分治与分块

陈嘉乐

北京大学信息科学技术学院

基本分治

分治算法

分治算法

- 基本思想

分治算法

- 基本思想
  - 当我们求解某些问题时，由于这些问题要处理的数据相当多，或求解过程相当复杂，使得直接求解法在时间上相当长，或者根本无法直接求出。对于这类问题，我们往往先把它分解成几个子问题，找到求出这几个子问题的解法后，再找到合适的方法，把它们组合成求整个问题的解法。

分治算法

- 基本思想
  - 当我们求解某些问题时，由于这些问题要处理的数据相当多，或求解过程相当复杂，使得直接求解法在时间上相当长，或者根本无法直接求出。对于这类问题，我们往往先把它分解成几个子问题，找到求出这几个子问题的解法后，再找到合适的方法，把它们组合成求整个问题的解法。
  - 如果这些子问题还较大，难以解决，可以再把它们分成几个更小的子问题，以此类推，直至可以直接求出解为止。

分治算法

- 基本思想
  - 当我们求解某些问题时，由于这些问题要处理的数据相当多，或求解过程相当复杂，使得直接求解法在时间上相当长，或者根本无法直接求出。对于这类问题，我们往往先把它分解成几个子问题，找到求出这几个子问题的解法后，再找到合适的方法，把它们组合成求整个问题的解法。
  - 如果这些子问题还较大，难以解决，可以再把它们分成几个更小的子问题，以此类推，直至可以直接求出解为止。
  - 利用分治策略求解时，所需时间取决于分解后子问题的个数、子问题的规模大小等因素，而二分法，由于其划分的简单和均匀的特点，是经常采用的一种有效的方法。

例题

归并排序
给定一个长度为 n 的序列，请将其中的元素按从小到大的顺序输出。

$n \leq 10^5$

例题

归并排序
给定一个长度为 n 的序列，请将其中的元素按从小到大的顺序输出。

$n \leq 10^5$

- 分解问题：将整个序列分为前后两部分，分别求解。

例题

归并排序
给定一个长度为 n 的序列，请将其中的元素按从小到大的顺序输出。

$n \leq 10^5$

- 分解问题：将整个序列分为前后两部分，分别求解。
- 求解子问题：当序列的长度已经为 1 的时候，自然是有序的。

例题

归并排序
给定一个长度为 n 的序列，请将其中的元素按从小到大的顺序输出。

$n \leq 10^5$

- 分解问题：将整个序列分为前后两部分，分别求解。
- 求解子问题：当序列的长度已经为 1 的时候，自然是有序的。
- 合并子问题的解：该怎么合并两个有序数组呢？

树分治

为什么要讲树分治

为什么要讲树分治

- 树相对于一般图有特殊的性质，在竞赛中有更广泛的应用。

为什么要讲树分治

- 树相对于一般图有特殊的性质，在竞赛中有更广泛的应用。
- 关于树上路径的问题更是频繁出现在各类比赛之中。

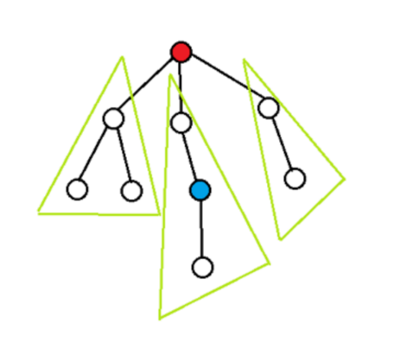
为什么要讲树分治

- 树相对于一般图有特殊的性质，在竞赛中有更广泛的应用。
- 关于树上路径的问题更是频繁出现在各类比赛之中。
- 树分治是用来解决一类树上路径问题的高效算法。

点分治

基于点的分治

首先选择一个点，将无根树变为有根树，再递归处理每一棵以根节点的儿子为根的子树。



如何选点

如何选点

- 因为是递归的，所以我们希望递归的层数尽量少。

如何选点

- 因为是递归的，所以我们希望递归的层数尽量少。
- 选择一个点使得删去这个点之后，剩余结点数量最多的连通块结点数量最少。我们将这个点称为“重心”。

如何选点

- 因为是递归的，所以我们希望递归的层数尽量少。
- 选择一个点使得删去这个点之后，剩余结点数量最多的连通块结点数量最少。我们将这个点称为“重心”。
- 重心可以通过动态规划求得。

如何选点

- 因为是递归的，所以我们希望递归的层数尽量少。
- 选择一个点使得删去这个点之后，剩余结点数量最多的连通块结点数量最少。我们将这个点称为“重心”。
- 重心可以通过动态规划求得。
    - dfs 一遍求得以每个点为根的子树大小。

如何选点

- 因为是递归的，所以我们希望递归的层数尽量少。
- 选择一个点使得删去这个点之后，剩余结点数量最多的连通块结点数量最少。我们将这个点称为“重心”。
- 重心可以通过动态规划求得。
  - dfs 一遍求得以每个点为根的子树大小。
  - 选一个点 u 为根并删去后，结点最多的联通块结点个数为 $\max\{size[v_1], \cdots, size[v_m], n - size[u]\}$，其中 v_i 为 u 的儿子节点。

如何选点

- 因为是递归的，所以我们希望递归的层数尽量少。
- 选择一个点使得删去这个点之后，剩余结点数量最多的连通块结点数量最少。我们将这个点称为“重心”。
- 重心可以通过动态规划求得。
  - dfs 一遍求得以每个点为根的子树大小。
  - 选一个点 u 为根并删去后，结点最多的联通块结点个数为 $\max\{size[v_1], \cdots, size[v_m], n - size[u]\}$，其中 v_i 为 u 的儿子节点。
  - 从中选择最优的一个节点。

如何选点

- 因为是递归的，所以我们希望递归的层数尽量少。
- 选择一个点使得删去这个点之后，剩余结点数量最多的连通块结点数量最少。我们将这个点称为“重心”。
- 重心可以通过动态规划求得。
  - dfs 一遍求得以每个点为根的子树大小。
  - 选一个点 u 为根并删去后，结点最多的联通块结点个数为 $\max\{size[v_1], \cdots, size[v_m], n - size[u]\}$，其中 v_i 为 u 的儿子节点。
  - 从中选择最优的一个节点。
- 总复杂度 $O(n)$。

效率分析

效率分析

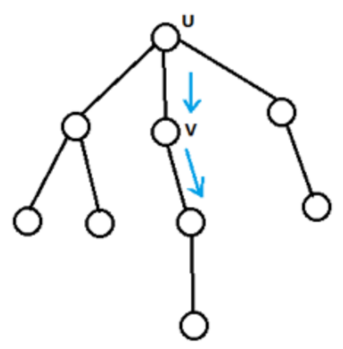
- 删去重心后剩余联通块的大小不超过 $\frac{1}{2}n$。

效率分析

- 删去重心后剩余联通块的大小不超过 $\frac{1}{2}n$。
- 为什么?

效率分析

- 删去重心后剩余联通块的大小不超过 $\frac{1}{2}n$。
- 为什么?



效率分析

效率分析

- 基于以上理论，每次分治之后，联通块大小至少减少一半。

效率分析

- 基于以上理论，每次分治之后，联通块大小至少减少一半。
- 递归深度为 $O(\log n)$ 层。

效率分析

- 基于以上理论，每次分治之后，联通块大小至少减少一半。
- 递归深度为 $O(\log n)$ 层。
- 因此总复杂度为 $O(n \log n)$。

例题 1

IOI2011 Race
给定一棵 n 个点的有边权的树，求一条路径使得权值和为 K 且边的数量最小。

$1 \leq n \leq 10^5, K \leq 10^6$

例题 1

IOI2011 Race
给定一棵 n 个点的有边权的树，求一条路径使得权值和为 K 且边的数量最小。

$1 \leq n \leq 10^5, K \leq 10^6$

- 考虑树上路径表示在点分树中的形式。

例题 1

IOI2011 Race
给定一棵 n 个点的有边权的树，求一条路径使得权值和为 K 且边的数量最小。

$1 \leq n \leq 10^5, K \leq 10^6$

- 考虑树上路径表示在点分树中的形式。

1. 某一点到根的路径。

例题 1

IOI2011 Race
给定一棵 n 个点的有边权的树，求一条路径使得权值和为 K 且边的数量最小。

$1 \leq n \leq 10^5, K \leq 10^6$

- 考虑树上路径表示在点分树中的形式。

1. 某一点到根的路径。
2. 来自根节点不同儿子所在子树的两个点的路径。

例题 1

IOI2011 Race
给定一棵 n 个点的有边权的树，求一条路径使得权值和为 K 且边的数量最小。

$1 \le n \le 10^5, K \le 10^6$

- 考虑树上路径表示在点分树中的形式。

1. 某一点到根的路径。
2. 来自根节点不同儿子所在子树的两个点的路径。

- 归纳起来就是一切经过根的路径。

例题 1

IOI2011 Race
给定一棵 n 个点的有边权的树，求一条路径使得权值和为 K 且边的数量最小。

$1 \le n \le 10^5, K \le 10^6$

- 考虑树上路径表示在点分树中的形式。

1. 某一点到根的路径。
2. 来自根节点不同儿子所在子树的两个点的路径。

- 归纳起来就是一切经过根的路径。
- 本题中只要记录到根的长度为 x 的路径最小边数是多少即可。

边分治

基于边的分治

基于边的分治

- 在树中选择一条边，统计与这条边有关的信息。

基于边的分治

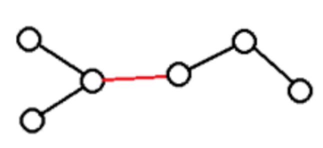
- 在树中选择一条边，统计与这条边有关的信息。
- 将这条边删去后得到两棵不相交的子树。

基于边的分治

- 在树中选择一条边，统计与这条边有关的信息。
- 将这条边删去后得到两棵不相交的子树。
- 递归处理。

基于边的分治

- 在树中选择一条边，统计与这条边有关的信息。
- 将这条边删去后得到两棵不相交的子树。
- 递归处理。



如何选边

如何选边

- 回顾点分治选点的策略：选择一个点使得删去这个点之后，剩余结点数量最多的连通块结点数量最少。

如何选边

- 回顾点分治选点的策略：选择一个点使得删去这个点之后，剩余结点数量最多的连通块结点数量最少。
- 同样地，选择的边要保证将其删去后，节点数最多的联通块的结点数量最少。

如何选边

- 回顾点分治选点的策略：选择一个点使得删去这个点之后，剩余结点数量最多的连通块结点数量最少。
- 同样地，选择的边要保证将其删去后，节点数最多的联通块的结点数量最少。
- 也可以用动态规划来实现，做法和找重心基本一样。

效率分析

效率分析

- 不妨设 D 为所有点的度数最大值。

效率分析

- 不妨设 D 为所有点的度数最大值。
- 当 $D > 1$ 时，我们设最优方案是边 (u, v)，且以 u, v 为根的两棵子树的结点个数分别为 $s, n - s$，不妨设 $s \geq n - s$。

效率分析

- 不妨设 D 为所有点的度数最大值。
- 当 $D > 1$ 时，我们设最优方案是边 (u, v)，且以 u, v 为根的两棵子树的结点个数分别为 $s, n - s$，不妨设 $s \geq n - s$。
- 设 x 为 u 的儿子中以 x 为根的子树中结点个数最大的一个。

效率分析

- 不妨设 D 为所有点的度数最大值。
- 当 $D > 1$ 时，我们设最优方案是边 (u, v)，且以 u, v 为根的两棵子树的结点个数分别为 $s, n - s$，不妨设 $s \geq n - s$。
- 设 x 为 u 的儿子中以 x 为根的子树中结点个数最大的一个。
- 考虑另一个方案 (u, x)，设以 x 为根的子树节点数为 p，显然有 $p \geq \frac{s-1}{D-1}$，由于 $p < s$，且 (u, v) 为最优方案，所以 $n - p \geq s$，联立后可以得到 $s \leq \frac{(D-1)n+1}{D}$

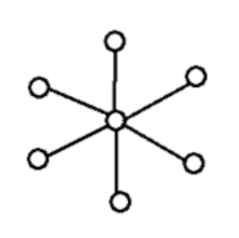
效率分析

效率分析

- 由此可见，当 D 为常数是递归深度为 $O(\log n)$ 的。

效率分析

- 由此可见，当 D 为常数是递归深度为 $O(\log n)$ 的。
- 但碰到一般的图时，D 可以达到 $O(n)$ 级别，这时候的算法效率非常的低。



改进边分治的复杂度

改进边分治的复杂度

- 我们要做的实际上是使得 D 变为常数。

改进边分治的复杂度

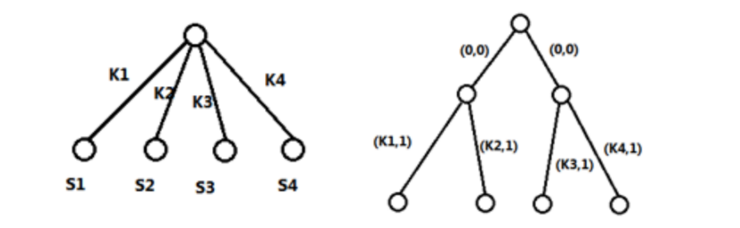
- 我们要做的实际上是使得 D 变为常数。
- 考虑增加虚点。

改进边分治的复杂度

- 我们要做的实际上是使得 D 变为常数。
- 考虑增加虚点。
- 如下图所示的等价转换，使得每个点的度数至多为 3，新树点数不超过 $2n$。

改进边分治的复杂度

- 我们要做的实际上是使得 D 变为常数。
- 考虑增加虚点。
- 如下图所示的等价转换，使得每个点的度数至多为 3，新树点数不超过 $2n$。
- 这样边分治的递归层数降为 $O(\log n)$ 级别。



例题

BZOJ2870
给定一棵 n 个点的带边权的树，求树上的一条链使得链长与链最小值乘积最大。

$1 \leq n \leq 100000$

链分治（树链剖分）

基于链的分治

基于链的分治

- 链分治能够支持链修改，链查询甚至子树查询的功能。

基于链的分治

- 链分治能够支持链修改，链查询甚至子树查询的功能。
- 链分治的结构，就是将树上的路径分为重链和轻链，并用线段树，平衡树等数据结构维护重链的信息。

基于链的分治

名词解释

基于链的分治

名词解释

- 重儿子：子树中节点数最多的儿子。

基于链的分治

名词解释

- 重儿子：子树中节点数最多的儿子。
- 轻儿子：除了重儿子之外的儿子结点。

基于链的分治

名词解释

- 重儿子：子树中节点数最多的儿子。
- 轻儿子：除了重儿子之外的儿子结点。
- 重边：与重儿子的连边。

基于链的分治

名词解释

- 重儿子：子树中节点数最多的儿子。
- 轻儿子：除了重儿子之外的儿子结点。
- 重边：与重儿子的连边。
- 轻边：与轻儿子的连边。

基于链的分治

名词解释

- 重儿子：子树中节点数最多的儿子。
- 轻儿子：除了重儿子之外的儿子结点。
- 重边：与重儿子的连边。
- 轻边：与轻儿子的连边。
- 重链：由重边连成的路径。

基于链的分治

名词解释

- 重儿子：子树中节点数最多的儿子。
- 轻儿子：除了重儿子之外的儿子结点。
- 重边：与重儿子的连边。
- 轻边：与轻儿子的连边。
- 重链：由重边连成的路径。
- 轻链：轻边。

效率分析

效率分析

- 剖分后的树有良好的性质：如果 (u, v) 为轻边，u 是 v 的儿子，则 u 的子树大小小于 v 子树大小的一半。

效率分析

- 剖分后的树有良好的性质：如果 (u, v) 为轻边，u 是 v 的儿子，则 u 的子树大小小于 v 子树大小的一半。
- 从而我们得到从根到某一点的路径上轻链、重链的个数都不大于 $\log n$。

效率分析

- 剖分后的树有良好的性质：如果 (u, v) 为轻边，u 是 v 的儿子，则 u 的子树大小小于 v 子树大小的一半。
- 从而我们得到从根到某一点的路径上轻链、重链的个数都不大于 $\log n$。
- 由此可得，任意两个点之间的轻链、重链个数都是 $O(\log n)$ 级别的。

CDQ 分治 & 整体二分

什么是在线和离线?

什么是在线和离线?

- 在线算法：可以以序列化的方式一个一个的处理输入，不必事先知道所有输入数据。

什么是在线和离线?

- 在线算法：可以以序列化的方式一个一个的处理输入，不必事先知道所有输入数据。
- 离线算法：必须事先知道所有的输入数据。

而 CDQ 分治和整体二分都是离线算法。

CDQ 分治思想

- 普通分治的流程：对于区间 $[L, R]$，递归解决 $[L, M], [M+1, R]$ 后将两个区间的答案合并即得到答案。

CDQ 分治思想

- 普通分治的流程：对于区间 $[L, R]$，递归解决 $[L, M], [M+1, R]$ 后将两个区间的答案合并即得到答案。
- 而 CDQ 分治的流程：对于区间 $[L, R]$，递归解决 $[L, M], [M+1, R]$ 后将两个区间的答案合并再加上 $[L, M]$ 对 $[M+1, R]$ 的贡献即得到答案。

例题

二维偏序

给定 N 个有序对 (a, b)，求对于每个 (a, b)，满足 $a_2 < a$ 且 $b_2 < b$ 的有序对 (a_2, b_2) 有多少个。

例题

二维偏序

给定 N 个有序对 (a, b)，求对于每个 (a, b)，满足 $a_2 < a$ 且 $b_2 < b$ 的有序对 (a_2, b_2) 有多少个。

例题

二维偏序

给定 N 个有序对 (a, b)，求对于每个 (a, b)，满足 $a_2 < a$ 且 $b_2 < b$ 的有序对 (a_2, b_2) 有多少个。

- 首先将这 N 个有序对按 a 为第一关键字从小到大，b 为第二关键字从大到小排序。

例题

二维偏序

给定 N 个有序对 (a, b)，求对于每个 (a, b)，满足 $a_2 < a$ 且 $b_2 < b$ 的有序对 (a_2, b_2) 有多少个。

- 首先将这 N 个有序对按 a 为第一关键字从小到大，b 为第二关键字从大到小排序。
- 对应上述的流程，将 $[L, R]$ 的问题分为 $[L, M], [M+1, R]$ 的问题解决，直到区间大小为 1。

例题

二维偏序

给定 N 个有序对 (a,b)，求对于每个 (a,b)，满足 $a_2 < a$ 且 $b_2 < b$ 的有序对 (a_2,b_2) 有多少个。

- 首先将这 N 个有序对按 a 为第一关键字从小到大，b 为第二关键字从大到小排序。
- 对应上述的流程，将 $[L,R]$ 的问题分为 $[L,M],[M+1,R]$ 的问题解决，直到区间大小为 1。
- 重点在于如何计算 $[L,M]$ 对 $[M+1,R]$ 的贡献。

例题

二维偏序

给定 N 个有序对 (a, b)，求对于每个 (a, b)，满足 $a_2 < a$ 且 $b_2 < b$ 的有序对 (a_2, b_2) 有多少个。

- 首先将这 N 个有序对按 a 为第一关键字从小到大，b 为第二关键字从大到小排序。
- 对应上述的流程，将 $[L, R]$ 的问题分为 $[L, M], [M+1, R]$ 的问题解决，直到区间大小为 1。
- 重点在于如何计算 $[L, M]$ 对 $[M+1, R]$ 的贡献。
- 由于子区间内部的贡献已经计算完毕，我们不妨对这两个区间都按 b 权值从小到大排序。考虑到我们最初的排序方式，因此在 $[L, M]$ 中的 (u, v) 如果对 $[M+1, R]$ 中的 (a, b) 产生贡献，等价于 $v < b$。因此使用双指针扫一遍即可。

例题

二维偏序
给定 N 个有序对 (a,b)，求对于每个 (a,b)，满足 $a_2 < a$ 且 $b_2 < b$ 的有序对 (a_2,b_2) 有多少个。

- 首先将这 N 个有序对按 a 为第一关键字从小到大，b 为第二关键字从大到小排序。
- 对应上述的流程，将 $[L,R]$ 的问题分为 $[L,M],[M+1,R]$ 的问题解决，直到区间大小为 1。
- 重点在于如何计算 $[L,M]$ 对 $[M+1,R]$ 的贡献。
- 由于子区间内部的贡献已经计算完毕，我们不妨对这两个区间都按 b 权值从小到大排序。考虑到我们最初的排序方式，因此在 $[L,M]$ 中的 (u,v) 如果对 $[M+1,R]$ 中的 (a,b) 产生贡献，等价于 $v < b$。因此使用双指针扫一遍即可。
- 然后我们考虑这个子区间内按照 b 排序的过程，实际上不必在每一层递归的时候都排序，而是将两个子区间排完序的有序对组归并即可。

整体二分思想

使用整体二分的题要满足以下性质：

整体二分思想

使用整体二分的题要满足以下性质：

- 询问的答案具有可二分性。

整体二分思想

使用整体二分的题要满足以下性质：

- 询问的答案具有可二分性。
- 修改对询问的贡献是独立的，相互之间不影响。

整体二分思想

使用整体二分的题要满足以下性质：

- 询问的答案具有可二分性。
- 修改对询问的贡献是独立的，相互之间不影响。
- 不同的修改的贡献是可叠加的，不必重复计算。

整体二分思想

使用整体二分的题要满足以下性质：

- 询问的答案具有可二分性。
- 修改对询问的贡献是独立的，相互之间不影响。
- 不同的修改的贡献是可叠加的，不必重复计算。
- 贡献满足交换律、结合律、有可加性。

整体二分思想

使用整体二分的题要满足以下性质：

- 询问的答案具有可二分性。
- 修改对询问的贡献是独立的，相互之间不影响。
- 不同的修改的贡献是可叠加的，不必重复计算。
- 贡献满足交换律、结合律、有可加性。
- 题目允许离线。

具体实现

具体实现

- 顾名思义，对所有的询问一起二分。

具体实现

- 顾名思义，对所有的询问一起二分。
- 通常而言，这类题的询问是类似乎第几次修改后满足条件，因此二分的就是修改序列，至多二分 $O(\log m)$ 层。

具体实现

- 顾名思义，对所有的询问一起二分。
- 通常而言，这类题的询问是类似乎第几次修改后满足条件，因此二分的就是修改序列，至多二分 $O(\log m)$ 层。
- 看一道例题来理解一下。

例题

Meteors

有 n 个国家和 m 个空间站，每个空间站都属于一个国家，一个国家可以有多个空间站，所有空间站按照顺序形成一个环，也就是说，m 号空间站和 1 号空间站相邻。现在，将会有 k 场流星雨降临，每一场流星雨都会给区间 $[l_i, r_i]$ 内的每个空间站带来 a_i 单位的陨石，每个国家都有一个收集陨石的目标 p_i，即第 i 个国家需要收集 p_i 单位的陨石。

询问：每个国家最早完成陨石收集目标是在第几场流星雨过后。

$1 \leq n, m, k \leq 300000$

例题

Meteors
有 n 个国家和 m 个空间站，每个空间站都属于一个国家，一个国家可以有多个空间站，所有空间站按照顺序形成一个环，也就是说，m 号空间站和 1 号空间站相邻。现在，将会有 k 场流星雨降临，每一场流星雨都会给区间 $[l_i, r_i]$ 内的每个空间站带来 a_i 单位的陨石，每个国家都有一个收集陨石的目标 p_i，即第 i 个国家需要收集 p_i 单位的陨石。

询问：每个国家最早完成陨石收集目标是在第几场流星雨过后。

$1 \leq n, m, k \leq 300000$

例题

Meteors
有 n 个国家和 m 个空间站，每个空间站都属于一个国家，一个国家可以有多个空间站，所有空间站按照顺序形成一个环，也就是说，m 号空间站和 1 号空间站相邻。现在，将会有 k 场流星雨降临，每一场流星雨都会给区间 $[l_i, r_i]$ 内的每个空间站带来 a_i 单位的陨石，每个国家都有一个收集陨石的目标 p_i，即第 i 个国家需要收集 p_i 单位的陨石。

询问：每个国家最早完成陨石收集目标是在第几场流星雨过后。

$1 \le n, m, k \le 300000$

- 二分答案，假设当前二分的区间是 $[L, R]$，答案在 $[L, R]$ 区间中的询问是 $A_1, \cdots, A_l$。

例题

Meteors

有 n 个国家和 m 个空间站，每个空间站都属于一个国家，一个国家可以有多个空间站，所有空间站按照顺序形成一个环，也就是说，m 号空间站和 1 号空间站相邻。现在，将会有 k 场流星雨降临，每一场流星雨都会给区间 $[l_i, r_i]$ 内的每个空间站带来 a_i 单位的陨石，每个国家都有一个收集陨石的目标 p_i，即第 i 个国家需要收集 p_i 单位的陨石。

询问：每个国家最早完成陨石收集目标是在第几场流星雨过后。

$1 \le n, m, k \le 300000$

- 二分答案，假设当前二分的区间是 $[L, R]$，答案在 $[L, R]$ 区间中的询问是 $A_1, \cdots, A_l$。
- 用线段树模拟 $[L, M]$ 区间中的修改。

例题

Meteors
有 n 个国家和 m 个空间站，每个空间站都属于一个国家，一个国家可以有多个空间站，所有空间站按照顺序形成一个环，也就是说，m 号空间站和 1 号空间站相邻。现在，将会有 k 场流星雨降临，每一场流星雨都会给区间 $[l_i, r_i]$ 内的每个空间站带来 a_i 单位的陨石，每个国家都有一个收集陨石的目标 p_i，即第 i 个国家需要收集 p_i 单位的陨石。

询问：每个国家最早完成陨石收集目标是在第几场流星雨过后。

$1 \leq n, m, k \leq 300000$

- 二分答案，假设当前二分的区间是 $[L, R]$，答案在 $[L, R]$ 区间中的询问是 $A_1, \cdots, A_l$。
- 用线段树模拟 $[L, M]$ 区间中的修改。
- 枚举每一个询问，达到 p_i 要求的答案应在 $[L, M]$ 区间中，否则在 $[M+1, R]$ 中，继续二分下去。

例题

Meteors

有 n 个国家和 m 个空间站，每个空间站都属于一个国家，一个国家可以有多个空间站，所有空间站按照顺序形成一个环，也就是说，m 号空间站和 1 号空间站相邻。现在，将会有 k 场流星雨降临，每一场流星雨都会给区间 $[l_i, r_i]$ 内的每个空间站带来 a_i 单位的陨石，每个国家都有一个收集陨石的目标 p_i，即第 i 个国家需要收集 p_i 单位的陨石。

询问：每个国家最早完成陨石收集目标是在第几场流星雨过后。

$1 \le n, m, k \le 300000$

- 二分答案，假设当前二分的区间是 $[L, R]$，答案在 $[L, R]$ 区间中的询问是 $A_1, \cdots, A_l$。
- 用线段树模拟 $[L, M]$ 区间中的修改。
- 枚举每一个询问，达到 p_i 要求的答案应在 $[L, M]$ 区间中，否则在 $[M+1, R]$ 中，继续二分下去。
- 当二分的区间长度为 1 时结束递归。

复杂度分析

复杂度分析

- 每一层的询问总个数都是 m，算上线段树的复杂度 $O(m\log m)$。

复杂度分析

- 每一层的询问总个数都是 m，算上线段树的复杂度 $O(m \log m)$。
- 二分共 $O(\log k)$ 层。

复杂度分析

- 每一层的询问总个数都是 m，算上线段树的复杂度 $O(m \log m)$。
- 二分共 $O(\log k)$ 层。
- 总复杂度为 $O(m \log m \log k)$。

三分

三分

三分

- 回想二分是用来解决一类单调的问题的，而三分则是用来解决凸函数上最值的问题。

三分

- 回想二分是用来解决一类单调的问题的，而三分则是用来解决凸函数上最值的问题。
- 例如，具体来讲，给定一个严格下凸函数 $f(x)$，如何求 $[L, R]$ 区间中的最小值?

三分

- 回想二分是用来解决一类单调的问题的，而三分则是用来解决凸函数上最值的问题。
- 例如，具体来讲，给定一个严格下凸函数 $f(x)$，如何求 $[L, R]$ 区间中的最小值?
- 将区间三等分为 $[L, M_1], [M_1, M_2], [M_2, R]$。如果 $f(M_1) \leq f(M_2)$，由下凸函数的性质，我们知道 $[M_2, R]$ 这段区间上的函数值一定大于 $f(M_2)$，因此不会取到最小值，因此答案区间缩小为 $[L, M_2]$，长度变为原来的 $\frac{2}{3}$。

三分

- 回想二分是用来解决一类单调的问题的，而三分则是用来解决凸函数上最值的问题。
- 例如，具体来讲，给定一个严格下凸函数 $f(x)$，如何求 $[L, R]$ 区间中的最小值?
- 将区间三等分为 $[L, M_1], [M_1, M_2], [M_2, R]$。如果 $f(M_1) \leq f(M_2)$，由下凸函数的性质，我们知道 $[M_2, R]$ 这段区间上的函数值一定大于 $f(M_2)$，因此不会取到最小值，因此答案区间缩小为 $[L, M_2]$，长度变为原来的 $\frac{2}{3}$。
- 当 $f(M_1) > f(M_2)$ 同理。

三分

- 回想二分是用来解决一类单调的问题的，而三分则是用来解决凸函数上最值的问题。
- 例如，具体来讲，给定一个严格下凸函数 $f(x)$，如何求 $[L, R]$ 区间中的最小值?
- 将区间三等分为 $[L, M_1], [M_1, M_2], [M_2, R]$。如果 $f(M_1) \leq f(M_2)$，由下凸函数的性质，我们知道 $[M_2, R]$ 这段区间上的函数值一定大于 $f(M_2)$，因此不会取到最小值，因此答案区间缩小为 $[L, M_2]$，长度变为原来的 $\frac{2}{3}$。
- 当 $f(M_1) > f(M_2)$ 同理。
- 这样在 $O(\log C)$ 的时间内就能达到题目所需要的精度要求。

分块

分块思想

分块思想

- 将连续 x 个元素分为一个块。

分块思想

- 将连续 x 个元素分为一个块。
- 对于每一个操作 $[L, R]$，可以表示为 $[L, K*X], [K*X+1, (K+1)*X], \cdots, [K_2*X+1, R]$ 的形式，即中间包含了若干完整块，两边剩余不超过 $2x$ 个位置。

分块思想

- 将连续 x 个元素分为一个块。
- 对于每一个操作 $[L,R]$，可以表示为 $[L,K*X],[K*X+1,(K+1)*X],\cdots,[K_2*X+1,R]$ 的形式，即中间包含了若干完整块，两边剩余不超过 $2x$ 个位置。
- 对于完整块可以统一处理，如果可以做到 $O(1)$, 时间复杂度 $O(\frac{n}{x})$

分块思想

- 将连续 x 个元素分为一个块。
- 对于每一个操作 $[L, R]$，可以表示为 $[L, K*X], [K*X+1, (K+1)*X], \cdots, [K_2*X+1, R]$ 的形式，即中间包含了若干完整块，两边剩余不超过 $2x$ 个位置。
- 对于完整块可以统一处理，如果可以做到 $O(1)$, 时间复杂度 $O(\frac{n}{x})$
- 对于剩余的点，可以暴力处理 $O(x)$。

分块思想

- 将连续 x 个元素分为一个块。
- 对于每一个操作 $[L, R]$，可以表示为 $[L, K*X], [K*X+1, (K+1)*X], \cdots, [K_2*X+1, R]$ 的形式，即中间包含了若干完整块，两边剩余不超过 $2x$ 个位置。
- 对于完整块可以统一处理，如果可以做到 $O(1)$, 时间复杂度 $O(\frac{n}{x})$
- 对于剩余的点，可以暴力处理 $O(x)$。
- 取 $x=\sqrt{n}$ 时，每次操作的时间复杂度降到最低 $O(\sqrt{n})$

分块思想

- 将连续 x 个元素分为一个块。
- 对于每一个操作 $[L, R]$，可以表示为 $[L, K*X], [K*X+1, (K+1)*X], \cdots, [K_2*X+1, R]$ 的形式，即中间包含了若干完整块，两边剩余不超过 $2x$ 个位置。
- 对于完整块可以统一处理，如果可以做到 $O(1)$, 时间复杂度 $O(\frac{n}{x})$
- 对于剩余的点，可以暴力处理 $O(x)$。
- 取 $x = \sqrt{n}$ 时，每次操作的时间复杂度降到最低 $O(\sqrt{n})$
- 若上面两种情况的任意一种需要多一个 log 的复杂度，那么可以通过调整块的大小来使得每次操作的复杂度变为 $O(\sqrt{n \log n})$

例题

Example

n 个数，q 个操作。

操作 1: 将一段区间内每个数增加 x。

操作 2: 询问一段区间内数的总和。

$n, q \leq 10^5$。

例题

Example
n 个数，q 个操作。

操作 1: 将一段区间内每个数增加 x。

操作 2: 询问一段区间内数的总和。

$n, q \leq 10^5$。

例题

Example

n 个数，q 个操作。

操作 1: 将一段区间内每个数增加 x。

操作 2: 询问一段区间内数的总和。

$n, q \leq 10^5$。

- 按照前面所说的，将序列每 x 个分一块。

例题

Example
n 个数，q 个操作。

操作 1: 将一段区间内每个数增加 x。

操作 2: 询问一段区间内数的总和。

$n, q \leq 10^5$。

- 按照前面所说的，将序列每 x 个分一块。
- 修改操作：

例题

Example

n 个数，q 个操作。

操作 1: 将一段区间内每个数增加 x。

操作 2: 询问一段区间内数的总和。

$n, q \leq 10^5$。

- 按照前面所说的，将序列每 x 个分一块。
- 修改操作：
    - 对于完整的块，每个块记一个标记 t，表示该块内每个元素的值都需要加上 t。

例题

Example

n 个数，q 个操作。

操作 1: 将一段区间内每个数增加 x。

操作 2: 询问一段区间内数的总和。

$n, q \le 10^5$。

- 按照前面所说的，将序列每 x 个分一块。
- 修改操作：
  - 对于完整的块，每个块记一个标记 t，表示该块内每个元素的值都需要加上 t。
  - 对于多余的 $2x$ 个位置，直接暴力修改值。

例题

Example

n 个数，q 个操作。

操作 1: 将一段区间内每个数增加 x。

操作 2: 询问一段区间内数的总和。

$n, q \leq 10^5$。

- 按照前面所说的，将序列每 x 个分一块。
- 修改操作：
  - 对于完整的块，每个块记一个标记 t，表示该块内每个元素的值都需要加上 t。
  - 对于多余的 $2x$ 个位置，直接暴力修改值。
- 询问操作：

例题

Example

n 个数，q 个操作。

操作 1: 将一段区间内每个数增加 x。

操作 2: 询问一段区间内数的总和。

$n, q \leq 10^5$。

- 按照前面所说的，将序列每 x 个分一块。
- 修改操作：
  - 对于完整的块，每个块记一个标记 t，表示该块内每个元素的值都需要加上 t。
  - 对于多余的 $2x$ 个位置，直接暴力修改值。
- 询问操作：
  - 对于完整的块，返回块内元素总和 + $t*$ 块大小。

例题

Example

n 个数，q 个操作。

操作 1: 将一段区间内每个数增加 x。

操作 2: 询问一段区间内数的总和。

$n, q \leq 10^5$。

- 按照前面所说的，将序列每 x 个分一块。
- 修改操作：
    - 对于完整的块，每个块记一个标记 t，表示该块内每个元素的值都需要加上 t。
    - 对于多余的 $2x$ 个位置，直接暴力修改值。
- 询问操作：
    - 对于完整的块，返回块内元素总和 $+\ t*$ 块大小。
    - 对于多余的 $2x$ 个位置，直接暴力累加。

例题

Example

n 个数，q 个操作。

操作 1: 将一段区间内每个数增加 x。

操作 2: 询问一段区间内数的总和。

$n, q \le 10^5$。

- 按照前面所说的，将序列每 x 个分一块。
- 修改操作：
  - 对于完整的块，每个块记一个标记 t，表示该块内每个元素的值都需要加上 t。
  - 对于多余的 $2x$ 个位置，直接暴力修改值。
- 询问操作：
  - 对于完整的块，返回块内元素总和 + $t*$ 块大小。
  - 对于多余的 $2x$ 个位置，直接暴力累加。
- 由于两种情况都可以 $\sqrt{n}$ 完成，所以 x 设为 $\sqrt{n}$，总时间复杂度 $O(q\sqrt{n})$

莫队算法

莫队

莫队

- 对于无修改的题目，如果知道区间 $[l, r]$ 的答案可以快速算出 $[l, r+1], [l, r-1], [l+1, r], [l-1, r]$ 的答案的题目，可以套用这一通用解法。

莫队

- 对于无修改的题目，如果知道区间 $[l, r]$ 的答案可以快速算出 $[l, r+1], [l, r-1], [l+1, r], [l-1, r]$ 的答案的题目，可以套用这一通用解法。
- 先将序列分成 $\sqrt{n}$ 分块，然后将所有询问做双关键字排序，第一关键字为询问的左端点所在的块，第二关键字为询问的右端点

莫队

- 对于无修改的题目，如果知道区间 $[l, r]$ 的答案可以快速算出 $[l, r+1], [l, r-1], [l+1, r], [l-1, r]$ 的答案的题目，可以套用这一通用解法。
- 先将序列分成 $\sqrt{n}$ 分块，然后将所有询问做双关键字排序，第一关键字为询问的左端点所在的块，第二关键字为询问的右端点
- 那么两个询问 $[l_1, r_1], [l_2, r_2]$ 之间转移的时间为 $(|l_1 - l_2| + |r_1 - r_2|) * T$，其中 T 为移动一步的复杂度。

复杂度分析

复杂度分析

- 对于左端点，在同一块内的转移，一次不超过 $\sqrt{n}$，在不同块之间转移不超过 $\sqrt{n}$ 次。

复杂度分析

- 对于左端点，在同一块内的转移，一次不超过 $\sqrt{n}$，在不同块之间转移不超过 $\sqrt{n}$ 次。
- 对于右端点，在同一类内，右端点是单调不降的，所以同一类内最多转移 n 次，在不同类之间转移不超过 $\sqrt{n}$ 次。

复杂度分析

- 对于左端点，在同一块内的转移，一次不超过 $\sqrt{n}$，在不同块之间转移不超过 $\sqrt{n}$ 次。
- 对于右端点，在同一类内，右端点是单调不降的，所以同一类内最多转移 n 次，在不同类之间转移不超过 $\sqrt{n}$ 次。
- 那么总转移次数就是 $O(n\sqrt{n})$ 级别。

树上莫队算法

树上莫队

树上莫队

- 解决树上问题的一个很重要的方法就是转化为序列上的问题。

树上莫队

- 解决树上问题的一个很重要的方法就是转化为序列上的问题。
- 借助 dfs 序（括号序列）就可以做到这一点。

例题

SPOJ COT2
给定一棵 n 个点的有颜色的树。

m 次查询，每次询问 (u, v) 路径上的颜色种类。

$1 \leq n, m \leq 100000$

块状链表

块状链表

块状链表

- 数组定位的复杂度为 $O(1)$, 插入删除的复杂度是 $O(n)$。

块状链表

- 数组定位的复杂度为 $O(1)$, 插入删除的复杂度是 $O(n)$。
- 链表定位的复杂度是 $O(n)$, 插入删除的复杂度是 $O(1)$。

块状链表

- 数组定位的复杂度为 $O(1)$, 插入删除的复杂度是 $O(n)$。
- 链表定位的复杂度是 $O(n)$, 插入删除的复杂度是 $O(1)$。
- 块状数组结合了链表和数组的优点，使得所有操作的复杂度均为 $O(\sqrt{n})$。

块状链表

- 数组定位的复杂度为 $O(1)$, 插入删除的复杂度是 $O(n)$。
- 链表定位的复杂度是 $O(n)$, 插入删除的复杂度是 $O(1)$。
- 块状数组结合了链表和数组的优点，使得所有操作的复杂度均为 $O(\sqrt{n})$。
- 块状链表从宏观上看是链表，而链表中的每个节点又是一个数组。

块状链表

- 数组定位的复杂度为 $O(1)$, 插入删除的复杂度是 $O(n)$。
- 链表定位的复杂度是 $O(n)$, 插入删除的复杂度是 $O(1)$。
- 块状数组结合了链表和数组的优点，使得所有操作的复杂度均为 $O(\sqrt{n})$。
- 块状链表从宏观上看是链表，而链表中的每个节点又是一个数组。
- 注意插入删除后要及时合并小的块。

例题

BZOJ3337

输入格式	说明	示例 （例如原序列为 5 2 6 3 1 4）
1 x val	**在第 x 个数后插入一个 val** （0≤x≤序列长度，val>0）	输入：1 2 7 序列：5 2 **7** 6 3 1 4
2 x	**删除第 x 个数** （1≤x≤序列长度）	输入：2 1 序列：**2 6 3 1 4**
3 x y	**翻转第 x 至第 y 个数** （1≤x≤y≤序列长度）	输入：3 3 5 序列：5 2 **1 3 6** 4
4 x y k	**将第 x 至第 y 个数旋转（向右移动）k 次** （1≤x≤y≤序列长度，1≤k≤y-x）	输入：4 1 6 1 序列：**4 5 2 6 3 1**
5 x y val	**将第 x 至第 y 个数加上 val** （1≤x≤y≤序列长度，val>0）	输入：5 3 4 5 序列：5 2 **11 8** 1 4
6 x y val	**将第 x 至第 y 个数都修改为 val** （1≤x≤y≤序列长度，val>0）	输入：6 1 4 7 序列：**7 7 7 7** 1 4
7 x y	**询问第 x 至第 y 个数的和** （1≤x≤y≤序列长度）	输入：7 2 4 输出：**11**
8 x y	**询问第 x 至第 y 个数中最大值与最小值的差** （1≤x≤y≤序列长度）	输入：8 1 3 输出：**4**
9 x y val	**询问第 x 至第 y 个数中与 val 的差的绝对值的最小值** （1≤x≤y≤序列长度，val>0）	输入：9 2 4 5 输出：**1**
10 x y k	**询问第 x 至第 y 个数中第 k 小的数** （1≤x≤y≤序列长度，1≤k≤y-x+1）	输入：10 1 6 4 输出：**4**
11 x y val	**询问第 x 至第 y 个数中比 val 小的数的个数** （1≤x≤y≤序列长度，val>0）	输入：11 2 5 4 输出：**3**