线段树专题1-线段树原理和代码详解

前置知识
讲解020、讲解021、讲解022 - 递归、归并排序、归并分治，有助于理解线段树的代码

线段树专题讲述顺序
专题1：线段树原理和代码详解，讲解110，本节
专题2：线段树的离散化、二分搜索、特别修改，讲解111
专题3：线段树维护更多类型的信息，讲解112
专题4：线段树解决区间合并的问题，讲解113
专题5：开点线段树、区间最值和历史最值，讲解114
专题6：线段树与扫描线结合的题目，讲解115
线段树与动态规划结合的内容，后续【扩展】标签下的课程里继续安排
树套树、可持久化线段树、树链剖分等内容，后续【挺难】标签下的课程里会安排

这个系列一定是全网有关线段树最好的教学视频，觉得好帮忙推荐给身边的人！

左程云

线段树专题1-线段树原理和代码详解

线段树维护的信息类型
父范围上的某个信息，可以用O(1)的时间，从子范围的信息加工得到
满足的信息比如：累加和、最大值、最小值；不满足的信息比如：某范围上出现次数最多的数

线段树的经典功能，如下操作单次调用的时间复杂度为O(log n)
1，范围查询，包括范围内累加和、最大值、最小值等等信息
2，范围修改，包括范围内每个数都增加、重置等等操作

线段树的范围修改功能，想做到单次调用时间复杂度为O(log n)的要求：
一段范围上统一进行了某种修改操作，可以用O(1)的时间，就把这段范围维护的信息加工出来
满足的情况，比如：这段范围所有数字都加v，累加和可以快速的加工出来
不满足的情况，比如：这段范围上每个数字都逆序，累加和不能快速的加工出来

线段树非常灵活，维护信息的种类很多，支持范围修改的类型也很多，整个专题会讲述大量的题目和用法

左程云

线段树专题1-线段树原理和代码详解

线段树的组织，以最经典的累加和举例
1，线段树开始下标可以为1，也可以为0，下标从1开始是最经典的设定
2，线段树需要在初始化时，就指定范围的规模[1 ~ n]，一旦确定不能更改
3，任何一个大范围[l ~ r]，严格从中点mid，拆分成左范围[l ~ mid]、右范围[mid+1 ~ r]
4，每个范围的信息，填写在独立的、连续数组sum中，最大的范围[1 ~ n]，把信息填写在sum[1]
5，如果父范围把信息填写在sum[i]，那么左范围填写在sum[i*2]，右范围填写在sum[i*2+1]
6，范围[l ~ r]和i值的对应，是由公式限制死的，由递归参数维护，无需去记录对应关系
这种设定最为经典，并且为最佳实践，线段树整个专题的代码都遵守这种设定

如果线段树的范围是1 ~ n，那么记录信息的数组，开多大才够用？4倍的n！
为什么？课上重点图解，本节课代码HowManySpace，也展示了这一点
范围1 ~ n，形成的满二叉树高度会达到(log n + 2)，log以2为底
那么这棵满二叉树节点数会达到，2的(logn + 2)次方，log以2为底
2的(logn + 2)次方 = n * 4

左程云

线段树专题1-线段树原理和代码详解

建树过程课上重点图解 void build(l, r, i)

范围查询过程课上重点图解 int query(jobl, jobr, l, r, i)

建树过程时间复杂度O(n)，范围查询单次操作时间复杂度O(log n)

此时还没有牵扯到范围修改这件事

左程云

线段树专题1-线段树原理和代码详解

范围修改操作，以最经典的范围内每个数字都增加来举例
sum数组：范围累加和(查询信息)　　add数组：范围上每个数的增加值(懒信息)

范围内每个数字都增加：void add(jobl, jobr, jobv, l, r, i)
前三个是任务参数，表示jobl ~ jobr范围上，每个数增加jobv，递归过程中这三个参数永远固定
后三个是范围参数，表示当前来到线段树的l ~ r范围上，信息存储位置是i，递归过程中这三个参数可变

开始时调用add(jobl, jobr, jobv, 1, n, 1)，范围增加的递归过程，懒更新机制！课上重点图解
1，如果发现任务范围(jobl, jobr)把当前范围(l,r)全覆盖了，不再向下传递任务，懒住！
　add[i] += jobv; sum[i] += jobv * (r - l + 1);
2，如果任务范围不能把当前范围全包，把该范围上积攒的懒信息，往下只下发一层，down过程
　然后决定当前任务是否要去往，左范围、右范围，继续调用子递归过程
　子递归完成后，利用左右范围的sum信息，把当前范围的sum[i]信息修改正确，up过程
3，退出当前递归过程

左程云

线段树专题1-线段树原理和代码详解

范围查询时，也要结合懒更新机制，增加down过程，课上重点图解

子范围的懒更新信息，发生的时间一定早于父范围上的懒更新信息

如果修改操作不是范围修改，而是单点修改
那么懒更新的机制不需要建立，也不需要懒信息的下发
每次修改都是一走到底，反而更简单
整个专题有很多相关题目讲解

左程云

线段树专题1-线段树原理和代码详解

题目1
线段树支持范围增加、范围查询
维护累加和
测试链接：https://www.luogu.com.cn/problem/P3372

左程云

线段树专题1-线段树原理和代码详解

题目2
线段树支持范围重置、范围查询
维护累加和
对数器验证

左程云

线段树专题1-线段树原理和代码详解

题目3
线段树支持范围增加、范围查询
维护最大值
对数器验证

左程云

线段树专题1-线段树原理和代码详解

题目4
线段树支持范围重置、范围查询
维护最大值
对数器验证

左程云

线段树专题1-线段树原理和代码详解

题目5
线段树同时支持范围重置、范围增加、范围查询
维护累加和
对数器验证

本题需要同时支持范围更新、范围增加的操作

有一个很重要的内容需要掌握：多种修改操作之间的优先级整理

以本题来说，如下事实非常明显：
1，一段范围的更新操作会彻底取消之前的增加操作
2，一段范围的增加操作不会取消之前的更新操作，而是在之前更新操作的基础上进行增加
这两点在懒更新时需要得到体现

左程云

线段树专题1-线段树原理和代码详解

题目6
线段树同时支持范围重置、范围增加、范围查询
维护最大值
测试链接：https://www.luogu.com.cn/problem/P1253

本题需要同时支持范围更新、范围增加的操作

有一个很重要的内容需要掌握：多种修改操作之间的优先级整理

以本题来说，如下事实非常明显：
1，一段范围的更新操作会彻底取消之前的增加操作
2，一段范围的增加操作不会取消之前的更新操作，而是在之前更新操作的基础上进行增加
这两点在懒更新时需要得到体现

线段树专题1-线段树原理和代码详解

线段树常见方法一览
void up(i..)：根据子范围的查询信息，把父范围的查询信息更新正确
void down(i..)：父范围的懒信息，往下只下发一层，给左范围、右范围，然后父范围的懒信息清空
void lazy(i..)：当前范围被修改任务全覆盖时 或 父范围发下来的懒更新时，信息数组们如何修改
void build(l, r, i) ：建树
void update(jobl, jobr, jobv, l, r, i)：范围上数值的重置任务
void add(jobl, jobr, jobv, l, r, i)：范围上数值的增加任务
int query(jobl, jobr, l, r, i)：范围上的信息查询任务

题目不同，需要的方法也不尽相同，可能删减或者增加

线段树用法非常灵活，线段树整个专题会给大家练熟

左程云