第4章 莫队算法

一、简介

有这样一类问题：给出序列，需要完成序列的区间询问（先默认没有修改）。对于静态序列，求解区间和、区间最值问题可以使用线段树等数据结构。然而使用线段树等数据结构的前提条件是：区间信息能够由子区间信息合并得到，例如区间和，区间最值等问题。对于不满足上述性质的问题，我们可采用莫队算法求解。

莫队算法的核心思想是：用左右指针S_l、S_r表示当前维护的区间是$[S_l, S_r]$。若询问区间$[L, R]$的答案，则将S_l移动至L，将S_r移动至R，并在移动的过程中实时维护区间$[S_l, S_r]$的答案。并且为了保证时间复杂度，还需要提前将询问区间按照某种规则排序后回答。

莫队算法可以解决一类离线的区间询问问题，适用性极为广泛。同时将其加以扩展，便能轻松处理树上路径询问，也能支持修改操作。

二、普通莫队

【前提条件】

使用莫队算法时，题目需要满足下列条件：

（1）题目允许离线

（2）题目没有修改操作（后面会进行拓展，使得莫队能支持一些简单的修改操作）

（3）若已知区间$[L_1, R_1]$的答案，我们可以**花费$O(|L_1 - L_2|)$的时间**将区间左指针从L_1移动至L_2，**花费$O(|R_1 - R_2|)$的时间**将区间右指针从R_1移动至R_2，最终得到区间$[L_2, R_2]$的答案。注意：这里我们假设指针移动一步的复杂度是$O(1)$的。

二、普通莫队

【初步想法】

莫队算法的核心是将所有的询问重新排序，按照这个新的顺序依次回答询问，并且回答每一个询问时会以上一个询问为基础。

我们有这样的想法：将询问$[L_i, R_i]$看成是平面上的一个坐标(L_i, R_i)，并定义两个询问间转移的花费为它们在二维平面上的曼哈顿距离。想象一下平面上有若干个点，你以不同的顺序走完这些点，走过的总长度肯定不一样。

若将询问看做平面上的点，回答询问也有类似的性质：以一个较优的顺序回答所有询问会花费较少的时间。显然最短哈密顿路径（经过图中所有点恰好一次的最短路径）是最优的方案。然而求最短哈密顿路径是个NP问题，我们需要找到合适的替代方案。

莫队算法的核心思想就在于此：合理安排回答询问的顺序，以达到优化时间复杂度的目的。

二、普通莫队

【莫队的排序方法】

对于长度为n的序列，先将序列平均分成$\sqrt{n}$块，然后把m个询问按照左端点所在块的编号为第一关键字、右端点所在位置为第二关键字排序。

以该顺序处理询问，可以证明时间复杂度为$O((n+m)\sqrt{n})$。

【时间复杂度证明】

统计区间的左右端点各移动了多少次：

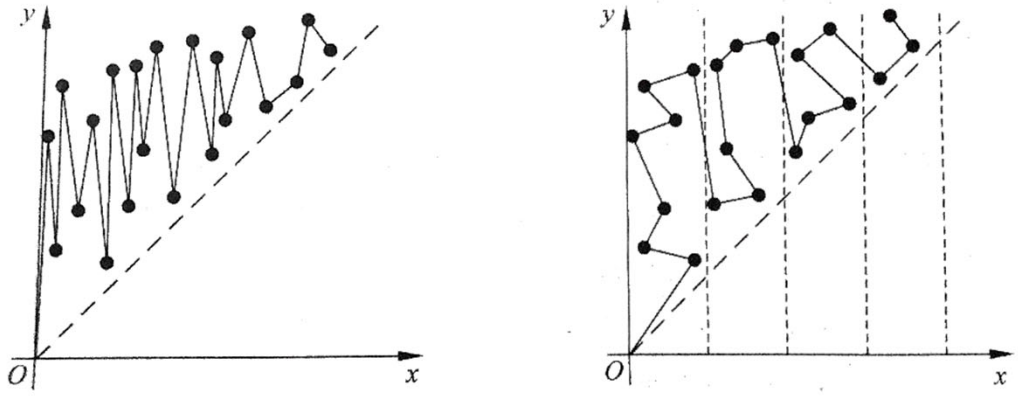
右端点：由于排序的第二关键字为右端点所在的位置，若左端点所在块不变，则右端点下标是不降的，右端点最多移动$O(n)$步（从1移动到n）；当左端点所在块变化时，右端点最多移动$O(n)$步。因为只有$\sqrt{n}$个块，所以左端点所在块最多变化$\sqrt{n}$次，右端点最多移动$O(n\sqrt{n})$步。时间复杂度为$O(n\sqrt{n})$。

左端点：排序的第一关键字是左端点所在块编号，因此按顺序处理询问的过程中，左端点所在块是不降的。由于左端点所在块改变所产生的左端点移动的总时间复杂度为$O(n)$；当左端点所在块不变时，每次询问使得左端点最多移动$\sqrt{n}$步。若有m个询问，则左端点移动的总时间复杂度为$O(n+m\sqrt{n})$。

所以整个算法的时间复杂度是$O\left((n+m)\sqrt{n}\right)$的。

二、普通莫队

从几何角度解释，如果将暴力法是以左端点 L 排序再按顺序处理询问的话，那么莫队的排序算法，就是将同一块内询问的 x 方向振幅限定在 $O(\sqrt{n})$ 的范围，从而缩短路径长度，提高效率。



(a) 暴力法　　(b) 莫队算法

从这个证明也可以看出，如果希望在常数层面进一步优化运行时间，可以考虑**奇偶性排序**的优化方法。即对于左端点L在奇数块的询问，将R递增排序；对于左端点L在偶数块的询问，将R递减排序。这样能够在L所在块变化时，尽量减少R的移动距离。

二、普通莫队

另外，如果 n,m 的数量级相差较大，需要通过调整块大小得到尽量优的时间复杂度。

设块大小为 S，那么根据上面的分析，左右端点的移动总次数是 $O\left(\frac{n}{S}\cdot n+mS\right)=O\left(\frac{n^2}{S}+mS\right)\leq O(n\sqrt{m})$，当 $S=\frac{n}{\sqrt{m}}$ 时取等。也就是说，如果块大小取到与 $\frac{n}{\sqrt{m}}$ 同阶，就可以做到最优的时间复杂度 $O(n\sqrt{m})$。

【移动端点的注意点】

在移动当前维护区间 $[l,r]$ 的端点时，如果某时刻出现 $l>r+1$ 的情况，那么会存在一个元素，它的加入次数是负数，这在某些题目会出现问题。特别地，如果 $l=r+1$，我们看做是“空区间”，这种情况规定它是合法的。

所以为了避免这种情况，一种比较推荐的写法是：前两步先扩大区间（`l--` 和 `r++`），后两步再缩小区间（`l++` 和 `r--`）。这样能够保证维护过程中任何时候都有 $l\leq r+1$。

二、普通莫队

【例题1】区间不同数

【题目大意】

给定n个数的序列，接下来m次询问，每次询问区间$[l,r]$内有多少个不同的数。

【算法分析】

考虑使用莫队算法，把m个询问按照左端点所在块的编号为第一关键字、右端点所在位置为第二关键字排序。

我们用两个指针s_l,s_r表示当前在处理询问区间$[s_l,s_r]$，同时用cnt_i统计数字i的出现次数。考虑两个指针移动时答案的改变。

如果s_l向前或s_r向后移动，则区间扩大，一个数字进入当前区间，若这个数在之前没出现过，则答案加一；如果s_l向后或s_r向前移动，则区间缩小，一个数字被移出当前区间，若这个数曾出现一次，则新区间中不存在这个数，答案减一。

指针的移动是$O(1)$的，总时间复杂度$O((n+m)\sqrt{n})$。

【回滚莫队】

有些题目在区间转移时，**可能会出现增加或者删除无法实现的问题**。在只有增加不可实现或者只有删除不可实现的时候，就可以使用回滚莫队在 $O(n\sqrt{m})$ 的时间内解决问题。回滚莫队的核心思想就是既然只能实现一个操作，那么就只使用一个操作，剩下的交给回滚解决。

以只使用增加操作的回滚莫队为例，同样对原序列进行分块，对询问按以左端点所属块编号升序为第一关键字，右端点升序为第二关键字的方式排序。

按顺序处理询问：如果询问左端点所属块 B 和上一个询问左端点所属块的不同，那么将莫队区间的**左端点初始化为 B 的右端点加 1**，将莫队区间的**右端点初始化为 B 的右端点**；

如果询问的左右端点所属的块相同，那么直接扫描区间回答询问；

如果询问的左右端点所属的块不同：

- 不断扩展右端点直至莫队区间的右端点等于询问的右端点；
- 不断扩展莫队区间的左端点直至莫队区间的左端点等于询问的左端点；
- 回答询问；
- **撤销**莫队区间左端点的改动，使莫队区间的左端点回滚到 B 的右端点加 1。

这样，在移动端点的过程中，就不需要进行“删除”，而只需要处理“插入”。这个技巧也可以应用到后面的带修莫队。

三、带修改的莫队

一般情况下莫队算法不支持修改，但如果修改操作比较简单，我们仍可以使用莫队算法解决。比如在上题中，我们加入修改操作：每次修改某个位置上的值，那么我们可以通过拓展让莫队算法支持这样的单点修改。

【基本思路】

先前没有修改操作，询问的先后次序无所谓；现在有了修改操作，当前询问会受先前修改的影响。于是我们将时间戳（第i个操作的时间戳为i）作为询问的**第三个属性**，把每个询问记为 (L_i, R_i, t_i)。规定出现时间在询问之前的修改才能产生效果。

将询问按某一原则排序，考虑两个询问间的转移。设上一个询问的时间戳为t_1，下一个询问的时间戳为t_2，那么我们**要处理t_1~t_2时间段的修改操作**（可能是产生影响，也可能是撤销影响）。若被修改的元素位于当前询问区间中，则它会对答案产生影响。

将询问看作**三维空间中的点**，前两维是询问区间的左右两端点下标（这与之前是一致的），第三维是询问操作的时间戳，我们发现两个询问间转移的花费等于它们在三维空间上的曼哈顿距离。我们依然可以像之前那样分块解决问题。

三、带修改的莫队

【莫队思想】

考虑分块，设置块的大小为 B（通常取$B = n^{\frac{2}{3}}$），则共有$n^{\frac{1}{3}}$个块。将询问按左端点所在块为第一关键字、右端点所在块为第二关键字、时间戳为第三关键字进行排序。

回答询问时需要多添加一个时间指针，左右指针的移动方式不变。时间指针的移动与普通指针类似，只要每次在左右指针不变的前提下，将当前的时间指针 t 移动到 $t-1$ 或 $t+1$ 即可。但修改操作的处理方式因题而异，时间的倒流与顺流也会有所区别。

三、带修改的莫队

【时间复杂度分析】

假设序列长度为 n，操作总数为 m，块大小取 B。我们分析一下时间指针和左右端点移动的时间复杂度。

时间指针：当左右端点所在块不变时，询问是按时间排序的，因此询问的时间戳不降，时间指针最多向后移动$O(m)$步；当左右两端点所在块改变时，时间指针最坏情况下移动$O(m)$步。而由于左右两端点所在块的组合方案共有$\left(\frac{n}{B}\right)^2$种，每种组合中时间指针只会向后移动$O(m)$步；并且左右端点所在块改变也只有$\left(\frac{n}{B}\right)^2$次，每次改变使得时间指针最坏情况下移动$O(m)$步。所以时间指针移动的总复杂度为$O\left(\frac{mn^2}{B^2}\right)$。

三、带修改的莫队

右端点：当左右端点所在块不变时，每次询问右端点最多移动$O(B)$步，由于有m次询问，所以这里的复杂度不超过$O(mB)$；当左端点所在块不变、右端点所在块改变时，由于右端点所在块改变所带来的移动不超过$O(n)$，左端点所在块有$\frac{n}{B}$种取值，所以这里的复杂度不超过$O\left(\frac{n^2}{B}\right)$；当左端点所在块改变时，右端点最多移动$O(n)$步，与之前类似，左端点所在块有$\frac{n}{B}$种取值，所以这里的复杂度不超过$O\left(\frac{n^2}{B}\right)$。右端点移动的总时间复杂度为$O\left(mB+\frac{n^2}{B}\right)$。

左端点：左端点所在块不变时，每次询问左端点最多移动$O(B)$步，由于有m次询问，所以这里的复杂度不超过$O(mB)$。当左端点所在块改变时，因为排序是按左端点所在块为第一关键字的（左端点所在块编号不降），所以由左端点所在块改变所带来的移动总共不超过$O(n)$。左端点移动的总时间复杂度也为$O(mB+n)$。

综上，总时间复杂度为$O\left(\frac{mn^2}{B^2}+mB+\frac{n^2}{B}+n\right)$，通常忽略 $\frac{n^2}{B}+n$（即通常忽略左右端点在不同块之间移动的复杂度）。于是将时间复杂度写成 $O\left(\frac{mn^2}{B^2}+mB\right)$。取 $B=n^{\frac{2}{3}}$ 时有最优时间复杂度 $O\left(mn^{\frac{2}{3}}\right)$。假设询问数和序列长度是同阶的，也就是说询问数是$O(n)$的，那么总时间复杂度为 $O\left(n^{\frac{5}{3}}\right)$。

三、带修改的莫队

【例题2】数颜色

【题目大意】

给定n个数的序列，m次操作，操作有两种：

1.询问给定区间内有多少个不同的数

2.修改序列中某一位置的数

$n, m \leq 10^4$。

【算法分析】

如果没有修改操作，我们能用普通莫队轻松解决该题。但是题目有单点修改，所以用带修改的莫队。

考虑两个询问间的转移。设上一个询问的左右端点分别是L_1, R_1，下一个询问的左右端点分别是L_2, R_2。那么我们将左指针从L_1移动至L_2，将右指针从R_1移动至R_2。

加入一个数字时，若这个数在之前没出现过，则答案加一。

删除一个数字时，若这个数曾出现一次，则新区间中不存在这个数，答案减一。

三、带修改的莫队

考虑修改。设上一个询问的时间戳为t_1，下一个询问的时间戳为t_2，那么我们要处理t_1~t_2时间段的修改操作。枚举t_1~t_2时间段修改操作的过程相当于将时间指针从t_1移动至t_2的过程。

假设现在要将pos位置的元素由a改成b。若pos在区间$[L_2, R_2]$内，则加入数字b，删除数字a，并计算“加入删除”操作对答案的影响。

这样这道题就可以用带修改莫队轻松解决。假设n, m同阶，那么总时间复杂度为$O(n^{\frac{5}{3}})$

四、树上莫队

有时需要处理的问题并不是序列上的区间问题，而是树上的路径问题，例如询问树上路径信息。

事实上，我们也可以把莫队算法拓展到树上，即树上莫队。对于一棵有根树，我们将树边(x, fa[x])的信息记在x点上，那么路径信息就储存在节点上。

树上莫队的核心思想仍然是分块，主要有两种分块方法：按照dfs生成的序列分块、按照树的结构分块。

四、树上莫队

【方法1.1】

我们将树上节点按特定原则排列，将树上问题转化为序列问题。我们希望将树上的路径问题转化为序列上的区间问题，这样就能用普通的莫队算法解决。

若将莫队搬到树上，需要解决两个问题：**第一，如何通过合理的分块来保证两个指针的移动总次数的上界；第二，如何在树上移动指针。**一种比较方便的方法是，记录所有结点DFS的入栈出栈序（括号序）。即维护一个序列，当我们访问到一个结点（即入栈）时将其加入序列，退出时也将其加入序列，这个序列的长度为 $2n$。记 $st[x]$ 表示结点 x 的入栈序，$ed[x]$ 表示结点 x 的出栈序。

我们发现，一个路径 (u,v) 可以被表示成序列上的一个区间，这个区间的表示方法在不同情况下有两种：

- 若 $\boldsymbol{u,v}$ **是祖先和后代关系**，则区间可以被表示为 $[\boldsymbol{st[u],st[v]}]$ **或** $[\boldsymbol{st[v],st[u]}]$。这个区间内的点中，路径 (u,v) 上的点都出现且仅出现一次，出现两次或没有出现的不属于路径。
- 若 $\boldsymbol{u,v}$ **不是祖先和后代关系**，则区间可以被表示为 $[\boldsymbol{ed[u],st[v]}]$ **或** $[\boldsymbol{ed[v],st[u]}]$。这个区间内的点中，路径 (u,v) 上的点除去 $\boldsymbol{lca(u,v)}$ 都出现且仅出现一次，出现两次或没有出现的不属于路径或是 $lca(u,v)$。

这样我们仍然可以沿用序列的方法，只要将序列分块，维护两个指针然后移动即可。如果没有修改，取 $S=\sqrt{2n}$ 可以得到最优复杂度。

四、树上莫队

【方法1.2】

使用**括号序**分块（与方法**1.1**类似），依据普通莫队的原则将所有询问排序。具体来说，就是以路径 (u,v) 的第一个点 $st[u]$ 所在块为第一关键字，以第二个点的 $st[v]$ 为第二关键字排序。

假设上一个询问是(u_1,v_1)，当前询问(u_2,v_2)。

在方法**1.1**中，两个指针分别指向序列中区间的两个端点，通过指针在序列上的移动来实现从(u_1,v_1)到(u_2,v_2)的转移。改进以上做法：**将指针指向树中的节点**。将指针从u_1移动至u_2的过程中把树上u_1到u_2**路径上的标记取反**，将指针从v_1移动至v_2的过程中把v_1到v_2路径上的标记取反。通过画图或者简单推导，可以发现：树上打有标记的点（边）构成了路径(u_2,v_2)。

但注意到，给边打标记用这种方法不会有问题，但给点打标记**可能会在某些LCA重复标记**。所以不妨将“标记点”转化成“标记边”，即指针从 u_1 移动到 u_2 的过程中，我们只对 $LCA(u_1,u_2)$ 之外的点改变标记。最后**在查询路径 (u,v) 时再修改 $LCA(u,v)$ 的标记**。

因为任意两点 u,v 的距离，必然不超过它们括号序之差绝对值，所以方法**1.2**的端点移动次数比方法**1.1**来得更少，时间复杂度也是正确的。

四、树上莫队

【方法2】

对树分块，设块的大小为B。依然对树进行DFS，假如当前正在遍历x的子树，若x的子树节点中至少有B个在队列里，那么将队列中x的子树节点分为一块，然后从队列中清除这些点。访问完x的子树后将x放进队列。最后如果剩下没有分块的点，必然是包含根的一个连通块，就单独分成一块即可。

完成分块之后我们按莫队的原则将所有询问排序（路径第一个点所在块编号为第一关键字，路径第二个点 dfs 序为第二关键字排序），询问间的转移与方法1.2完全相同。因为块内任意两点间距离是O(B)的，所以指针“在树中的块内移动”和“在序列上的块内移动”具有相同的性质，时间复杂度得以保证。

可以发现，树上莫队和序列莫队算法大致相同，时间复杂度也是相同的。

比较分析：在实际操作中，虽然方法1.2和方法2在转移时跳的点数比方法1.1少，但是转移时判断的条件比较多，效率不一定更好。将树上路径直接当成序列区间处理的方法1.1，实际表现有时优于其余两种方法。一般来说，只要熟练掌握一种方法即可。

四、树上莫队

【例题3】糖果公园

【题目大意】

给定一棵n个点的树，每个点有一种颜色，共有m种不同的颜色。q组操作，每次可以修改一个点的颜色，或是询问路径(s_i,t_i)的权值。

其中路径的权值定义如下：令$Tim[i]$表示颜色i的出现次数，val[i]表示颜色i的价值，w[j]是题目给出的常数，则该路径的权值是$\sum_{i=1}^{m}(val[i]*\sum_{j=1}^{tim[i]}w[j])$。

四、树上莫队

【算法分析】

我们考虑用带修改的树上莫队解决本题。假设第i个操作是询问操作，那么该询问可以写成(s_i,t_i,i)的形式。排序方法如下（参考代码采用的是方法2）：

- 如果采用方法1.1转化成序列问题，就和序列的带修莫队一样处理。
- 如果采用方法1.2分块（括号序分块），就以 $st[s_i]$ 所在块编号为第一关键字、$st[t_i]$ 所在块编号为第二关键字、时间戳为第三关键字，将m个询问排序。
- 如果采用方法2分块（树上分块），就以s_i所在块编号为第一关键字、t_i所在块编号为第二关键字、时间戳为第三关键字，将m个询问排序。

接下来以方法1.2和方法2为例说明解法。

假设上一个询问是(u_1,v_1,t_1)，当前询问(u_2,v_2,t_2)。$col[x]$表示在t_1时刻节点x的颜色。显然路径(u_2,v_2)的答案只和路径(u_2,v_2)上每种颜色的出现次数有关。

在指针从u_1移动至u_2的过程中将沿路的节点状态取反。具体的，假设指针移动到节点x，若点x未计入答案，则将点x计入答案$(Tim[col[x]]+=1)$；若点x已计入答案，则取消点x对答案的影响$(Tim[col[x]]-=1)$。同理将路径(v_1,v_2)上的节点状态取反。到此为止，我们得出了t_1时刻路径(u_2,v_2)上每种颜色的出现次数，顺带统计出了t_1时刻路径(u_2,v_2)上的答案。

四、树上莫队

【算法分析】

考虑t_1~t_2时间段的修改操作对答案的影响。枚举t_1~t_2时间段的修改操作，修改操作形如：将节点x的颜色由a改为b，我们将$col[x]$改为b，**若点x在路径(u_2, v_2)上**，则修改$Tim[a]$和$Tim[b]$，顺带维护答案。到此为止，我们得出了t_2时刻路径(u_2, v_2)上每种颜色的出现次数，并且得到了询问(u_2, v_2, t_2)的答案。

假设n与q同阶，令$B = n^{2/3}$ （参考程序取 $n^{0.6}$） ，则时间复杂度为$O(n^{5/3})$。