各种基础知识点搞搞

东北师大附中
neither_nor

目录

- 单调栈
- 树状数组
- 线段树
- 主席树
- 树套树

单调栈

- 当然要从最简单的开始啦！相信大家都会啦！
- 单调栈嘛，就是一个栈，满足里边的元素是单调（递增/递减）的
- 如果一个元素入栈之后不满足单调性了，就把栈顶先弹出，然后再尝试入栈
- 弹出的元素就再也没有用了，被淘汰了
- 优胜劣汰，适者生存
- 可以用来选拔优秀的人类与外星人作斗争
- 就知识点来讲没什么好说的，主要在于应用

单调栈

- 给定一个 1~n 的排列，对每个 i 求有多少个区间是以 i 为最大值的
- 复杂度要求 O(n)

单调栈

- Mooo
- 有 n 座塔，每个塔有一个高度，每个塔会向左边和右边发射动感光波
- 塔同时会拦截并接收动感光波
- 问每个塔会收到多少动感光波

单调栈

- 最大长方形
- 有一堵墙，每个位置都有一个高度，现在你要在墙上找一个矩形，然后把这个矩形里画满可爱的男孩子，你想让你找到的矩形尽量大

单调栈

- 最大长方形：
- 给定一个 n*m 的 01 矩阵，询问最大的全为 0 的长方形
- 复杂度要求 O(n*m)

单调栈

- BZOJ1510
- 有一个管道，有 n 层，每层都是一个圆形，半径为 ri ，高度为 1
- 现在依次落下了 m 个圆盘，每个圆盘的半径是 Ri ，高度为 1 ，问每个圆盘会落到哪一层

先假设你有一只兔子。

假设有人又给了你另一只兔子。

现在，数一下你所拥有的兔子数量，你会得到结果是两只。也就是说一只兔子加一只兔子等于两只兔子，也就是一加一等于二。

$$1 + 1 = 2$$

这就是算术的运算方法了。

那么，现在你已经对算术的基本原理有了一定了解，就让我们来看一看下面这个简单的例子，来把我们刚刚学到的知识运用到实践中吧。

试试看！
例题 1.7

$$\log \Pi(N) = \left(N + \frac{1}{2}\right) \log N - N + A - \int_N^\infty \frac{\overline{B}_1(x)\mathrm{d}x}{x}, \quad A = 1 + \int_1^\infty \frac{\overline{B}_1(x)\mathrm{d}x}{x}$$

$$\log \Pi(s) = \left(s + \frac{1}{2}\right) \log s - s + A - \int_0^\infty \frac{\overline{B}_1(t)\mathrm{d}t}{t+s}$$

$$\log \Pi(s) = \lim \left[s \log(N+1) + \sum^{N} \log n - \sum^{N} \log(s+n) \right]$$

单调栈

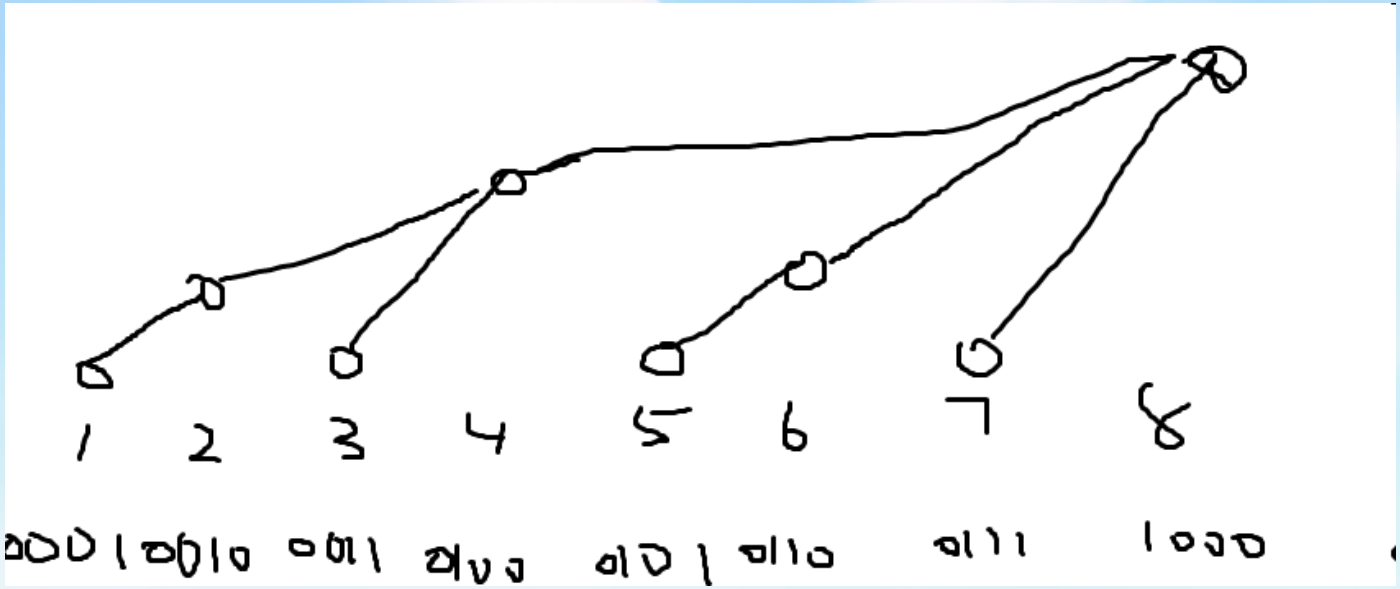
- 现在你已经掌握单调栈的基本运用方法了，让我们做一道题来实践一下吧！
- BZOJ4709 [Jsoi2011] 柠檬
- Flute 很喜欢柠檬。它准备了一串用树枝串起来的贝壳，打算用一种魔法把贝壳变成柠檬。贝壳一共有 N (1 ≤ N ≤ 100,000) 只，按顺序串在树枝上。为了方便，我们从左到右给贝壳编号 1..N 。每只贝壳的大小不一定相同，
- 贝壳 i 的大小为 si(1 ≤ si ≤10,000) 。变柠檬的魔法要求， Flute 每次从树枝一端取下一小段连续的贝壳，并选择一种贝壳的大小 s0 。如果这一小段贝壳中大小为 s0 的贝壳有 t 只，那么魔法可以把这一小段贝壳变成 s0×t^2 只柠檬。 Flute 可以取任意多次贝壳，直到树枝上的贝壳被全部取完。各个小段中， Flute 选择的贝壳大小 s0 可以不同。而最终 Flute 得到的柠檬数，就是所有小段柠檬数的总和。 Flute 想知道，它最多能用这一串贝壳变出多少柠檬。请你帮忙解决这个问题。
- 样例：
- 5
- 2 2 5 2 3
- 最优解是先取 2,2,5,2 ，选择 2 ，获得 2*3^2=18 的柠檬，然后取 3 ，选择 3 ，获得 3*1^2=3 的柠檬，总共可以获得 21 个柠檬

单调栈

- 发现：题目告诉你可以从左边取或者从右边取，而事实上就等价于让你把原序列划分成几段
- 在最优解中，每一段的两个端点一定相同（否则的话把某一端的一个单分一段可以更优）
- 那么我们可以列出 DP 式， f[i]=max(f[j-1]+a*(sa[i]-sa[j]+1)^2) ，其中 i 和 j 的颜色都为 a ， sa 为统计颜色为 a 的贝壳有多少个的前缀和
- 直接 DP 的话是 n^2 的
- 我们发现，对于四个颜色都为 a 的位置 i<j<k<l ，在计算 f[l] 的时候 i 和 j 的贡献相比于计算 f[k] 的时候都增加了，而 i 增加的一定比 j 多（因为函数 y=x^2 上凸）
- 同时我们发现，对于 i 和 j ，在我们知道了 f[i] 和 f[j] 之后就能 O(log n) 计算出 i 变得比 j 优的位置
- 那么我们对每种颜色维护一个单调栈，栈里是最有决策点，并关于前一个超过后一个的时间单调，每次求 DP 值的时候如果发现栈顶第二个元素比栈顶优了就弹栈
- 于是就 O(n log n) 解决了这个问题

树状数组

- 相信大家也是都会的啦
- lowbit(x) 代表 x 的最低的 1 的位以及其后面的 0
- 比如 lowbit(1101100)=100
- i+lowbit(i) 的意义就是把最低的连续的 1 全变成 0 ，然后把再高一位变成 1
- i-lowbit(i) 的意义就是把最低的 1 变成 0
- 那么我们维护一个树状结构， i 的父亲为 i+lowbit(i)



树状数组

- 在修改的时候我们不断使 c[x]+=val ，然后 x+=lowbit(x) ，直到 x>n
- 意义就是每次走到他的父亲
- 在查询的时候我们不断使 ans+=c[x] ，然后 x-=lowbit(x) ，直到 x==0
- 意义就是每次走到最大的不在他子树里的点
- 容易发现每个操作的复杂度都是 O(log n) 的
- 正确性？
- 假设你对 x 进行修改，对 y 进行查询
- 那么如果 x<=y ，那么 x 的影响会在 x 到根的路径上 <=y 的最大的点处被统计一次
- 如果 x>y ，那么 x 的影响不会被统计

树状数组

- 思考：如果修改的时候令 x-=lowbit(x) ，查询的时候令 x+=lowbit(x) ，会是什么效果？

树状数组

- 思考：如果修改的时候令 x-=lowbit(x) ，查询的时候令 x+=lowbit(x) ，会是什么效果？
- 事实上就是变成后缀查询了

线段树

- 线段树是一个很博大精深的东西.....

线段树

- 单点修改 / 单点查询

线段树

- lazy 标记

线段树

- 线段树维护最长连续 1

线段树

- 区间乘（非0数）、加

线段树

- 加等差数列，区间求和

线段树

- 动态开点

线段树

- 动态开点：
- 假设区间范围很大，而又无法进行事先离散化，我们可以对每个点维护其左右儿子，然后在需要的时候如果发现需要的节点没有再新建
- 就像平衡树一样

线段树

- BZOJ4592 [Shoi2015] 脑洞治疗仪
- 曾经发明了自动刷题机的发明家 SHTSC 又公开了他的新发明：脑洞治疗仪 -- 一种可以治疗他因为发明而日益增大的脑洞的神秘装置。
- 为了简单起见，我们将大脑视作一个 01 序列。 1 代表这个位置的脑组织正常工作， 0 代表这是一块脑洞。
- 1 0 1 0 0 0 1 1 1 0
- 脑洞治疗仪修补某一块脑洞的基本工作原理就是将另一块连续区域挖出，将其中正常工作的脑组织填补在这块脑洞中。
- （所以脑洞治疗仪是脑洞的治疗仪？）
- 例如，用上面第 8 号位置到第 10 号位置去修补第 1 号位置到第 4 号位置的脑洞。我们就会得到：
- 1 1 1 1 0 0 1 0 0 0
- 如果再用第 1 号位置到第 4 号位置去修补第 8 号位置到第 10 号位置：
- 0 0 0 0 0 0 1 1 1 1
- 这是因为脑洞治疗仪会把多余出来的脑组织直接扔掉。
- 如果再用第 7 号位置到第 10 号位置去填补第 1 号位置到第 6 号位置：
- 1 1 1 1 0 0 0 0 0 0
- 这是因为如果新脑洞挖出来的脑组织不够多，脑洞治疗仪仅会尽量填补位置比较靠前的脑洞。
- 假定初始时 SHTSC 并没有脑洞，给出一些挖脑洞和脑洞治疗的操作序列，你需要即时回答 SHTSC 的问题：
- 在大脑某个区间中最大的连续脑洞区域有多大。

线段树

- UOJ#228 基础数据结构练习题
- 区间加，区间开根下取整，区间求和

线段树

- zkw 线段树
- 递归版线段树常数巨大
- 是王逸松这种人所不能接受的
- 所以我们可能需要一些 zkw 线段树
- 就是从底向上地做线段树
- Zkw 说他比树状数组常数还小

线段树

- 线段树合并

线段树

- 线段树合并：
- 初始有 n 个只有一个叶子有值的线段树，在合并两个线段树的时候对于共有的节点把信息合并，对于不共有的节点直接返回，则最后把所有线段树合并成一颗的复杂度是 O(n log n)（假设信息的合并是 O(1) 的）
- 证明：对于共有的部分，相当于删去了那么多点，最初有 O(n log n) 个点，最后剩 O(n) 个点，没有新建点的操作，所以总复杂度是 O(n log n) 的

线段树

- BZOJ2702 二叉树
- 现在有一棵二叉树，所有非叶子节点都有两个孩子。在每个叶子节点上有一个权值 (有 n 个叶子节点，满足这些权值为 1..n 的一个排列)。可以任意交换每个非叶子节点的左右孩子。
  要求进行一系列交换，使得最终所有叶子节点的权值按照中序遍历写出来，逆序对个数最少。

线段树

- 3545: [ONTAK2010]Peaks
- 在 Bytemountains 有 N 座山峰，每座山峰有他的高度 h_i 。有些山峰之间有双向道路相连，共 M 条路径，每条路径有一个困难值，这个值越大表示越难走，现在有 Q 组询问，每组询问询问从点 v 开始只经过困难值小于等于 x 的路径所能到达的山峰中第 k 高的山峰，如果无解输出 -1 。

线段树

- BZOJ4730: Alice 和 Bob 又在玩游戏
- Alice 和 Bob 在玩游戏。有 n 个节点， m 条边 (0<=m<=n-1) ，构成若干棵有根树，每棵树的根节点是该连通块内编号最
- 小的点。 Alice 和 Bob 轮流操作，每回合选择一个没有被删除的节点 x ，将 x 及其所有祖先全部删除，不能操作的人输
- 。注：树的形态是在一开始就确定好的，删除节点不会影响剩余节点父亲和儿子的关系。比如： 1-3-2 这样一条链
- ， 1 号点是根节点，删除 1 号点之后， 3 号点还是 2 号点的父节点。问有没有先手必胜策略。 n<=10^5 。

线段树

- 线段树优化网络流 / 最短路：
- 比如建图中出现了某个点向一段区间连边的情况
- 朴素连法连一次是 O(len) 的
- 对序列建一棵线段树，然后把边连到区间对应的线段树节点上
- 复杂度就变成 O(log n) 的了

线段树

- BZOJ3073: [Pa2011]Journeys
- Seter 建造了一个很大的星球，他准备建造 N 个国家和无数双向道路。 N 个国家很快建造好了，用 1..N 编号，但是他发现道路实在太多了，他要一条条建简直是不可能的！于是他以如下方式建造道路： (a,b),(c,d) 表示，对于任意两个国家 x,y ，如果 a<=x<=b,c<=y<=d ，那么在 xy 之间建造一条道路。 Seter 保证一条道路不会修建两次，也保证不会有一个国家与自己之间有道路。
- Seter 好不容易建好了所有道路，他现在在位于 P 号的首都。 Seter 想知道 P 号国家到任意一个国家最少需要经过几条道路。当然， Seter 保证 P 号国家能到任意一个国家。

主席树

- 主席树，就是可持久化线段树
- 什么叫可持久化？就是你要支持访问任意历史版本
- 什么叫历史版本？比如你现在有一颗线段树，这是一个版本，然后你进行了一次修改，这就又多了一个版本

主席树

- 朴素做法？直接复制一份然后修改
- 显然时空复杂度都不可接受

主席树

- 先只考虑单点修改的情况，我们发现进行一次单点修改只会影响根到修改点的路径上 O(log n) 个节点
- 也就是说我们不需要把整棵树都新建一份
- 我们对每个节点都维护左右儿子，然后对于新版本的主席树，把被修改了的儿子新建，没被修改的儿子直接指向上一个版本的儿子
- 时空复杂度 O(log n)

主席树

- 区间 K 小问题
- 给定一个序列，多次询问区间 K 小值

主席树

- 区间修改与区间查询？
- 下传标记会使得之前的版本受到影响
- 当然我们可以每次标记下传的时候都新建节点，不过这会给查询操作也带来 O(log n) 的空间复杂度
- 标记永久化
- 比如维护区间加和区间求和
- 维护两个标记 ch 和 v ，分别表示区间整体加和只考虑子区间的加的情况下的区间和
- 这样就不需要对标记进行上传和下传了

主席树

- 树上主席树：
- 给定一棵树，询问树链 K 大

主席树

- 带修改：
- 区间 K 小问题，支持更改某个点的点权

主席树

- 带修改：
- 外套一个树状数组，事实上这时候就已经不是主席树了，只是树状数组套线段树而已

树套树

- 刚刚我们已经见识了树状数组套线段树了，这其实也是树套树的一种，所谓树套树就是两个树形结构套在一起，外层树的每个节点都是一个内层树
- 最常见的是线段树套平衡树，另外还有树状数组套线段树，二维线段树等等


各种基础知识点搞搞
东北师大附中
neither_nor


目录
单调栈
树状数组
线段树
主席树
树套树

单调栈

- 当然要从最简单的开始啦！相信大家都会啦！
- 单调栈嘛，就是一个栈，满足里边的元素是单调（递增/递减）的
- 如果一个元素入栈之后不满足单调性了，就把栈顶先弹出，然后再尝试入栈
- 弹出的元素就再也没有用了，被淘汰了
- 优胜劣汰，适者生存
- 可以用来选拔优秀的人类与外星人作斗争
- 就知识点来讲没什么好说的，主要在于应用

单调栈

- 给定一个 1~n 的排列，对每个 i 求有多少个区间是以 i 为最大值的
- 复杂度要求 O(n)

单调栈

- Mooo
- 有 n 座塔，每个塔有一个高度，每个塔会向左边和右边发射动感光波
- 塔同时会拦截并接收动感光波
- 问每个塔会收到多少动感光波

单调栈

- 最大长方形
- 有一堵墙，每个位置都有一个高度，现在你要在墙上找一个矩形，然后把这个矩形里画满可爱的男孩子，你想让你找到的矩形尽量大

单调栈

- 最大长方形：
- 给定一个 n*m 的 01 矩阵，询问最大的全为 0 的长方形
- 复杂度要求 O(n*m)

单调栈

- BZOJ1510
- 有一个管道，有 n 层，每层都是一个圆形，半径为 ri ，高度为 1
- 现在依次落下了 m 个圆盘，每个圆盘的半径是 Ri ，高度为 1 ，问每个圆盘会落到哪一层

算术入门

先假设你有一只兔子。

假设有人又给了你另一只兔子。

现在，数一下你所拥有的兔子数量，你会得到结果是两只。也就是说一只兔子加一只兔子等于两只兔子，也就是一加一等于二。

$$1 + 1 = 2$$

这就是算术的运算方法了。

那么，现在你已经对算术的基本原理有了一定了解，就让我们来看一看下面这个简单的例子，来把我们刚刚学到的知识运用到实践中吧。

试试看！
例题 1.7

$$\log\Pi(N) = \left(N + \frac{1}{2}\right)\log N - N + A - \int_N^\infty \frac{\overline{B}_1(x)\mathrm{d}x}{x}, \quad A = 1 + \int_1^\infty \frac{\overline{B}_1(x)\mathrm{d}x}{x}$$

$$\log\Pi(s) = \left(s + \frac{1}{2}\right)\log s - s + A - \int_0^\infty \frac{\overline{B}_1(t)\mathrm{d}t}{t+s}$$

$$\log\Pi(s) = \lim\left[s\log(N+1) + \sum_1^N \log n - \sum_1^N \log(s+n)\right]$$

单调栈

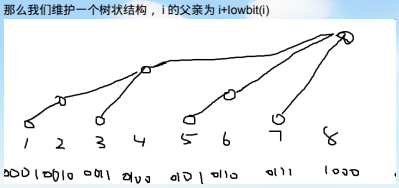
- 现在你已经掌握单调栈的基本运用方法了，让我们做一道题来实践一下吧！
- BZOJ4709 [Jsoi2011] 柠檬
- Flute 很喜欢柠檬。它准备了一串用树枝串起来的贝壳，打算用一种魔法把贝壳变成柠檬。贝壳一共有 N (1 ≤ N ≤ 100,000) 只，按顺序串在树枝上。为了方便，我们从左到右给贝壳编号 1..N 。每只贝壳的大小不一定相同，
- 贝壳 i 的大小为 si(1 ≤ si ≤10,000)。变柠檬的魔法要求，Flute 每次从树枝一端取下一小段连续的贝壳，并选择一种贝壳的大小 s0 。如果这一小段贝壳中大小为 s0 的贝壳有 t 只，那么魔法可以把这一小段贝壳变成 s0×t^2 只柠檬。Flute 可以取任意多次贝壳，直到树枝上的贝壳被全部取完。各个小段中，Flute 选择的贝壳大小 s0 可以不同。而最终 Flute 得到的柠檬数，就是所有小段柠檬数的总和。Flute 想知道，它最多能用这一串贝壳变出多少柠檬。请你帮忙解决这个问题。
- 样例：
- 5
- 2 2 5 2 3
- 最优解是先取 2,2,5,2 ，选择 2 ，获得 2*3^2=18 的柠檬，然后取 3 ，选择 3 ，获得 3*1^2=3 的柠檬，总共可以获得 21 个柠檬

单调栈

- 发现：题目告诉你可以从左边取或者从右边取，而事实上就等价于让你把原序列划分成几段
- 在最优解中，每一段的两个端点一定相同（否则的话把某一端的一个单分一段可以更优）
- 那么我们可以列出 DP 式，f[i]=max(f[j-1]+a*(sa[i]-sa[j]+1)^2)，其中 i 和 j 的颜色都为 a，sa 为统计颜色为 a 的贝壳有多少个的前缀和
- 直接 DP 的话是 n^2 的
- 我们发现，对于四个颜色都为 a 的位置 i<j<k<l，在计算 f[l] 的时候 i 和 j 的贡献相比于计算 f[k] 的时候都增加了，而 i 增加的一定比 j 多（因为函数 y=x^2 上凸）
- 同时我们发现，对于 i 和 j，在我们知道了 f[i] 和 f[j] 之后就能 O(log n) 计算出 i 变得比 j 优的位置
- 那么我们对每种颜色维护一个单调栈，栈里是最有决策点，并关于前一个超过后一个的时间单调，每次求 DP 值的时候如果发现栈顶第二个元素比栈顶优了就弹栈
- 于是就 O(n log n) 解决了这个问题

树状数组

- 相信大家也是都会的啦
- lowbit(x) 代表 x 的最低的 1 的位以及其后面的 0
- 比如 lowbit(1101100)=100
- i+lowbit(i) 的意义就是把最低的连续的 1 全变成 0 ，然后把再高一位变成 1
- i-lowbit(i) 的意义就是把最低的 1 变成 0
- 那么我们维护一个树状结构， i 的父亲为 i+lowbit(i)



树状数组

- 在修改的时候我们不断使 c[x]+=val ，然后 x+=lowbit(x) ，直到 x>n
- 意义就是每次走到他的父亲
- 在查询的时候我们不断使 ans+=c[x] ，然后 x-=lowbit(x) ，直到 x==0
- 意义就是每次走到最大的不在他子树里的点
- 容易发现每个操作的复杂度都是 O(log n) 的
- 正确性？
- 假设你对 x 进行修改，对 y 进行查询
- 那么如果 x<=y ，那么 x 的影响会在 x 到根的路径上 <=y 的最大的点处被统计一次
- 如果 x>y ，那么 x 的影响不会被统计

树状数组

- 思考：如果修改的时候令 x-=lowbit(x)，查询的时候令 x+=lowbit(x)，会是什么效果？

树状数组

- 思考：如果修改的时候令 x-=lowbit(x) ，查询的时候令 x+=lowbit(x) ，会是什么效果？
- 事实上就是变成后缀查询了


线段树
线段树是一个很博大精深的东西.....


线段树
单点修改 / 单点查询


线段树
lazy 标记


线段树
线段树维护最长连续 1


线段树
区间乘（非 0 数）、加


线段树
加等差数列，区间求和


线段树
动态开点

线段树

- 动态开点：
- 假设区间范围很大，而又无法进行事先离散化，我们可以对每个点维护其左右儿子，然后在需要的时候如果发现需要的节点没有再新建
- 就像平衡树一样

线段树

- BZOJ4592 [Shoi2015] 脑洞治疗仪
- 曾经发明了自动刷题机的发明家 SHTSC 又公开了他的新发明：脑洞治疗仪 -- 一种可以治疗他因为发明而日益增大的脑洞的神秘装置。
- 为了简单起见，我们将大脑视作一个 01 序列。 1 代表这个位置的脑组织正常工作， 0 代表这是一块脑洞。
- 1 0 1 0 0 0 1 1 1 0
- 脑洞治疗仪修补某一块脑洞的基本工作原理就是将另一块连续区域挖出，将其中正常工作的脑组织填补在这块脑洞中。
- （ 所以脑洞治疗仪是脑洞的治疗仪？ ）
- 例如，用上面第 8 号位置到第 10 号位置去修补第 1 号位置到第 4 号位置的脑洞。我们就会得到：
- 1 1 1 1 0 0 1 0 0 0
- 如果再用第 1 号位置到第 4 号位置去修补第 8 号位置到第 10 号位置：
- 0 0 0 0 0 0 1 1 1 1
- 这是因为脑洞治疗仪会把多余出来的脑组织直接扔掉。
- 如果再用第 7 号位置到第 10 号位置去填补第 1 号位置到第 6 号位置：
- 1 1 1 1 0 0 0 0 0 0
- 这是因为如果新脑洞挖出来的脑组织不够多，脑洞治疗仪仅会尽量填补位置比较靠前的脑洞。
- 假定初始时 SHTSC 并没有脑洞，给出一些挖脑洞和脑洞治疗的操作序列，你需要即时回答 SHTSC 的问题：
- 在大脑某个区间中最大的连续脑洞区域有多大。


线段树
UOJ#228 基础数据结构练习题
区间加，区间开根下取整，区间求和

线段树

- zkw 线段树
- 递归版线段树常数巨大
- 是王逸松这种人所不能接受的
- 所以我们可能需要一些 zkw 线段树
- 就是从底向上地做线段树
- Zkw 说他比树状数组常数还小


线段树
线段树合并

线段树

- 线段树合并：
- 初始有 n 个只有一个叶子有值的线段树，在合并两个线段树的时候对于共有的节点把信息合并，对于不共有的节点直接返回，则最后把所有线段树合并成一颗的复杂度是 O(n log n)（假设信息的合并是 O(1) 的）
- 证明：对于共有的部分，相当于删去了那么多点，最初有 O(n log n) 个点，最后剩 O(n) 个点，没有新建点的操作，所以总复杂度是 O(n log n) 的

线段树

- BZOJ2702 二叉树
- 现在有一棵二叉树，所有非叶子节点都有两个孩子。在每个叶子节点上有一个权值（有 n 个叶子节点，满足这些权值为 1..n 的一个排列）。可以任意交换每个非叶子节点的左右孩子。
  要求进行一系列交换，使得最终所有叶子节点的权值按照中序遍历写出来，逆序对个数最少。

线段树

- 3545: [ONTAK2010]Peaks
- 在 Bytemountains 有 N 座山峰，每座山峰有他的高度 h_i。有些山峰之间有双向道路相连，共 M 条路径，每条路径有一个困难值，这个值越大表示越难走，现在有 Q 组询问，每组询问询问从点 v 开始只经过困难值小于等于 x 的路径所能到达的山峰中第 k 高的山峰，如果无解输出 -1 。

线段树

- BZOJ4730: Alice 和 Bob 又在玩游戏
- Alice 和 Bob 在玩游戏。有 n 个节点， m 条边 (0<=m<=n-1) ，构成若干棵有根树，每棵树的根节点是该连通块内编号最
- 小的点。 Alice 和 Bob 轮流操作，每回合选择一个没有被删除的节点 x ，将 x 及其所有祖先全部删除，不能操作的人输
- 。注：树的形态是在一开始就确定好的，删除节点不会影响剩余节点父亲和儿子的关系。比如： 1-3-2 这样一条链
- ， 1 号点是根节点，删除 1 号点之后， 3 号点还是 2 号点的父节点。问有没有先手必胜策略。 n<=10^5 。

线段树

- 线段树优化网络流 / 最短路：
- 比如建图中出现了某个点向一段区间连边的情况
- 朴素连法连一次是 O(len) 的
- 对序列建一棵线段树，然后把边连到区间对应的线段树节点上
- 复杂度就变成 O(log n) 的了

线段树

- BZOJ3073: [Pa2011]Journeys
- Seter 建造了一个很大的星球，他准备建造 N 个国家和无数双向道路。 N 个国家很快建造好了，用 1..N 编号，但是他发现道路实在太多了，他要一条条建简直是不可能的！于是他以如下方式建造道路： (a,b),(c,d) 表示，对于任意两个国家 x,y ，如果 a<=x<=b,c<=y<=d ，那么在 xy 之间建造一条道路。 Seter 保证一条道路不会修建两次，也保证不会有一个国家与自己之间有道路。
- Seter 好不容易建好了所有道路，他现在在位于 P 号的首都。 Seter 想知道 P 号国家到任意一个国家最少需要经过几条道路。当然， Seter 保证 P 号国家能到任意一个国家。

主席树

- 主席树，就是可持久化线段树
- 什么叫可持久化？就是你要支持访问任意历史版本
- 什么叫历史版本？比如你现在有一颗线段树，这是一个版本，然后你进行了一次修改，这就又多了一个版本


主席树
朴素做法？直接复制一份然后修改
显然时空复杂度都不可接受

主席树

- 先只考虑单点修改的情况，我们发现进行一次单点修改只会影响根到修改点的路径上 O(log n) 个节点
- 也就是说我们不需要把整棵树都新建一份
- 我们对每个节点都维护左右儿子，然后对于新版本的主席树，把被修改了的儿子新建，没被修改的儿子直接指向上一个版本的儿子
- 时空复杂度 O(log n)


主席树
区间 K 小问题
给定一个序列，多次询问区间 K 小值

主席树

- 区间修改与区间查询？
- 下传标记会使得之前的版本受到影响
- 当然我们可以每次标记下传的时候都新建节点，不过这会给查询操作也带来 O(log n) 的空间复杂度
- 标记永久化
- 比如维护区间加和区间求和
- 维护两个标记 ch 和 v，分别表示区间整体加和只考虑子区间的加的情况下的区间和
- 这样就不需要对标记进行上传和下传了


主席树
树上主席树：
给定一棵树，询问树链 K 大


主席树
• 带修改：
• 区间 K 小问题，支持更改某个点的点权

主席树

- 带修改：
- 外套一个树状数组，事实上这时候就已经不是主席树了，只是树状数组套线段树而已

树套树

- 刚刚我们已经见识了树状数组套线段树了，这其实也是树套树的一种，所谓树套树就是两个树形结构套在一起，外层树的每个节点都是一个内层树
- 最常见的是线段树套平衡树，另外还有树状数组套线段树，二维线段树等等