根号数据结构 & 根号分治

Isaunoya

2024-07-10

1. 根号数据结构

序列分块

- 首先，这个是传统意义上的分块。
-
-
-
-

序列分块

- 首先，这个是传统意义上的分块。
- 疑似有人后面要讲分块，所以这部分可以简单一些！
-
-
-

序列分块

- 首先，这个是传统意义上的分块。
- 疑似有人后面要讲分块，所以这部分可以简单一些！
- 分块是一个块状结构。
-
-

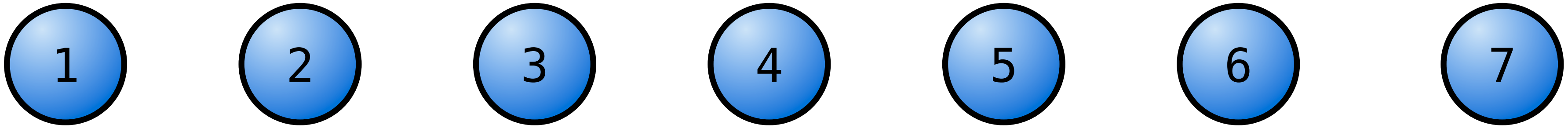
序列分块

- 首先，这个是传统意义上的分块。
- 疑似有人后面要讲分块，所以这部分可以简单一些！
- 分块是一个块状结构。
- 可以做一些简单的区间修改，区间查询的操作。
- 比如说，区间加法，区间查询和。

图解

假设一共有 7 个元素，我们设定块长为 2。（初始全都为 0。）

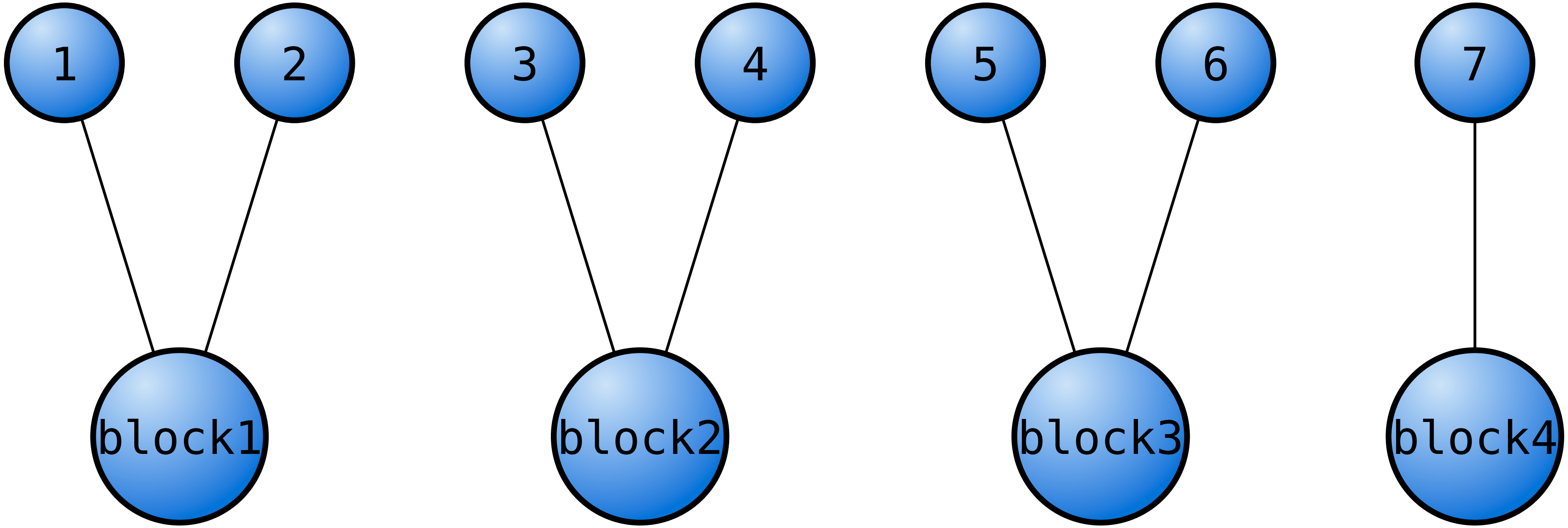
我们可以做如下划分。



图解

假设一共有 7 个元素，我们设定块长为 2。（初始都为 0。）

我们可以做如下划分。

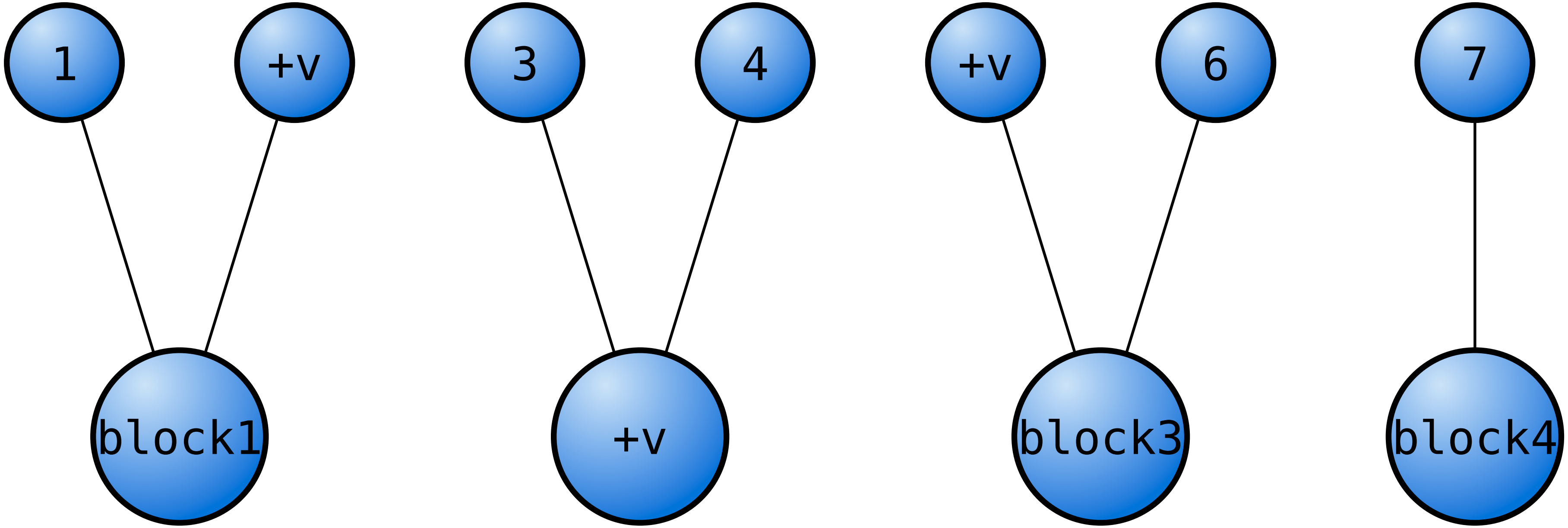


图解

假设我们要将 [2,5] 区间整体加 v，我们发现 [2,5] 区间完整的包含了 block2，以及两个散块，散块不考虑整体处理，而是直接暴力将其加 v（只要将 (2) 和 (5) 直接加），整块（只有 block2）直接加上 v。

图解

假设我们要将 [2,5] 区间整体加 v，我们发现 [2,5] 区间完整的包含了 block2，以及两个散块，散块不考虑整体处理，而是直接暴力将其加 v（只要将 (2) 和 (5) 直接加），整块（只有 block2）直接加上 v。



总结

所以其实是类似线段树的懒标记。

在做这一问题的时候可以简单的认为这就是一个根号叉的线段树。

假设有 n 个点，块大小是 B，那么我们修改的复杂度是 $O(\frac{n}{B} + B)$ 的，查询的复杂度也是 $O(\frac{n}{B} + B)$ 的。

通过均值不等式，我们可以知道复杂度最优是在 $B = \sqrt{n}$ 的时候。

试试看!

- 你有一个长度是 2^n 的序列 a,下标从 1 到 2^n.($n \le 18$, $-10^9 \le a_i \le 10^9$).
- 你需要对这个序列进行 q ($q \le 2 \cdot 10^5$) 次操作,每次操作你都会得到一个整数 k ($0 \le k \le n-1$). 你需要进行如下操作:
- 对于 $\forall i \in [1, 2^n - 2^k]$, 如果 a_i 在本次操作中已经被交换过了, 那么忽略它;否则,将 a_i 和 a_{i+2^k} 进行交换.
- 在这之后,输出序列最大子段和. 本题中,最大子段可以一个数都不选.

注意,每次操作之后不会撤销.

经典应用

- 也许你想到了一个 log 做法，但是我还是想讲根号做法。
-
-
-

经典应用

- 也许你想到了一个 log 做法，但是我还是想讲根号做法。
- 按照以往经验，得到最大子段和只需要，前缀最大值，后缀最大值，中间的最大值，这些信息合并起来就可以得到区间的最大子段和。
-
-

经典应用

- 也许你想到了一个 log 做法，但是我还是想讲根号做法。
- 按照以往经验，得到最大子段和只需要，前缀最大值，后缀最大值，中间的最大值，这些信息合并起来就可以得到区间的最大子段和。
- 其实这题也不例外，我们可以维护一个每个块的前缀，后缀，中间的最大子段和。
-

经典应用

- 也许你想到了一个 log 做法，但是我还是想讲根号做法。
- 按照以往经验，得到最大子段和只需要，前缀最大值，后缀最大值，中间的最大值，这些信息合并起来就可以得到区间的最大子段和。
- 其实这题也不例外，我们可以维护一个每个块的前缀，后缀，中间的最大子段和。
- 现在这个交换看起来非常棘手，但是你发现他是 2^k，所以那其实非常好做，因为我们可以很自然想到根号分治，将 $k = 9$ 为界，分成两个部分，也就是我取 $B = 2^9$.

第一个部分

- 我们先假设交换的都是 $k \geq 9$ 的部分。
-
-

第一个部分

- 我们先假设交换的都是 $k \geq 9$ 的部分。
- 那其实是相当于我有 $\frac{N}{B}$ 个块，需要打乱相对顺序。
-

第一个部分

- 我们先假设交换的都是 $k \geq 9$ 的部分。
- 那其实是相当于我有 $\frac{N}{B}$ 个块，需要打乱相对顺序。
- 而我每次查询的时候只需要 $O(\frac{N}{B})$ 个信息再合并即可。

第二个部分

- 第二个部分是 $k < 9$ 的部分。
-
-
-
-

第二个部分

- 第二个部分是 $k < 9$ 的部分。
- 这等价于块内交换。
-
-
-

第二个部分

- 第二个部分是 $k < 9$ 的部分。
- 这等价于块内交换。
- 但是无论你再怎么交换，块内的排布也顶多是 B 种。
-
-

第二个部分

- 第二个部分是 $k < 9$ 的部分。
- 这等价于块内交换。
- 但是无论你再怎么交换，块内的排布也顶多是 B 种。
- 也就是我可以事先预处理好，块内的排布对于所有的情况是什么样子的。
- 然后结合第一个部分就可以通过这题了。

参考代码：https://codeforces.com/contest/1716/submission/262397977

值域分块

这里是一个经典的问题。

初始有一个空集合，有两类操作：

- 插入或者删除一个自然数
- 查询集合中的 mex 是多少（mex 即为集合中最小未出现的自然数）

假设第一类操作的个数是 q_1 次。

假设第二类操作的个数是 q_2 次。

这个问题显然可以通过树状数组修改+树状数组上二分来实现。

（只有删空或者第一次添加的时候需要对树状数组做操作）

这样做的复杂度是 $O((q_1 + q_2) \log n)$。

值域分块

根号做法。

- 这题显然的也会有一个根号的做法。
-
-
-
-
-

值域分块

这里是一个经典的问题。

初始有一个空集合，有两类操作：

- 插入或者删除一个自然数
- 查询集合中的 mex 是多少（mex 即为集合中最小未出现的自然数）

假设第一类操作的个数是 q_1 次。

假设第二类操作的个数是 q_2 次。

这个问题显然可以通过树状数组修改+树状数组上二分来实现。

（只有删空或者第一次添加的时候需要对树状数组做操作）

这样做的复杂度是 $O((q_1 + q_2) \log n)$。

值域分块

根号做法。

- 这题显然的也会有一个根号的做法。
- 注意到我只需要单点修改。
- 单点修改在块里面只需要 $O(1)$ 的时间复杂度。
- 全局查询 mex 只需要 $O(\sqrt{n})$ 的复杂度，因为我需要查看每个块是否是满的，和二分的方式是类似的。
-
-

值域分块

这里是一个经典的问题。

初始有一个空集合，有两类操作：
- 插入或者删除一个自然数
- 查询集合中的 mex 是多少（mex 即为集合中最小未出现的自然数）

假设第一类操作的个数是 q_1 次。

假设第二类操作的个数是 q_2 次。

这个问题显然可以通过树状数组修改+树状数组上二分来实现。

（只有删空或者第一次添加的时候需要对树状数组做操作）

这样做的复杂度是 $O((q_1 + q_2) \log n)$。

值域分块

根号做法。

- 这题显然的也会有一个根号的做法。
- 注意到我只需要单点修改。
- 单点修改在块里面只需要 $O(1)$ 的时间复杂度。
- 全局查询 mex 只需要 $O(\sqrt{n})$ 的复杂度，因为我需要查看每个块是否是满的，和二分的方式是类似的。
- 那么我们容易注意到这个做法其实是 $O(q_1 + q_2\sqrt{n})$ 的。
- 在对于 q_1 很大而 q_2 较小的时候，我们可以采用这个根号平衡的做法。

这题就是 luogu P4137 的部分内容。会莫队的同学可以做一下！

HDU7337

多组数据，给一个长度为 n 的排列，多次查询区间 $[l,r]$ 有多少个满足 $l \le i < j \le r$ 的 (i,j) 满足 $a_i + a_j$ 是一个完全平方数。

数据范围：$T \le 5, n \le 10^5, q \le 10^5$。

做法

- 考虑到能产生贡献的只有至多 $n\sqrt{n}$ 组，因为是得是完全平方数，对于单个 a_i，不可能有超过 $\sqrt{n}$ 组和它相加是一个完全平方数。
-
-
-
-

做法

- 考虑到能产生贡献的只有至多 $n\sqrt{n}$ 组，因为是得是完全平方数，对于单个 a_i，不可能有超过 $\sqrt{n}$ 组和它相加是一个完全平方数。
- 假设 $i < j$，我们可以枚举右端点，类似 HH 的项链一样。
-
-
-

做法

- 考虑到能产生贡献的只有至多 $n\sqrt{n}$ 组，因为是得是完全平方数，对于单个 a_i，不可能有超过 $\sqrt{n}$ 组和它相加是一个完全平方数。
- 假设 $i < j$，我们可以枚举右端点，类似 HH 的项链一样。
- 假设我枚举到 r，我们计算了 $1 \leq i < j \leq r$ 的所有配对。
-
-

做法

- 考虑到能产生贡献的只有至多 $n\sqrt{n}$ 组，因为是得是完全平方数，对于单个 a_i，不可能有超过 $\sqrt{n}$ 组和它相加是一个完全平方数。
- 假设 $i < j$，我们可以枚举右端点，类似 HH 的项链一样。
- 假设我枚举到 r，我们计算了 $1 \leq i < j \leq r$ 的所有配对。
- 我们只需要将所有符合条件的点对 (i, j) 的贡献加在 i 这个点上，也就是说，如果我此时想查询 $[l, r]$ 有多少个配对，我只需要查询 $[l, r]$ 的区间和就可以。
-

做法

- 考虑到能产生贡献的只有至多 $n\sqrt{n}$ 组，因为是得是完全平方数，对于单个 a_i，不可能有超过 $\sqrt{n}$ 组和它相加是一个完全平方数。
- 假设 $i < j$，我们可以枚举右端点，类似 HH 的项链一样。
- 假设我枚举到 r，我们计算了 $1 \le i < j \le r$ 的所有配对。
- 我们只需要将所有符合条件的点对 (i, j) 的贡献加在 i 这个点上，也就是说，如果我此时想查询 $[l, r]$ 有多少个配对，我只需要查询 $[l, r]$ 的区间和就可以。
- 同样的，这个操作可以用 $n\sqrt{n}$ 次树状数组解决，这样复杂度是 $O(n\sqrt{n}\log(n))$，难以通过本题。

根号做法好啊！

- 但是我们可以通过 $O(n\sqrt{n})$ 次 $O(1)$ 修改，以及 $O(q)$ 次 $O(\sqrt{n})$ 查询，就可以把复杂度维持在根号了！
- 就完美的去掉了一个 $\log(n)$ 的复杂度。
-

根号做法好啊！

- 但是我们可以通过 $O(n\sqrt{n})$ 次 $O(1)$ 修改，以及 $O(q)$ 次 $O(\sqrt{n})$ 查询，就可以把复杂度维持在根号了！
- 就完美的去掉了一个 $\log(n)$ 的复杂度。
- 所以在复杂度不均衡的时候，通常可以采用这种“根号平衡”的技巧，否则你可能需要采用“莫队二次离线”之类的科技来实现这个东西。

不删除莫队的引入

- 这里想讲一个不删除莫队。(莫队能做的事情大概是这个的子集，并且疑似后续有人也要讲这部分。)
-
-
-

不删除莫队的引入

- 这里想讲一个不删除莫队。（莫队能做的事情大概是这个的子集，并且疑似后续有人也要讲这部分。）

引出一个例题：AT_joisc2014_c

- 给一个数组 A，给 q 次询问，每个询问查询 $[l, r]$ 中 $(c \cdot A_i)$ 的最大值。（其中 c 为 A_i 在 $[l, r]$ 出现的次数。）
-
-

不删除莫队的引入

- 这里想讲一个不删除莫队。（莫队能做的事情大概是这个的子集，并且疑似后续有人也要讲这部分。）

引出一个例题：AT_joisc2014_c

- 给一个数组 A，给 q 次询问，每个询问查询 $[l, r]$ 中 $(c \cdot A_i)$ 的最大值。（其中 c 为 A_i 在 $[l, r]$ 出现的次数。）
- 显然这个操作不可以$O(1)$撤销，如果一定要强行撤销的话，需要配合上 multiset 等工具做到$O(\log)$的撤销。
-

不删除莫队的引入

- 这里想讲一个不删除莫队。（莫队能做的事情大概是这个的子集，并且疑似后续有人也要讲这部分。）

引出一个例题：AT_joisc2014_c

- 给一个数组 A，给 q 次询问，每个询问查询 $[l,r]$ 中 $(c \cdot A_i)$ 的最大值。（其中 c 为 A_i 在 $[l,r]$ 出现的次数。）
- 显然这个操作不可以$O(1)$撤销，如果一定要强行撤销的话，需要配合上 multiset 等工具做到$O(\log)$的撤销。
- 下面我们引入回滚莫队这一工具来帮我们解决这个问题。

不删除莫队的具体写法

- 定义 q_i 是询问，回滚莫队的做法大概就是:
- 分类讨论一下,如果左右指针共处一个块内,直接暴力,就是根号级别的,这种就不用放进 q_i 里面。
- 否则把询问丢到左端点的块里,在同一个块里的,按右端点升序排序，假设块是 $[L, R]$。
-
-
-

不删除莫队的具体写法

- 定义 q_i 是询问，回滚莫队的做法大概就是:
- 分类讨论一下,如果左右指针共处一个块内,直接暴力,就是根号级别的,这种就不用放进 q_i 里面。
- 否则把询问丢到左端点的块里,在同一个块里的,按右端点升序排序，假设块是 $[L, R]$。
- 然后对于每个块求解，由于右端点是递增的，考虑移动右端点。并同时记录 $[R+1, r]$ 的答案。
- 左边的贡献直接从 $[q[i].l, R]$，暴力就行了，这样就能得到 q_i 的答案。
- 由于不能删除，每次在询问之前记录状态，然后复制一遍，再操作，最后回退到状态 $[R+1, r]$。

不删除莫队的复杂度

- 考虑首先你有 $\frac{N}{B}$ 个块，那么对于每个块，都要移动到最右侧（这是最坏情况）。
-
-

不删除莫队的复杂度

- 考虑首先你有 $\frac{N}{B}$ 个块，那么对于每个块，都要移动到最右侧（这是最坏情况）。
- 然后，我们要考虑每个询问，一个询问只需要特殊处理块内的部分，这部分是 QB
-

不删除莫队的复杂度

- 考虑首先你有 $\frac{N}{B}$ 个块，那么对于每个块，都要移动到最右侧（这是最坏情况）。
- 然后，我们要考虑每个询问，一个询问只需要特殊处理块内的部分，这部分是 QB
- 所以分析出来总复杂度是 $O\left(\frac{N^2}{B} + QB\right)$ 的。

取块大小为 $\sqrt{\frac{N^2}{Q}}$ 理论最优，实际上要看常数。

CF736E

- 给你 n 个点，已知 m 对关系 $[u,v](|u-v| \le k)$，k 给出，询问 q 次，每次问你 $[l,r]$ 有多少个连通块。

$n,q \le 10^5, k \le 5, m \le 5 \cdot 10^5$

做法

- 其实我感觉是裸题。（狗头）
-
-
-

做法

- 其实我感觉是裸题。(狗头)
- 感觉这题的问题在于复原部分，就是需要一个可撤销并查集。
- 而这个只是并查集需要按秩合并/不路径压缩就可以完成的事情。
-

做法

- 其实我感觉是裸题。（狗头）
- 感觉这题的问题在于复原部分，就是需要一个可撤销并查集。
- 而这个只是并查集需要按秩合并/不路径压缩就可以完成的事情。
- 记录一下改了哪些点，然后把块内暴力的部分回退即可。

不删除莫队题目合集

- https://www.luogu.com.cn/problem/P5906
- https://www.luogu.com.cn/problem/AT_joisc2014_c
- https://www.luogu.com.cn/problem/CF763E

2. 根号分治

根号分治引入（CF797E）

- 给定长度为 n 的序列 a，q 次询问。
- 每次询问给出 p, k。您要不断地执行操作 $p \leftarrow p + a_p + k$，直到 $p > n$ 为止。询问的答案为操作次数。
- $1 \leq n, q \leq 10^5$，$1 \leq a_i \leq n$，$1 \leq p, k \leq n$。

如何用根号分治解决本题

- 注意到这是一个比较显然的根号分治。
-
-

如何用根号分治解决本题

- 注意到这是一个比较显然的根号分治。
- 因为当 $k \geq \sqrt{n}$ 的时候，无论 a_i 具体是多少，答案至多是 $\frac{n}{k} \leq \sqrt{n}$
-

如何用根号分治解决本题

- 注意到这是一个比较显然的根号分治。
- 因为当 $k \geq \sqrt{n}$ 的时候，无论 a_i 具体是多少，答案至多是 $\frac{n}{k} \leq \sqrt{n}$
- 当 $k < \sqrt{n}$ 的时候，这样的 k 只会有 $\sqrt{n}$ 种，直接预处理出来每个位置的答案即可（从后面转移到前面即可。）

代码：https://codeforces.com/contest/797/submission/97404205

除此之外有什么作用呢？（ABC359G）

给你一棵有 N 个顶点的树。i 这条边双向连接顶点 u_i 和 v_i 。

此外，还给出了一个整数序列 $A = (A_1, ..., A_N)$ 。

定义 $f(i, j)$ 如下：如果是 $A_i = A_j$ ，那么 $f(i, j)$ 就是从顶点 i 移动到顶点 j 所需的最小边数。如果是 $A_i \neq A_j$ ，那么就是 $f(i, j) = 0$ 。

计算下面表达式的值：

$\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} f(i, j)$

- $2 \leq N \leq 2 \cdot 10^5, 1 \leq u_i, v_i \leq N, 1 \leq A_i \leq N$

简单的虚树做法

- 首先注意到，如果 A_i 全部相同，那么可以直接计算每条边的贡献，也就是 $size_i \cdot (n - size_i)$，$size_i$ 表示 i 的子树大小。
-
-
-

简单的虚树做法

- 首先注意到，如果 A_i 全部相同，那么可以直接计算每条边的贡献，也就是 $size_i \cdot (n - size_i)$，$size_i$ 表示 i 的子树大小。
- 然后我们只需要计算颜色为 c 的部分，也就是说，我只要把颜色都为 c 的一些点，建成一棵树，然后加入一些虚点（虚点是为了让那些点联通），组成一颗虚树。
-
-

简单的虚树做法

- 首先注意到，如果 A_i 全部相同，那么可以直接计算每条边的贡献，也就是 $size_i \cdot (n - size_i)$，$size_i$ 表示 i 的子树大小。
- 然后我们只需要计算颜色为 c 的部分，也就是说，我只要把颜色都为 c 的一些点，建成一棵树，然后加入一些虚点（虚点是为了让那些点联通），组成一颗虚树。
- 虚树的边 (u, v)，需要加权，也就是原树上的 $dist(u, v)$，因为这条虚边实际上代表了 $dist(u, v)$ 条边，但是贡献系数都相同，可以一起拿来计算。然后现在的 $size_i$ 其实是表示颜色 c 在 i 子树出现多少次。（也就是不能计算虚点进去）
-

简单的虚树做法

- 首先注意到，如果 A_i 全部相同，那么可以直接计算每条边的贡献，也就是 $size_i \cdot (n - size_i)$，$size_i$ 表示 i 的子树大小。
- 然后我们只需要计算颜色为 c 的部分，也就是说，我只要把颜色都为 c 的一些点，建成一棵树，然后加入一些虚点（虚点是为了让那些点联通），组成一颗虚树。
- 虚树的边 (u, v)，需要加权，也就是原树上的 $dist(u, v)$，因为这条虚边实际上代表了 $dist(u, v)$ 条边，但是贡献系数都相同，可以一起拿来计算。然后现在的 $size_i$ 其实是表示颜色 c 在 i 子树出现多少次。（也就是不能计算虚点进去）
- 按照上面的做法，我们已经做完了这个问题。只需要一个虚树。

但是能不能更简单呢！

可以的！

-
-
-

但是能不能更简单呢！

可以的！

- 首先考虑设定一个阈值 B。
-
-

但是能不能更简单呢！

可以的！

- 首先考虑设定一个阈值 B。
- 对于颜色出现次数 $cnt \le B$ 的，我们可以采用 cnt^2 的做法，这一部分的复杂度是 $\sum cnt^2 \le \sum cnt \cdot B \le n \cdot B$
-

但是能不能更简单呢！

可以的！

- 首先考虑设定一个阈值 B。
- 对于颜色出现次数 $cnt \leq B$ 的，我们可以采用 cnt^2 的做法，这一部分的复杂度是 $\sum cnt^2 \leq \sum cnt \cdot B \leq n \cdot B$
- 剩下的是颜色 c 出现次数 $cnt > B$ 的，这种颜色不会超过 $\frac{n}{B}$ 个，对于这种，我们可以采用整棵树都遍历一遍，也就是，对于考虑 $size_i$ 定义成 i 子树内 c 的出现次数，然后遍历一整棵树，计算 $\sum size_i \times (size_1 - size_i)$ 即可。

代码链接

- 根号分治：https://atcoder.jp/contests/abc359/submissions/54856187
- 虚树：https://atcoder.jp/contests/abc359/submissions/54829502

例题 1

https://www.codechef.com/problems/MONSTER

- 大厨正在玩一个打怪兽的小游戏。游戏中初始时有 n 只怪兽排成一排，从左到右编号为
- 0~$n-1$。第 i 只怪兽的初始血量为 h_i，当怪兽的血量小于等于 0 时，这只怪兽就挂了。
- 大厨要进行 q 次操作。每次操作中，大厨会选择两个整数 x 和 y，并向下标 k 满足 $k\&x=k$ 的怪兽开炮（此处 & 代表按位与操作）。被炮弹打到的怪兽会掉 y 点血。
- 请告诉大厨，在他每次操作后，还有多少怪兽活着。

做法

- 首先问题可以转化成，每个怪兽什么时候死掉，然后做一个前缀和就可以解决这个问题。
-
-
-

做法

- 首先问题可以转化成，每个怪兽什么时候死掉，然后做一个前缀和就可以解决这个问题。
- 如何知道一个怪兽什么时候死掉呢。
-
-

做法

- 首先问题可以转化成，每个怪兽什么时候死掉，然后做一个前缀和就可以解决这个问题。
- 如何知道一个怪兽什么时候死掉呢。
- 引入另一个问题：给一个序列 A_i，要找到第一个前缀和大于等于 S 的位置。（使用分块）
-

做法

- 首先问题可以转化成，每个怪兽什么时候死掉，然后做一个前缀和就可以解决这个问题。
- 如何知道一个怪兽什么时候死掉呢。
- 引入另一个问题：给一个序列 A_i，要找到第一个前缀和大于等于 S 的位置。（使用分块）
- 这个要如何处理，就是考虑在第 i 个块内能不能达到 S 这个值，如果达到了，再去块里面找到更精确的位置。

那这个题到底应该咋做？

- 其实和刚才引入的问题类似。

那这个题到底应该咋做？

- 其实和刚才引入的问题类似。
- 考虑每个怪物在一个块里面会受到多少伤害（这个可以通过高维前缀和解决）。
- 然后如果某个怪物 i 在这个块里面死了，我们直接暴力算这个 i 在具体什么时候死掉的。
-
-

那这个题到底应该咋做？

- 其实和刚才引入的问题类似。
- 考虑每个怪物在一个块里面会受到多少伤害（这个可以通过高维前缀和解决）。
- 然后如果某个怪物 i 在这个块里面死了，我们直接暴力算这个 i 在具体什么时候死掉的。
- 这样对于每个块，我们计算的复杂度是 $O(n\log n + q)$
-

那这个题到底应该咋做？

- 其实和刚才引入的问题类似。
- 考虑每个怪物在一个块里面会受到多少伤害（这个可以通过高维前缀和解决）。
- 然后如果某个怪物 i 在这个块里面死了，我们直接暴力算这个 i 在具体什么时候死掉的。
- 这样对于每个块，我们计算的复杂度是 $O(n \log n + q)$
- 对于每个怪物，我们计算的复杂度其实是 $O(B)$

那这个题到底应该咋做？

- 其实和刚才引入的问题类似。
- 考虑每个怪物在一个块里面会受到多少伤害（这个可以通过高维前缀和解决）。
- 然后如果某个怪物 i 在这个块里面死了，我们直接暴力算这个 i 在具体什么时候死掉的。
- 这样对于每个块，我们计算的复杂度是 $O(n \log n + q)$
- 对于每个怪物，我们计算的复杂度其实是 $O(B)$

那么最终复杂度是 $O\left(\frac{n}{B}(n \log n + q) + nB\right)$

例题 2

百度之星决赛

谢谢大家！