙，你有 k_i 把第 i 种钥匙，且每座魔塔不会有两层需要的钥匙种类相同。

每座塔都必须从底向上打，但不要求打完。

求最多可以得到的经验值。

- $1 \leq n \leq 100000$。
- $1 \leq k_i \leq 2$。
- $0 \leq E \leq 2000$。
- Source：BZOJ 2130

Solution

- 考虑从 0 到 n 枚举 A 的通关楼层，同时求出每座塔的经验值前缀和 sum。

Solution

- 考虑从 0 到 n 枚举 A 的通关楼层，同时求出每座塔的经验值前缀和 sum。
- 设 f_i 表示 B 通关 i 层时 C 最多能得到多少经验，因为经验值非负，所以也可以看作最多通关多少层。

Solution

- 考虑从 0 到 n 枚举 A 的通关楼层，同时求出每座塔的经验值前缀和 sum。
- 设 f_i 表示 B 通关 i 层时 C 最多能得到多少经验，因为经验值非负，所以也可以看作最多通关多少层。
- 当 A 的通关楼层往上多 1 的时候，这把钥匙必须给 A。

Solution

- 考虑从 0 到 n 枚举 A 的通关楼层，同时求出每座塔的经验值前缀和 sum。
- 设 f_i 表示 B 通关 i 层时 C 最多能得到多少经验，因为经验值非负，所以也可以看作最多通关多少层。
- 当 A 的通关楼层往上多 1 的时候，这把钥匙必须给 A。
- 如果这把钥匙还剩 0 把，那么说明：

Solution

- 考虑从 0 到 n 枚举 A 的通关楼层，同时求出每座塔的经验值前缀和 sum。
- 设 f_i 表示 B 通关 i 层时 C 最多能得到多少经验，因为经验值非负，所以也可以看作最多通关多少层。
- 当 A 的通关楼层往上多 1 的时候，这把钥匙必须给 A。
- 如果这把钥匙还剩 0 把，那么说明：
- (1)B 某些楼层 j 以上都不能到达，对应 $f_{\geq j}$ 变为 $-inf$。

Solution

- 考虑从 0 到 n 枚举 A 的通关楼层，同时求出每座塔的经验值前缀和 sum。
- 设 f_i 表示 B 通关 i 层时 C 最多能得到多少经验，因为经验值非负，所以也可以看作最多通关多少层。
- 当 A 的通关楼层往上多 1 的时候，这把钥匙必须给 A。
- 如果这把钥匙还剩 0 把，那么说明：
- (1)B 某些楼层 j 以上都不能到达，对应 $f_{\geq j}$ 变为 $-inf$。
- (2)C 某些楼层 j 以上都不能到达，对应 f 的某个后缀与 $sumc_{j-1}$ 取 min。

Solution

- 如果这把钥匙还剩 1 把，那么说明：

Solution

- 如果这把钥匙还剩 1 把，那么说明：
- 当 B 的楼层在 j 以上时，C 不能到达 k 以上的楼层，对应 f 的某个 $\geq j$ 的后缀中与 $sumc_{k-1}$ 取 min。

Solution

- 如果这把钥匙还剩 1 把，那么说明：
- 当 B 的楼层在 j 以上时，C 不能到达 k 以上的楼层，对应 f 的某个 $\geq j$ 的后缀中与 $sumc_{k-1}$ 取 min。
- 任意时刻，随着 i 的增加，f_i 单调不上升，所以可以用线段树维护。

Solution

- 如果这把钥匙还剩 1 把，那么说明：
- 当 B 的楼层在 j 以上时，C 不能到达 k 以上的楼层，对应 f 的某个 $\geq j$ 的后缀中与 $sumc_{k-1}$ 取 min。
- 任意时刻，随着 i 的增加，f_i 单调不上升，所以可以用线段树维护。
- 线段树上每个节点维护区间内最左边、最右边的 f 以及区间内某个 f 加上 $sumb$ 的最大值。

Solution

- 如果这把钥匙还剩 1 把，那么说明：
- 当 B 的楼层在 j 以上时，C 不能到达 k 以上的楼层，对应 f 的某个 $\geq j$ 的后缀中与 $sumc_{k-1}$ 取 min。
- 任意时刻，随着 i 的增加，f_i 单调不上升，所以可以用线段树维护。
- 线段树上每个节点维护区间内最左边、最右边的 f 以及区间内某个 f 加上 $sumb$ 的最大值。
- 当修改时，可以通过上下界判断是否可以打标记，否则暴力递归左右儿子。

Solution

- 如果这把钥匙还剩 1 把，那么说明：
- 当 B 的楼层在 j 以上时，C 不能到达 k 以上的楼层，对应 f 的某个 $\geq j$ 的后缀中与 $sumc_{k-1}$ 取 min。
- 任意时刻，随着 i 的增加，f_i 单调不上升，所以可以用线段树维护。
- 线段树上每个节点维护区间内最左边、最右边的 f 以及区间内某个 f 加上 $sumb$ 的最大值。
- 当修改时，可以通过上下界判断是否可以打标记，否则暴力递归左右儿子。
- 注意到每次修改的本质其实就是 f 的一段区间赋值，故时间复杂度为 $O(n \log n)$。

打的士

维护若干个整数集合，n 次操作：

打的士

维护若干个整数集合，n 次操作：

(1) 新增一个整数集合 $\{a_1, a_2, \ldots, a_m\}$。

打的士

维护若干个整数集合，n 次操作：

(1) 新增一个整数集合 $\{a_1, a_2, ..., a_m\}$。

(2) 给定 k，找一个长度最小的区间 $[l, r]$，满足 $l \geq k$ 且 $[l, r]$ 与所有集合都有交集。

打的士

维护若干个整数集合，n 次操作：

(1) 新增一个整数集合 $\{a_1, a_2, \ldots, a_m\}$。

(2) 给定 k，找一个长度最小的区间 $[l, r]$，满足 $l \geq k$ 且 $[l, r]$ 与所有集合都有交集。

- $1 \leq n \leq 200000$。

打的士

维护若干个整数集合，n 次操作：

(1) 新增一个整数集合 $\{a_1, a_2, \ldots, a_m\}$。

(2) 给定 k，找一个长度最小的区间 $[l, r]$，满足 $l \geq k$ 且 $[l, r]$ 与所有集合都有交集。

- $1 \leq n \leq 200000$。
- $1 \leq \sum m \leq 550000$。

打的士

维护若干个整数集合，n 次操作：

(1) 新增一个整数集合 $\{a_1, a_2, \dots, a_m\}$。

(2) 给定 k，找一个长度最小的区间 $[l, r]$，满足 $l \geq k$ 且 $[l, r]$ 与所有集合都有交集。

- $1 \leq n \leq 200000$。
- $1 \leq \sum m \leq 550000$。
- $0 \leq a_i, k \leq 10^9$。

打的士

维护若干个整数集合，n 次操作：

(1) 新增一个整数集合 $\{a_1, a_2, ..., a_m\}$。

(2) 给定 k，找一个长度最小的区间 $[l, r]$，满足 $l \geq k$ 且 $[l, r]$ 与所有集合都有交集。

- $1 \leq n \leq 200000$。
- $1 \leq \sum m \leq 550000$。
- $0 \leq a_i, k \leq 10^9$。
- Source： BZOJ 3898

Solution

- 设 f_i 表示选择的 $l = i$ 时，区间长度的最小值。

Solution

- 设 f_i 表示选择的 $l = i$ 时，区间长度的最小值。
- 每来一个新集合时，设 nxt 为 i 右边最近的一个可行决策，则 $f_i = \max(f_i, nxt - i)$。

Solution

- 设 f_i 表示选择的 $l = i$ 时，区间长度的最小值。
- 每来一个新集合时，设 nxt 为 i 右边最近的一个可行决策，则 $f_i = \max(f_i, nxt - i)$。
- 注意到 f 的形式是一条条斜率为 -1 的线段，且截距单调不下降，故每次修改可以转化为对截距的区间赋值。

Solution

- 设 f_i 表示选择的 $l = i$ 时，区间长度的最小值。
- 每来一个新集合时，设 nxt 为 i 右边最近的一个可行决策，则 $f_i = \max(f_i, nxt - i)$。
- 注意到 f 的形式是一条条斜率为 -1 的线段，且截距单调不下降，故每次修改可以转化为对截距的区间赋值。
- 用线段树维护 f，对于一个区间，如果无法覆盖最左端，则返回，如果可以覆盖最右端，则打标记，否则暴力递归左右儿子。

Solution

- 设 f_i 表示选择的 $l = i$ 时，区间长度的最小值。
- 每来一个新集合时，设 nxt 为 i 右边最近的一个可行决策，则 $f_i = \max(f_i, nxt - i)$。
- 注意到 f 的形式是一条条斜率为 -1 的线段，且截距单调不下降，故每次修改可以转化为对截距的区间赋值。
- 用线段树维护 f，对于一个区间，如果无法覆盖最左端，则返回，如果可以覆盖最右端，则打标记，否则暴力递归左右儿子。
- 本质也对应着区间赋值，故时间复杂度 $O(m \log m)$。

Subtract if Greater!

维护一个正整数序列 $a_1, a_2, \ldots, a_n$ ，m 次操作：

Subtract if Greater!

维护一个正整数序列 $a_1, a_2, \ldots, a_n$ ，m 次操作：

(1) 给定 k ，将序列 a 从小到大排序，输出 a_k 的值。

Subtract if Greater!

维护一个正整数序列 $a_1, a_2, \dots, a_n$ ，m 次操作：

(1) 给定 k，将序列 a 从小到大排序，输出 a_k 的值。

(2) 给定 x，将所有严格大于 x 的数 a_i 减去 x。

Subtract if Greater!

维护一个正整数序列 $a_1, a_2, ..., a_n$ ，m 次操作：

(1) 给定 k，将序列 a 从小到大排序，输出 a_k 的值。

(2) 给定 x，将所有严格大于 x 的数 a_i 减去 x。

- $1 \leq n \leq 100000$。

Subtract if Greater!

维护一个正整数序列 $a_1, a_2, \ldots, a_n$，m 次操作：

(1) 给定 k，将序列 a 从小到大排序，输出 a_k 的值。

(2) 给定 x，将所有严格大于 x 的数 a_i 减去 x。

- $1 \leq n \leq 100000$。
- $1 \leq m \leq 1000000$。

Subtract if Greater!

维护一个正整数序列 $a_1, a_2, \ldots, a_n$ ，m 次操作：

(1) 给定 k ，将序列 a 从小到大排序，输出 a_k 的值。

(2) 给定 x ，将所有严格大于 x 的数 a_i 减去 x。

- $1 \leq n \leq 100000$。
- $1 \leq m \leq 1000000$。
- $1 \leq a_i, x \leq 10^9$。

Subtract if Greater!

维护一个正整数序列 $a_1, a_2, ..., a_n$ ，m 次操作：

(1) 给定 k，将序列 a 从小到大排序，输出 a_k 的值。

(2) 给定 x，将所有严格大于 x 的数 a_i 减去 x。

- $1 \leq n \leq 100000$。
- $1 \leq m \leq 1000000$。
- $1 \leq a_i, x \leq 10^9$。
- Source：Petrozavodsk Summer-2016. Ural FU Dandelion Contest

Solution

- 对于 k 小值查询，只需用平衡树维护序列 a。

Solution

- 对于 k 小值查询，只需用平衡树维护序列 a。
- 对于每个修改操作，$[1, x]$ 的数无需修改。

Solution

- 对于 k 小值查询，只需用平衡树维护序列 a。
- 对于每个修改操作，$[1, x]$ 的数无需修改。
- $[x+1, 2x]$ 的数会减小至少一半，暴力修改即可。

Solution

- 对于 k 小值查询，只需用平衡树维护序列 a。
- 对于每个修改操作，$[1, x]$ 的数无需修改。
- $[x+1, 2x]$ 的数会减小至少一半，暴力修改即可。
- $[2x+1, inf)$ 的数减小之后排名不变，故可以在平衡树上打标记实现。

Solution

- 对于 k 小值查询，只需用平衡树维护序列 a。
- 对于每个修改操作，$[1, x]$ 的数无需修改。
- $[x+1, 2x]$ 的数会减小至少一半，暴力修改即可。
- $[2x+1, inf)$ 的数减小之后排名不变，故可以在平衡树上打标记实现。
- 每个数若被暴力修改，那么最多只会被暴力 $O(\log a)$ 次。

Solution

- 对于 k 小值查询，只需用平衡树维护序列 a。
- 对于每个修改操作，$[1, x]$ 的数无需修改。
- $[x+1, 2x]$ 的数会减小至少一半，暴力修改即可。
- $[2x+1, inf)$ 的数减小之后排名不变，故可以在平衡树上打标记实现。
- 每个数若被暴力修改，那么最多只会被暴力 $O(\log a)$ 次。
- 时间复杂度 $O(n \log n \log a + m \log n)$。

Heavy-Light Decomposition

维护一棵 n 个点的有根二叉树，对其进行轻重链剖分，每次删除一个叶子。

Heavy-Light Decomposition

维护一棵 n 个点的有根二叉树，对其进行轻重链剖分，每次删除一个叶子。

若删除叶子之后，某个点的重儿子选择另一个点更优，那么需要改变重儿子。

Heavy-Light Decomposition

维护一棵 n 个点的有根二叉树，对其进行轻重链剖分，每次删除一个叶子。

若删除叶子之后，某个点的重儿子选择另一个点更优，那么需要改变重儿子。

求出每次删除之后每个点重儿子的编号之和。

Heavy-Light Decomposition

维护一棵 n 个点的有根二叉树，对其进行轻重链剖分，每次删除一个叶子。

若删除叶子之后，某个点的重儿子选择另一个点更优，那么需要改变重儿子。

求出每次删除之后每个点重儿子的编号之和。

- $2 \leq n \leq 200000$。

Heavy-Light Decomposition

维护一棵 n 个点的有根二叉树，对其进行轻重链剖分，每次删除一个叶子。

若删除叶子之后，某个点的重儿子选择另一个点更优，那么需要改变重儿子。

求出每次删除之后每个点重儿子的编号之和。

- $2 \le n \le 200000$。
- Source： ASC 46

Solution

- 首先用树状数组维护 DFS 序来快速支持一个点子树大小的询问。

Solution

- 首先用树状数组维护 DFS 序来快速支持一个点子树大小的询问。
- 每次删掉一个叶子时，从根开始往叶子走，显然只有 $2size_x \leq size_{father}$ 的点的父亲才有可能更换重儿子。

Solution

- 首先用树状数组维护 DFS 序来快速支持一个点子树大小的询问。
- 每次删掉一个叶子时，从根开始往叶子走，显然只有 $2size_x \le size_{father}$ 的点的父亲才有可能更换重儿子。
- 从根开始往下，找到最高的满足条件的点，从那个点开始继续迭代。

Solution

- 首先用树状数组维护 DFS 序来快速支持一个点子树大小的询问。
- 每次删掉一个叶子时，从根开始往叶子走，显然只有 $2size_x \le size_{father}$ 的点的父亲才有可能更换重儿子。
- 从根开始往下，找到最高的满足条件的点，从那个点开始继续迭代。
- 每次点数至少减小一半，所以迭代只有 $O(\log n)$ 次。

Solution

- 首先用树状数组维护 DFS 序来快速支持一个点子树大小的询问。
- 每次删掉一个叶子时，从根开始往叶子走，显然只有 $2size_x \leq size_{father}$ 的点的父亲才有可能更换重儿子。
- 从根开始往下，找到最高的满足条件的点，从那个点开始继续迭代。
- 每次点数至少减小一半，所以迭代只有 $O(\log n)$ 次。
- 利用树链剖分配合二分查找可以很容易地在 $O(\log^2 n)$ 的时间内找到迭代点。

Solution

- 首先用树状数组维护 DFS 序来快速支持一个点子树大小的询问。
- 每次删掉一个叶子时，从根开始往叶子走，显然只有 $2size_x \leq size_{father}$ 的点的父亲才有可能更换重儿子。
- 从根开始往下，找到最高的满足条件的点，从那个点开始继续迭代。
- 每次点数至少减小一半，所以迭代只有 $O(\log n)$ 次。
- 利用树链剖分配合二分查找可以很容易地在 $O(\log^2 n)$ 的时间内找到迭代点。
- 时间复杂度 $O(n \log^3 n)$，常数很小。

课后习题

- Fan Li (HDU 5320)

课后习题

- Fan Li (HDU 5320)
- Differencia (HDU 5737)

课后习题

- Fan Li (HDU 5320)
- Differencia (HDU 5737)
- Bipartite Graph (HDU 5354)

课后习题

- Fan Li (HDU 5320)
- Differencia (HDU 5737)
- Bipartite Graph (HDU 5354)
- [Sdoi2016] 游戏 (BZOJ 4515)

课后习题

- Fan Li (HDU 5320)
- Differencia (HDU 5737)
- Bipartite Graph (HDU 5354)
- [Sdoi2016] 游戏 (BZOJ 4515)
- 七彩树 (BZOJ 4771)

课后习题

- Fan Li (HDU 5320)
- Differencia (HDU 5737)
- Bipartite Graph (HDU 5354)
- [Sdoi2016] 游戏 (BZOJ 4515)
- 七彩树 (BZOJ 4771)
- dC Loves Number Theory (BZOJ 4026)

课后习题

- Fan Li (HDU 5320)
- Differencia (HDU 5737)
- Bipartite Graph (HDU 5354)
- [Sdoi2016] 游戏 (BZOJ 4515)
- 七彩树 (BZOJ 4771)
- dC Loves Number Theory (BZOJ 4026)
- The Child and Sequence (Codeforces Round #250 Div. 1 D)

课后习题

- Fan Li (HDU 5320)
- Differencia (HDU 5737)
- Bipartite Graph (HDU 5354)
- [Sdoi2016] 游戏 (BZOJ 4515)
- 七彩树 (BZOJ 4771)
- dC Loves Number Theory (BZOJ 4026)
- The Child and Sequence (Codeforces Round #250 Div. 1 D)
- Nice boat (HDU 4902)

Thank you!