理解集合查询

蒋炎岩

南京大学计算机科学与技术系

课前预习

- 预习题 1：RMQ。给数列 $a_1, a_2, \ldots, a_n$ 和若干个形如 l, r 的询问，对每个询问求 $\min\limits_{l \le i \le r} a_i$
  - 理解 RMQ 问题的 $O(n \log n) - O(1)$ 的 sparse-table 算法，对每一个 i 和 k，预处理了区间 $[i, i+2^k)$ 的最小值，这样就可以在 $O(1)$ 的时间内求出询问的答案
- 预习题 2：GSS1。给数列 $a_1, a_2, \ldots, a_n$ 和若干个形如 l, r 的询问，对每个询问求 $f(l, r) = \max\limits_{l \le i \le j \le r} \left(\sum_{i \le k \le j} a_k\right)$
  - 请理解如何用线段树求解这个问题。解决问题的关键是如何用 $f(i, j)$ 和 $f(j+1, k)$ 推算出 $f(i, k)$ 的值，为此你可能需要一些辅助函数

集合查询

- 给定集合 A 和若干个询问，每个询问的形式都是
  - 对于 A 的子集 $A' \subseteq A$，$f(A')$ 的值是多少？
  - f 是一个预先给定的函数
- 例如给定 $A = \{a_1, a_2, \ldots, a_n\}$
  - 询问 $f(i,j) = \min_{i \le k \le j} a_k$
  - 询问 $f(i,j) = \sum_{i \le k \le j} a_k = a_i + a_{i+1} + \ldots + a_j$
- 有时候可能会支持修改操作

集合查询解析

- 给定集合 A 和若干个询问，每个询问的形式都是
  - 对于 A 的子集 $A' \subseteq A$，$f(A')$ 的值是多少？
  - f 是一个预先给定的函数
- 要素 1：A' 的结构
  - 最常见的两种类型：$A' = \{a_k \mid i \leq k \leq j\}$ (区间查询)，以及 $A' = \{a_k \mid i \leq a_k \leq j\}$ (数值查询)
- 要素 2：f 的性质
  - 常见的性质：交换律、结合率、能够求逆……

Range Minimum Query (RMQ)

- $A=\{a_1,a_2,\ldots,a_n\}$，询问 $f(i,j)=\min\limits_{i\le k\le j} a_k$
- 关于 $\min$ 的性质
  - $\forall X,Y\subseteq A$, $\min\{X\cup Y\}=\min\{\min\{X\},\min\{Y\}\}$
  - 这一性质比结合率更强：在 $X\cap Y\neq\varnothing$ 依然成立
  - 这意味着如果知道 $\min\{a_2,a_3,a_4,a_5\}$ 和 $\min\{a_4,a_5,a_6,a_7\}$，就可以算出 $\min\{a_2,a_3,a_4,\ldots,a_7\}$
  - Sparse Table $O(n\log n)-O(1)$ 算法[1]

[1]技巧 1：允许重叠的分治

Range Minimum Query (RMQ)

- $A = \{a_1, a_2, \ldots, a_n\}$，询问 $f(i,j) = \min_{i \le k \le j} a_k$
- 直觉：存储 $O(n \log n)$ 数量的信息似乎是不必要的？
  - 将连续 k 个数字组合[2]：$O((n/k)\log(n/k)) - O(k)$
  - 令 $k = \log n$，得到 $O(n) - O(\log n)$ 算法
  - 问题：时间主要花在求解 $\le k$ 的 RMQ 问题上
  - 小区间查询：$O((n/k)k\log k) - O(1)$
  - $k = \log n$ 时是一个 $O(n \log\log n) - O(1)$ 的算法
  - 核心性质：$\min\{X \cup Y\} = \min\{\min\{X\}, \min\{Y\}\}$，将大查询拆成可重叠的大小查询
  - 课后拓展：$O(n) - O(1)$，RMQ $\rightarrow$ LCA $\rightarrow$ RMQ$\pm$1

[2]技巧 2：组合

求和查询

- $A=\{a_1,a_2,\ldots,a_n\}$，询问 $f(i,j)=\sum_{i\le k\le j} a_k$
  - 不同于 min，$a+b+c\neq(a+b)+(b+c)$
- 利用加法逆元
  - $c+d=(a+b+c+d)+((-a)+(-b))$
  - 令 $S_i=a_1+a_2+\ldots a_i,\ S_0=0,\ f(i,j)=S_j-S_{i-1}$[3]
- 集合上的分治
  - $\forall i\le k\le j,\ f(i,j)=f(i,k)+f(k+1,j)$
  - 将 $f(i,j)$ 表示成 $f(i,j_1)+f(j_1+1,j_2)+\ldots+f(j_k+1,j)$
  - 预先求好若干个 $f(i,j)$，使得
    - 对任意查询，分解为 $O(\log n)$ 个子查询[4]
    - 如需支持修改，每个 i 只能被 $O(\log n)$ 个 $f(i,j)$ 包含[5]
    - 一种平衡的手段：将 $O(\sqrt{n})$ 个元素组合成一个[6]

---

[3] 技巧 3：部分和性质，树状数组

[4] 技巧 1：允许重叠的分治

[5] 技巧 4：线段树

[6] 技巧 2：组合

例子

- 假设 $A=\{a_1,a_2,\ldots,a_n\}$，利用分治的技巧回答以下查询。注意函数 f 具有的性质，以及如何从 $f(A)$ 和 $f(B)$ 推出 $f(A\cup B)$：
  - (rank) 询问 $f(i,j,x)=|\{a_k \mid i\le k\le j \text{ 且 } a_k\le x\}|$
  - (min-gap) 询问 $f(x,y)=\min\limits_{i,j\in A}\left\{|i-j| \,\middle|\, x\le i,j\le y\right\}$
- A 还可以是其他内容
  - (connectivity) A 是一维格点图，询问 $f(i,j)$ 为 i 和 j 的连通情况
  - (connectivity2) A 是 $2\times n$ 格点图，询问 $f(i,j)$ 为 i 和 j 的连通情况
  - (bracket) A 是括号序列，询问 $f(i,j)$ 是否正确配对

小结

- 集合查询的核心是分治
  - Sparse-table：预处理 $O(n\log n)$ 个集合，将查询分成 2 个相交的小集合
  - 线段树：预处理 $O(n)$ 个集合，将查询分成 $O(\log n)$ 个不相交的小集合
  - 块状链表：预处理 $O(\sqrt{n})$ 个集合，将查询分成 $O(\sqrt{n})$ 个不相交的集合，和 $O(\sqrt{n})$ 个元素

实现线段树

- 如何实现一个 $0, 1, \ldots, 2^n - 1$ 的线段树？
  - 第一级区间：$[0, 2^n)$
  - 第二级区间：$[0, 2^{n-1})$ 和 $[2^{n-1}, 2^n)$……
  - 对于 l，l 的二进制表示末尾有 k 个 0，则需要预处理一个 $[l, l+2^k)$ 的区间
- 问题 1：如何存储？
  - 每个区间的信息存储在数组中，$[l, r)$ 存储在下标 $l+r$，只需 $2n$ 个内存单元
- 问题 2：如何分解？
  - 对于 $[l, r)$ 区间的查询，找到一个从 l 开始的，不超过 r 的最大区间

非递归建树：从小区间开始

```
void build() {
  // 处理只有一个元素的 [l, l+1)
  for (int s = 2; s <= n; s *= 2) {
    for (int l = 0; l < n; l += s) {
      int r = l + s;
      // 用 [l, l+s/2) 和 [l+s/2, r) 更新 [l, r)
    }
  }
}
```

非递归查询：区间分解

```
int query(int l, int r) {
  int s = 0;
  while (l != r) {
    int s = min(1<<log2[r - l], lowbit[l]);
    // 分解出的一个区间是 [l, l+s)
    l += s;
  }
  return s;
}
```

集合查询的转换

- 假设 $A = \{a_1, a_2, \ldots, a_n\}$，有些集合查询表面上难以写成分治的形式 (无法从 $f(A)$ 和 $f(B)$ 推出 $f(A \cup B)$)：
  - (rank) 询问 $f(i, j, k) = \{a_p \mid i \leq p \leq j, a_p \leq k\}$
  - (unique) 询问 $f(i, j) = |\{a_k \mid i \leq k \leq j\}|$
  - (triangle) 询问 $f(i, j) = |\{i \leq x, y, z \leq j,\ a_x + a_y > a_z\}| > 0$
  - (min-gap2) 询问 $f(i, j) = \text{min-gap}(\{a_k \mid i \leq k \leq j\})$
  - (mod) 询问 $f(x) = \min\limits_{1 \leq i \leq n} \{a_i \mod x\}$
- 提示：如果无法分治，考虑改变问题的形式，或是预处理的内容

处理单点修改

- 集合查询的本质：分治和组合
  - 预处理了一些数量的 $f(i, j)$，用于查询时拼接使用
  - 如果修改了 k，则所有 $i \leq k \leq j$ 的 $f(i, j)$ 都需要修改
- 数据结构上的修改
  - Sparse-table 不利于修改
  - 线段树利于修改：每个点最多只被 $O(\log n)$ 个预处理区间覆盖
  - 块状链表：每个点只被 1 个区间覆盖且区间的长度都是 $O(\sqrt{n})$，因此适合修改代价很高的操作

处理区间修改

- 考虑一个区间修改涉及了多少个预处理的集合？
  - 块状链表：$O(\sqrt{n})$ 个，只要每个集合能 $O(1)$ 修改，就能求解
  - 线段树：$O(n)$ 个，看起来很困难
- 线段树的区间修改
  - 线段树预处理集的性质：除了根结点，每个区间 $[l,r)$ 都被更大的区间覆盖
  - 我们把区间 $[l,r)$ 的查询转换成 $[l,r_1),[r_1,r_2),\ldots,[r_k,r)$ 的查询，包含这些区间的集合只有 $O(\log n)$ 个
  - 只修改这些集合就能实现区间修改

总结

- 集合查询的核心
  - 将集合分解成子集，把子集的答案合并成查询的答案
  - 不同的分解方法有不同的特点
  - 通常，线段树都是一个很不错的选择
- 如何处理难以从分解答案推出查询答案的问题？
  - 变换问题，或是处理更多的信息
- 如何处理修改？
  - 考虑修改的内容会影响哪些已经求解过的子集