浅谈分块思想在一类数据处理问题中的应用

北京市第八十中学　罗剑桥

摘 要

在竞赛和实践中，如何高效地组织和处理大规模数据是一个重要问题。本文主要讨论竞赛中一类与线性序列和树形结构有关的数据处理问题，引入分块思想的概念，探讨分块思想在数据的组织，数据的预处理，以及设计离线算法中的应用，并举例说明如何将数据分块处理而得到时间复杂度低的优秀算法。

1　引言

在一类数据处理问题中，我们需要频繁地统计在规模庞大的数据集合中，满足一定条件的元素的有关信息，如数目或者最值等。传统的数据结构，如线段树，将集合中的所有数据按一定的方式有序地组织起来，使得每次询问可以划分成若干有序部分的信息的和，从而提高效率。

本文介绍一种分块思想，同样是将数据有序化和层次化，但方式有所不同，而且扩展性更强，应用范围更广泛。

分块思想的核心是将一个集合划分成若干个规模较小的子集。一般地，我们倾向于将性质接近的元素分到同一个子集中，并且把每个子集的元素按一定的结构组织起来。

在数据处理问题中，分块思想有着广泛的应用，原因是分块的方法具有一些良好的性质：

若子集规模很小，对每个子集可使用关于子集规模的复杂度较高的算法。

若子集数目很少，可使用与整个集合规模有关的算法分别处理每个子集[3]。

[3]容易发现前两条性质之间存在矛盾。为了均衡，既不能让子集规模过大，也不能让子集数

若每个子集内部的元素具有共同的性质，可以分别设计有针对性的算法。

本文主要讨论分块思想在与线性序列和树形结构有关的数据处理问题中的应用。第 2 节介绍在典型的数据结构中分块的一般方法。第 3 节介绍分块思想在数据的预处理中的应用。第 4 节介绍分块思想在设计离线算法时的应用。

2　数据的分块化

这一节介绍两种常见的数据结构即线性序列和树形结构的一般的分块方法，并分别给出应用分块方法解决的例题。

2.1　线性序列的分块化

在线性序列中，数据集合的所有元素按一定顺序排列。典型的问题是频繁地统计一个连续的子序列中满足一定性质的元素的和，此处“和”表示答案能够直接或间接地分拆成不同部分的答案的和。下面先介绍一种常用的分块方法，然后举例说明如何使用分块的方法高效地解决此类问题。

分块的方法　从序列的第一个元素起，每连续的 S 个元素组成一个块。若最终剩余的元素不足 S 个，令它们组成一个块。

性质2.1. *设 N 为序列的长度，则上述方法中块的数目不超过 $\lfloor N / S \rfloor + 1$。*

性质2.2. *设一次询问操作查询序列中第 L 个元素到第 R 个元素的连续子序列（以下简称区间 $[L, R]$）的信息。那么除了完整包含在内的块中的元素以外，区间 $[L, R]$ 涉及的元素的数目不超过 $2S$。*

根据以上两条性质，如果能够在块的层次维护块内所有元素的信息的和，令块的大小 S 取合适的值，对于任意区间，就能快速找到对应的完整的块与多余的元素，并合并得到整个区间的信息。

目过大。这就是为什么分块算法的时间复杂度往往与 $\sqrt{N}$ 有关的原因。

例题一[4]

有一个 N 个数的序列。每个数都是整数。你需要执行 M 次操作。操作有两种类型:

– $ADD\ D_i\ X_i$ 从第一个数开始，每隔 D_i 个位置的数增加 X_i。

– $QUERY\ L_i\ R_i$ 回答当前序列第 L_i 项到第 R_i 项的和。

数据规模: $1 \leq N,\ M \leq 10^5$，任何时刻数列中每个数的绝对值不超过 10^9。

算法分析

最坏情况下，每次 ADD 操作和 $QUERY$ 操作涉及的元素的数目会达到 N 个，所以单纯的模拟会非常慢。考虑将序列分块，在块的层次记录每个块内所有元素的和。

引理2.1. *若块的大小设为 $\sqrt{N}$* [5]*，则每次查询的时间复杂度是 $O(\sqrt{N})$。*

具体的做法是在查询时枚举每个块，若完整包含在查询的区间内则答案直接加上此块的元素的和。而剩余的元素数目不超过 $2\sqrt{N}$，可直接枚举求和。

不过，修改操作的复杂度依然没有改进。这时还是要用到分块的思想。我们注意到一个事实：

引理2.2. *当且仅当 ADD 操作中 $D_i < \sqrt{N}$ 时，需要更新的元素的数目超过 $\sqrt{N}$ 个。*

因此将 ADD 操作分类：当 $D_i \geq \sqrt{N}$ 时，可以直接更新涉及的元素以及块的和，复杂度是 $O(\sqrt{N})$；当 $D_i < \sqrt{N}$ 时，我们可以只用数组记录每个 D_i 的 ADD 操作的修改量的和，此时更新操作是 $O(1)$ 的。

这时记录的每个元素以及块的和都仅考虑了第一类的修改操作。在查询的时候，我们还要枚举第二类的所有 D_i，并计算每个 D_i 的修改量的和对答案的影响。由于第二类的 D_i 不超过 $\sqrt{N}$ 个，一次查询的时间复杂度依然是 $O(\sqrt{N})$。

[4]**题目来源：原创**

[5]为了方便，文中使用 $\sqrt{N}$ 表示不小于 $\sqrt{N}$ 的最小整数，即 $\lceil\sqrt{N}\rceil$。

定理2.3. 可以在 $O((N+M)\sqrt{N})$ 的时间复杂度内解决此问题。

2.2 树形结构的分块化

在树形结构中，相邻的点之间有很强的相关性。一个直观的想法是希望将树划分成若干个规模较小的连通块。下面介绍一种树的分块方法，对于适当的 S，此方法能够将树划分为 $\Theta(N\ /\ S)$ 个大小为 $\Theta(S)$ 的连通块。其中 N 表示树中的点的数目。

首先引入有根树的 DFS 序列的概念。无根树可以将任意一个点作为根从而转化为有根树。

定义2.1 (树的 DFS 序列)**.** 维护一个序列，最初是空的。从根开始进行深度优先遍历，每遇到一个点进栈或出栈就把它加入序列的末尾。遍历结束后得到的最终序列称为 DFS 序列。

这样，对任意一个树我们能得到一个含有 $2N$ 个元素的 DFS 序列，树中的每个点对应 DFS 序列中的两个元素。DFS 序列具有良好的性质。

引理2.4. DFS 序列中相邻的两个元素的关系仅有三种：要么是同一个点，要么是父子关系，要么是兄弟关系。最后一种关系成立当且仅当 DFS 时前一个元素是出栈的点而后一个元素是入栈的点。

根据引理 2.4，可以证明下面一条性质。

引理2.5. DFS 序列中的第 u 项到第 v 项的连续子序列恰好对应从第 u 项对应的点 D_u 到第 v 项对应的点 D_v 的一条路径。路径上的点除了 D_u 和 D_v 的最近公共祖先（以下简称为 LCA）以外都至少在子序列中出现了一次。

于是得到我们想要的结论：

定理2.6. 使用上一节介绍的分块方法将树的 DFS 序列分成 S 块，那么序列中每一块的所有元素对应的点以及它们的 LCA 就是树上的一个连通块。连通块的数目不超过 $2N\ /\ S$。

这里介绍的分块方法实现起来很简单，也具有分块思想的一般性质即每个块的规模并不大而且块的数目并不多[6]。下面我们来看一道有关的例题。

[6]遗憾的是存在某些点出现在不止 2 个连通块中。

例题二[7]

给出一个 N 个点的树和一个整数 K。每条边有权值。你需要计算对于每个结点 i，其它 $N-1$ 个点到点 i 的距离中第 K 小的值是多少。

数据规模: $1 \leq K < N \leq 50\,000$，边的权值是绝对值小于 $1\,000$ 的整数。

算法分析[8]

朴素的算法是依次将每个点作为根，计算其它 $N-1$ 个点到它的距离，排序以后即可得到答案。时间复杂度是 $O(N^2 \log N)$[9]。

我们将树按之前描述的方法分成若干个大小为 $\Theta(S)$ 的连通块。考虑利用相邻的点之间的相关性更快地计算出以一个连通块内的每个点为根时的答案。

引理2.7. *设点 u 是不在当前连通块内的任意一个点。无论以块内的哪个点作为根，从点 u 到根的路径上经过的第一个块内的点是不变的。*

根据引理 2.7，将不在当前块内的所有点按到根的路径上经过的第一个块内的点（以下简称最近块内祖先）分类。显然，不在块内的点被分成至多 S 类。

引理2.8. *设点 u 和点 v 是不在当前连通块内且属于同一类的任意两个点。设点 p 是点 u 和点 v 的最近块内祖先。则以连通块内的任意一个点为根时，点 u 到根的距离不大于点 v 到根的距离当且仅当点 u 到点 p 的距离不大于点 v 到点 p 的距离。*

首先，将树遍历一遍，计算出每个不在块内的点到最近块内祖先的距离。对每一类的所有点，把它们按到最近块内祖先的距离从小到大排序。接下来，枚举连通块内的每个点作为根，分别计算答案。

定理2.9. *以连通块内的任意一个点 r_i 为根时，可以在 $O(S \log^2 N)$ 的时间复杂度内算出其它点到点 r_i 的距离中第 K 小的值。*

[7]**题目来源：经典问题**

[8]**这道题目有时间复杂度更优但十分复杂的树链剖分的解法，此处介绍的解法为笔者原创**

[9]如果使用基数排序，可以将复杂度降为 $O(N^2)$。

具体的做法如下。设当前以点 r_i 为根。为计算其它点到根 r_i 的距离中第 K 小的值，我们使用二分答案的方法。

设二分的答案为 X 。从点 r_i 开始遍历块内的每个点，计算它们到点 r_i 的距离，并统计其中不超过 X 的值有多少个。然后对块内的每个点 p ，设点 p 到根 r_i 的距离为 D_p，在以点 p 为最近块内祖先的一类点（按到点 p 的距离排序）的有序表中二分查找不大于 $X - D_p$ 的最大的数的位置，从而计算出这一类不在块内的点中与根 r_i 的距离不超过 X 的点的数目。

这样，设（除点 r_i 外）到点 r_i 的距离不超过 X 的点的数目等于 cnt。若 $cnt < K$，说明实际答案大于 X；否则说明实际答案不大于 X。于是每次二分可以将实际答案的范围缩小一半，从而以 $O(S\log^2 N)$ 的时间复杂度算出其它点到点 r_i 的距离中第 K 小的值。

考虑排序的时间复杂度和连通块内的点的数目，

引理2.10. *可以在 $O(N\log N + S^2\log^2 N)$ 的时间复杂度内计算出一个连通块内的每个点的答案。*

定理2.11. *如果令块的大小为 $\Theta(\sqrt{N/\log N})$ ，此算法总的时间复杂度为 $O(N^{1.5}\log^{1.5} N)$。*

2.3 进一步扩展

有兴趣的读者可以自行思考线性序列的分块方法如何推广到 D 维的情况，以及树的分块方法是否适用于更复杂的结构。

3 分块与预处理

预处理是指维护一个数据集合时，在所有询问操作以前对数据进行一些处理，往往记录许多附加信息，在询问时利用已处理过的信息提高时间效率[10]。

如果将分块思想与预处理相结合，能够高效地解决一类范围很广的集合查询问题，这就是本节将讨论的内容。这一节首先说明此类可以使用预处理优化的问题要满足的性质，然后详细介绍在序列和树结构中预处理的一般方法。

[10]如果有修改操作，还要考虑修改对预处理的信息的影响。

3.1 预处理的条件

设维护的数据集合为集合 S。

条件3.1 (可增量). 设两次询问的集合分别为集合 S 的子集 A 和子集 B。则从询问集合 A 的相关信息转化为询问集合 B 的相关信息的复杂度是关于集合 A 与集合 B 的对称差（即 $(A-B)\bigcup(B-A)$）的势的函数。即此转移的复杂度仅与对称差含有的元素的数目有关。

设集合 T 为所有询问可能涉及的集合 S 的子集的集合。

条件3.2 (可预处理). 对于给定的整数 K 和 L，存在一个规模不超过 K 的集合 T 的子集 P，使得对于集合 T 的任意一个元素 X，存在集合 T 的某个元素 Y 属于集合 P 并且集合 Y 与集合 X 的对称差的势不超过 L。

如果对数据集合的询问满足条件 3.1，并且存在合适的 K 和 L 使得数据集合满足条件 3.2，就可以使用分块和预处理的方法提高查询操作的时间效率。

此时我们找到条件 3.2 中的集合 P，并预处理集合 P 中的元素的相关信息。每次询问时找到与查询的元素[11] X 对应的元素 Y。因为元素 Y 的有关信息已经预处理过，并且询问满足条件 3.1，就能以 $O(f(L))$ 的复杂度[12] 将元素 Y 的信息快速地转移为元素 X 的信息。

3.2 线性序列的预处理

本小节考察一类在线性序列上频繁查询某连续子序列的信息并且满足预处理的条件的问题。首先介绍预处理的方法，然后给出一道例题。

我们首先将序列分块。设每个块的大小为 S，则块的数目是 $\Theta(N\ /\ S)$ 的。根据条件 3.1，对任意含有 len 个元素的区间，可在 $O(f(len))$ 的时间复杂度得到询问的相关信息。于是，我们这样预处理任意两块之间[13]的相关信息：

设块 A 为任意的块，考虑按在序列中位置从前到后的顺序依次计算块 A 与它之后的每个块 B_i 之间的信息。根据条件 3.1，计算出块 A 与块 B_i 之间的信息以后，可在 $O(f(S))$ 的复杂度内转化为块 A 与块 B_{i+1} 之间的信息。于是有

[11]此处“元素”指集合 T 的元素，实际上是数据集合 S 的一个子集。
[12]此处 $f(L)$ 指一个以 L 为自变量的函数
[13]从靠前的块的左端点到靠后的块的右端点的连续子序列

定理3.1. 仅考虑位置在块 A 之后的每个块 B_i。可以在 $O(f(N))$ 的复杂度内计算出块 A 与所有块 B_i 之间的信息，如果不考虑记录信息的复杂度。

若记录任意两块之间信息的复杂度可以不考虑的话[14]，我们枚举块 A_i，就能以 $O(Nf(N)/S)$ 的时间复杂度完成分块和预处理的过程。

根据性质 2.2，任意区间 $[L,\ R]$ 的信息可以由某段已预处理过的两块之间的子序列的信息通过添加不超过 $2S$ 个元素得到[15]。从而得到

定理3.2. 对于线性序列上区间统计类型的问题，若问题满足预处理的条件，首先对序列分块和预处理，以后在查询任意区间 $[L,\ R]$ 时，可以在 $\Theta(f(S))$ 的复杂度内得到区间 $[L,\ R]$ 的信息。

例题三[16]

给出一个 N 个数的序列。每个数是 $1 \sim N$ 之间的整数。你需要回答 M 次询问。每次询问给出三个整数 $L_i,\ R_i,\ K_i$，求区间 $[L_i,\ R_i]$ 中出现了恰好 K_i 次的数的数目。

数据规模: $1 \leq N,\ M \leq 10^5$。

算法分析

这道题目中，相邻的区间的答案不能通过记录一些附加信息而快速合并为它们组成的整个区间的答案，因此不能使用线段树等数据结构维护。但是询问满足这样的性质：

引理3.3. 对任意的两个区间 $[L_1,\ R_1]$ 和 $[L_2,\ R_2]$，如果已经记录在区间 $[L_1,\ R_1]$ 中 $1 \sim N$ 之间的每个数的出现次数以及出现次数为 $1 \sim N$ 的数的个数，可以在 $O(|L_1 - L_2| + |R_1 - R_2|)$ 的时间复杂度内转化为区间 $[L_2,\ R_2]$ 中 $1 \sim N$ 之间的每个数的出现次数以及出现次数为 $1 \sim N$ 的数的个数。

因而可以使用预处理的方法解决此问题。

[14]可以采用离线算法或别的手段。

[15]在得到区间 $[L,\ R]$ 的信息以后还要删除这些元素，恢复原来记录的信息。

[16]**题目来源：原创**

将序列分块，令每块的大小为 $\Theta(\sqrt{N})$，则块的数目为 $\Theta(\sqrt{N})$。我们能以 $\Theta(N\sqrt{N})$ 的时间复杂度得出任意两块之间 $1 \sim N$ 之间的每个数的出现次数以及出现次数为 $1 \sim N$ 的数的个数。但是这样记录的信息是 $\Theta(N^2)$ 的，必须想办法减少记录的信息量。

引理3.4. *将所有块从前到后编号为 $1 \sim \sqrt{N}$。则任意一个数 x 在第 i 块到第 j 块的出现次数等于 x 在前 j 块的出现次数减去 x 在前 $i-1$ 块的出现次数。*

我们只维护 $1 \sim N$ 之间的每个数在前 $1 \sim \sqrt{N}$ 块的出现次数，复杂度降为 $\Theta(N\sqrt{N})$。这样做与维护它们在任意两块之间的出现次数是等价的[17]。

引理3.5. *在序列中出现的次数不少于 $\sqrt{N}$ 的数至多有 $\sqrt{N}$ 个。*

在预处理时我们只记录两块之间出现次数在 $1 \sim \sqrt{N}$ 的数的数目。于是记录的复杂度是 $\Theta(N\sqrt{N})$。根据定理 3.2，我们能以 $O(\sqrt{N})$ 的复杂度正确地回答所有 $K_i \leq \sqrt{N}$ 的询问。

对于 $K_i > \sqrt{N}$ 的询问，我们提前处理所有出现次数大于 $\sqrt{N}$ 的数的出现次数的前缀和，询问时以 $O(\sqrt{N})$ 的复杂度枚举这些数，$O(1)$ 判断每个数在询问区间中的出现次数是否等于 K_i。

定理3.6. *上述算法可以在 $O(N\sqrt{N})$ 的时间复杂度内完成预处理。对于每次询问，可以在 $O(\sqrt{N})$ 的复杂度内得到答案。总的时间复杂度和空间复杂度为 $O((N+M)\sqrt{N})$。*

3.3 树形结构的预处理

本小节考察一类树形结构中频繁查询两点之间的路径的信息并且满足预处理的条件的问题。首先将线性序列中分块预处理的方法推广到树上，说明一种在树上标记关键点的方法，然后给出应用此方法的例子。

引理3.7. *树形结构中，任意两点之间的简单路径[18] 是唯一的。*

引理3.8. *对于有根树中的任意点 u 和点 v，设点 p 为它们的 LCA，则点 u 到点 v 的路径恰好由点 u 到点 p 的路径和点 p 到点 v 的路径两部分组成。*

[17]将预处理的第 i 块到第 j 块的每个数的出现次数转化为任意区间 $[L, R]$ 的每个数出现次数时，只需对前 j 块的每个数出现次数做修改，使用时与前 $i-1$ 块的每个数出现次数做差即可。

[18]指经过每个点至多一次的路径。以下“路径”均指简单路径。

我们希望将树中的若干个点标记为关键点，并预处理任意两个关键点之间的路径，使得树上的任意一条较长的路径 p 涵盖了一条已处理过的路径[19] ，并且路径 p 中除此以外的部分的含有的点数较少。下面介绍一种贪心的算法。

对一个有根树，按深度从大到小的顺序枚举树中的每个点并判断是否将其标记为关键点。将任意一个点 u 标记为关键点当且仅当点 u 的子树中存在一个点到点 u 的路径上没有关键点且路径上的点的数目大于 S。 S 是给定的数。

引理3.9. 设 N 为树的点集大小，则贪心算法标记的关键点不超过 N / S 个。

引理 3.9 成立的原因是每个关键点至少是 S 个非关键点到根的路径上的第一个关键点。

引理3.10. 设根结点的深度为 0，则任意一个深度不小于 S 的点的最近的 $S+1$ 个祖先结点（含该点自身）中至少有一个是关键点。

推论3.11. 对于树中的任意一条经过的点数大于 $3S$ 的简单路径，设点 u 和点 v 分别是路径的两个端点。则点 u 到路径上最近的关键点的距离[20]不超过 $2S-2$，点 u 与点 v 分别到路径上最近的关键点的距离之和不超过 $3S-3$。

在标记关键点的贪心算法中，我们只需对每个点记录它的子树内到它的由非关键点组成的路径经过的点数的最大值，就能判断是否要标记它。

定理3.12. 在 $\Theta(N)$ 的时间复杂度内可以将树中的 $\Theta(S)$ 个点标记为关键点。在 $\Theta(S^2 f(N))$ 的复杂度内进行预处理，那么任意一条简单路径中除去已处理过的部分外的点的数目为 $O(N / S)$。

例题四[21]

给出一个 N 个点的树。每个结点的颜色是 $1 \sim M$ 之间的一个整数。

定义从树上任意点 u 到点 v 的旅行收益为 $\sum_{i=1}^{M} \sum_{j=1}^{C_i} V_i W_j$。其中 i 是每种颜色， V_i 是颜色 i 的权值，C_i 是颜色 i 在点 u 到点 v 的简单路径上出现的次数，而$W_1, W_2, \ldots, W_N$是给出的整数。

[19]此条件也可以是任意一条路径与一条已经处理过的路径的对称差较小，但要判断的情况很多，所以程序实现会复杂很多。

[20]指经过的边的数目

[21]**题目来源：**NOI Winter Camp 2013, **糖果公园**

你需要执行 Q 次操作。操作有两种类型：
– $CHANGE\ x_i\ y_i$　　将点 x_i 的颜色修改为 y_i。
– $QUERY\ u_i\ v_i$　　查询从点 u_i 到点 v_i 的旅行收益。

数据规模: $1 \leq N,\ M,\ Q \leq 10^5,\ 1 \leq V_i,\ W_i \leq 10^6$.
本题的时限为 10 秒。

算法分析

将任意一个点作为根。令 $S = \Theta(N^{\frac{2}{3}})$，使用标记关键点的方法对树进行预处理，记录任意两个关键点之间的路径中颜色 $1 \sim M$ 的出现次数和旅行收益。预处理的复杂度是 $\Theta(N^{\frac{5}{3}})$。

对于查询点 u_i 到点 v_i 的路径的操作，先用倍增等高效的算法找到两个点的 LCA。接着，分别从点 u_i 和点 v_i 出发沿着路径移动，找到路径上最接近点 u_i 和点 v_i 的关键点。然后就能利用已经记录的信息，经过若干次增量操作在 $O(N^{\frac{2}{3}})$ 的复杂度内得到答案。

对于修改点 x_i 颜色的操作，枚举任意两个关键点，判断点 x_i 是否在这两个点之间的路径上，如果是的话就要更新记录的信息。

引理3.13. *点 x_i 在点 u 到点 v 的路径上当且仅当点 u 和点 v 的 LCA 是点 x_i 的祖先并且点 x_i 是点 u 或点 v 的祖先。*

如果预处理每个点在 DFS 中进栈和出栈的时刻，那么点 x 是点 y 的祖先当且仅当点 y 进栈的时刻在点 x 的进栈和出栈的时刻之间。我们在预处理时记录任意两个关键点的 LCA，就可以 $O(1)$ 判断点 x_i 是否在它们之间的路径上。因此修改操作的复杂度是 $\Theta(N^{\frac{2}{3}})$。

定理3.14. *可以在 $O((N+Q)N^{\frac{2}{3}})$ 的复杂度内解决此问题。*

4　分块与离线算法

有些数据处理问题并不要求对每次询问即时得到答案，而是一次给出大量

的询问，要求程序尽快计算出这些询问的结果并统一反馈。离线算法就是指预先知道所有询问的内容因而一起解决所有询问的算法。

在离线问题中，因为已经得知所有询问，我们可以把询问按一定的方式组织起来，从而充分利用询问之间的重叠信息。实际上是将所有询问作为数据集合的一部分处理，于是可以用原来处理数据集合的方法处理询问。

本节将用两个例子说明分块思想在设计离线算法时的应用。

例题五[22]

给出一个 N 个数的序列。每个数是 $1 \sim N$ 之间的整数。你需要回答 M 次询问。每次询问给出三个整数 $L_i,\ R_i,\ K_i$，求区间 $[L_i,\ R_i]$ 中出现了恰好 K_i 次的数的数目。

数据规模: $1 \leq N,\ M \leq 10^5$。

题目分析

将序列分块，每个块的大小设为 $\Theta(\sqrt{N})$。块的数目也为 $\Theta(\sqrt{N})$。

一次读入所有的询问，并把询问按查询区间的左端点 L_i 所在的块分类。然后分别处理每一类询问。

考虑左端点在块 A_i 的所有询问。我们把这些询问按右端点在序列中的位置从小到大排序。然后按顺序扫描从块 A_i 的左端点到序列末尾的每个元素，用数组维护每个时刻已扫描过的所有元素中出现次数为 $1 \sim N$ 的数的数目和 $1 \sim N$ 之间的每个数的出现次数。

对于每个询问 $[L_i,\ R_i]$，当我们扫描到询问的右端点 R_i 时可以在 $O(\sqrt{N})$ 的复杂度内将当前维护的信息转化为区间 $[L_i,\ R_i]$ 的信息。在记录询问的答案以后再将信息恢复。每个询问都会被处理一次。

定理4.1. *此算法的时间复杂度为 $O((N+M)\sqrt{N})$，空间复杂度为 $\Theta(N)$。*

在这个例子中，我们采用离线的办法解决了例题三中需要记录的信息量过大的问题。可以看到，分块与离线算法的结合和预处理的方法有一定相似之处。

[22]题目同例题三

而且能使用分块与预处理的方法解决的问题一般可以在离线情况下降低空间复杂度。

下面再看一个树形结构的例子。

例题六[23]

一个 N 个点的树。每个点有权值，权值是小于 2^{31} 的非负整数。你要回答 M 次询问，每次查询点 u_i 到点 v_i 的简单路径中出现的不同点权的数目。

数据范围： $1 \le N \le 40\,000,\ 1 \le M \le 100\,000.$

题目分析

首先将所有点的权值离散化[24]，之后可以认为权值是不超过 N 的整数。使用 3.3 节介绍的树上标记关键点的方法，令 $S = \Theta(\sqrt{N})$，则关键点的数目也为 $\Theta(\sqrt{N})$。根据推论 3.11，每个点与最近的关键点的距离为 $O(\sqrt{N})$。

读入所有的询问并把每个询问按到对应的路径端点 u_i 距离最近的关键点分类。分别处理每一类询问。

考虑处理以关键点 r_i 为与 u_i 距离最近的关键点的一类询问。

以点 r_i 为根深度优先遍历整棵树，同时维护当前已入栈而尚未出栈的所有点中每个权值的出现次数以及不同权值的数目。

引理4.2. *访问任意点 x_i 时，已入栈而尚未出栈的所有点恰好是从点 x_i 到根的路径上的所有点。*

引理4.3. *任意点 i 到任意点 j 的路径等同于点 i 到根的路径加上点 j 到根的路径再减去点 i 与点 j 的 LCA 到根的路径。*

对于每个询问 $(u_i,\ v_i)$，遍历到点 v_i 时维护的栈内的点的信息就是点 v_i 到根的路径上的所有点的信息。此时枚举从点 u_i 到根 r_i 的路径上的每个点 x_i。如果点 x_i 是点 u_i 与点 v_i 的 LCA 则不作改动，否则如果点 x_i 是 LCA 到根 r_i 的路径上的点则减去点 x_i 的权值，否则添加点 x_i 的权值。

[23]**题目来源：** Sphere Online Judge 10707, Count On A Tree II

[24]即建立 N 个点的权值的集合到不超过 N 的自然数的集合的单射。一般使用 $O(N \log N)$ 的排序方法即可。

点 u_i 与根 r_i 的距离是 $O(\sqrt{N})$ 的，因此有

引理4.4. 可以在 $O(\sqrt{N})$ 的时间复杂度内得到一次询问的答案。

定理4.5. 此问题可以在 $O((N+M)\sqrt{N})$ 的时间复杂度和 $O(N)$ 的空间复杂度内解决。

5 总结

形象的说，分块思想的实质是将复杂度为 $O(f(N))$ 的朴素算法转化为复杂度为 $O(g(S)+h(N/S))$ 的算法。其中 S 是块的规模，N / S 是块的数目。只是对于不同的问题，有不同的体现。

以例题一为例，这道题目的参考算法中两处体现了分块思想。一处是对于 $D_i < \sqrt{N}$ 的修改和 $D_i \geq \sqrt{N}$ 的修改分别用两种数据结构维护。另一处是在修改操作很频繁的情况下用分块的结构维护序列的段元素和。在这个特殊问题上，此方法比线段树更加高效，因为修改的数目达到查询的数目的 $O(\sqrt{N})$ 倍。后面的例题也是类似的。不过有时我们将一个问题分割为若干个本质相同的子问题，有时则将一个问题分离为若干个差异较大的子问题。

本文给出了线性序列和树形结构上分块的基本模式，对一类范围很广的问题都是有效的。但是究竟应该如何应用分块思想是因题而异的，同学们不要拘泥于这里给出的几种形式，而要勇于创新。当然，如果存在比分块的方法更高效的算法，也要学会选择，不能墨守成规。希望这些例子能对读者有所启发。

6 参考文献

1. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein. “Introduction to Algorithms (2nd ed.)”, 潘金贵等译，机械工业出版社。

2. Jon Kleinberg, Eva Tardos. “Algorithm Design”, 清华大学出版社。

3. 吴文虎, 王建德.《世界大学生程序设计竞赛(ACM/ICPC)高级教程》(第一册), 中国铁道出版社。

4. 郑暾, 《平衡规划——浅析一类平衡思想的应用》。