基础数据结构

Shinetism

YALI

July 22, 2019

基础数据结构

- 数据结构是NOIP中的一大考点，考察的范围较广，但总体而言偏向于简单数据结构。
- 我们今天就讲一些简单实用的数据结构，全部属于NOIP范围，当然会有稍微难度大一点的题目。
- 像队列和栈今天就不讲解了。

目录

- 二叉堆
- 线段树
- 树状数组
- 并查集
- 按秩合并并查集
- Trie
- Hash Table
- Sparse Table
- 树上倍增

二叉堆

最大最小!

二叉堆

- 关于堆，大家都明白，它是一种能够实现下面操作的数据结构：
  - 插入一个元素
  - 取出最值元素
- 那么，二叉堆是一种常用的实现方式，它的每一次操作复杂度均为$O(\log n)$。
- 其实二叉堆是可以支持删除操作的，如果那样不如使用`std::set`。

二叉堆

- 我们看看代码实现的三种方式:
  - 自己用数组实现（比较麻烦）
  - 使用std::priority_queue（最常用，最快捷）
  - 使用std::make_heap,std::push_heap,std::pop_heap（卡常数专用）
- 以上函数均不属于C++11，可以放心使用。
- 关于二叉堆的题目，大多是贪心题或者DP优化。

T1 「BZOJ1528」[POI2005] sam – Toy Cars

- Description:
  - 地板上最多放k个玩具，柜子里一共有n个玩具。
  - 每一次操作一定将柜子里的一个玩具放到地板上，同时可以将地板上的一个玩具放到柜子里。
  - 已知Jasio想玩玩具编号的序列（长度为p），他只能玩在地板上的玩具。现在地板上没有玩具，求满足Jasio要求的最小操作次数。
- Restraints:
  - $1 \le k \le n \le 1e5, 1 \le p \le 5e5$

T1 「BZOJ1528」[POI2005] sam – Toy Cars

- 首先得到贪心结论，当地板上满了的时候，我们将下一次出现最远的玩具放回柜子里。
- 用堆维护即可。

线段树

分治!

线段树

- 线段树可以说是入门基础数据结构。
- 它有非常多的种类，最重要的是，大家要形成自己固定的写法，不要每次写得都不一样（比如函数名、是否动态等），那样会大大降低效率。
- 我们平常讲的线段树，通常支持区间、单点操作，区间、单点询问（当然对操作和询问有限制）。
- 我们讲几道

T2 「BZOJ3211」花神游历各国

- Description:
  - 给出一个序列$\{a_n\}$，进行m次操作，每次可以将$[l,r]$中的每个数开根号向下取整，也可以询问区间$[l,r]$中的数字和。
- Restraints:
  - $n \le 1e5, m \le 2e5, 0 \le a_i < 1e9$

T2 「BZOJ3211」花神游历各国

- 我们想一想，一个10^9以内的数，最多被开多少次根号之后变为1?
- 用计算器算一下，发现只有5次。（实际上是$O(\log\log n)$）
- 所以我们可以放心暴力操作，维护一个标记表示这个区间是否全部为1。
- 到达一个节点时，如果全部全部为1，我们直接返回，否则往下走。
- Time Complexity:$O(n\log n\log\log n)$

T3「LOJ2019」「AHOI / HNOI2017」影魔

- Description:
  - 给定一个长度为n的排列$\{a_n\}$，对于一个区间$[l,r]$，设$c=\min\limits_{l<i<r} a_i$，如果$c<a_l$且$\mathrm{c}<\mathrm{a_r}$，则有$p_1$的贡献，如果$a_l<c<a_r$或$a_r<c<a_l$则有$p_2$的贡献。现在给你$q$个询问，每次询问一个区间$[L,R]$，问所有被区间完全包含的区间的贡献和。
- Restraints:
  - $1\le n,q\le 2e5, 1\le p_1,p_2\le 1e3$

T3「LOJ2019」「AHOI / HNOI2017」影魔

- 题目问的是某区间包含的区间。
- 一个区间包含的区间数目是平方级别的，不是很方便在序列上处理。
- 所以我们尝试将区间转化为二维平面上的一个点。
- 也就是说点(l,r)代表区间$[l,r]$的贡献。
- 那么询问$[l,r]$所包含的区间的贡献，即是该点右下方矩形内贡献的和。

T3「LOJ2019」「AHOI / HNOI2017」影魔

- 怎样把贡献对应到每个区间上?
- 我们处理出$r(i)$表示i右边第一个大于i的值的位置，那么区间$\forall j \in [i, r(i)), [i, j]$有$p_2$的贡献，对应二维平面中一列。
- 对于右端点，反过来就行了。
- 如果是左边次大值，右边最大值，那么肯定是区间$[i, r(i)]$，对这个点加上$p_1 - p_2$就行了。
- 反之亦然。

T3「LOJ2019」「AHOI / HNOI2017」影魔

- 现在问题变为了，对于某列进行区间加，对于矩形进行询问（注意该矩形的特点）
- 我们用扫描线从右往左扫，对某列的区间加后不撤销，那么矩形询问就变为了区间询问。
- 扫描线+线段树
- Time Complexity:$O(n\log n)$

树状数组

少快好写!

树状数组

- 当问题仅限于单点修改、区间（单点）查询时（或者用差分实现区间修改，单点查询），可以使用树状数组代替线段树。
- 树状数组一个最大特点是常数小！好写！
- 它每次操作的$O(\log n)$通常是不满的，而且五分钟不要就可以写完。
- 缺点是不够灵活，操作有限。

T4「LOJ112」三维偏序

- Description:
  - 有n个元素，第i个元素有a_i、b_i、c_i三个属性，设$f(i)$表示满足$a_j \le a_i$且$b_j \le b_i$ 且$c_j \le c_i$的j的数量。对于$d \in [0, n)$，求$f(i) = d$的i的数量。
- Restraints:
  - $1 \le n \le 1e5, 1 \le \max(a_i, b_i, c_i) \le 2e5$

T4「LOJ112」三维偏序

- 三维偏序我们可以一维一维地解决。
- 简单来说，就是一维排序、二维分治、三维树状数组。
- 具体来讲，我们先把数组按照a_i排序。接着分治，在合并时，考虑前面部分对后面部分的影响。两部分都已经变成按照b_i排序的了，我们只要弄两个指针，把前面的点的c_i插入树状数组，后面的点进行查询，最后归并成按照b_i排序，然后返回。
- 时间复杂度$O(n\log^2 n)$。

T5「BZOJ1452」[JSOI2009] Count

- Description:
  - 给出一个$n \times m$的网格，初始时每个格子有一个整数权值。接下来有q次操作，共两种操作：
    - 改变一个格子的权值
    - 求一个子矩阵中某种特定权值的出现次数
- Restraints:
  - $n, m \le 300, q \le 2e5, 1 \le$ 权值 ≤ 100

T5「BZOJ1452」[JSOI2009] Count

- 我们看到权值的种类很少，不妨对每一种权值分别考虑。
- 也就变成了二维平面内单点修改，子矩阵求和。
- 使用二维树状数组（非常好写！）
- Time Complexity:$O((mn+q)\log^2 n)$

并查集

巧妙的集合结构!

并查集

- 并查集是一种功能简单的数据结构，它支持一下两种操作：
  - 将两个集合合并
  - 查询某元素属于哪个集合
- 而并查集的时间复杂度为$O(\alpha(n))$，几乎就是常数。
- 单独考察并查集的题目并不多，难度也不是很大。

T6「POJ1182」食物链

- Description:
  - 有三种动物A,B,C形成了环形食物关系（A吃B，B吃C，C吃A，虽然这并不科学）
  - 现在有n只动物，但是我们并不清楚它们是哪种动物。
  - 有人对动物间关系进行K次如下两种描述：
    - X和Y是同类
    - X吃Y
  - 如果一个描述不自我矛盾也不与前文矛盾，就认为它是正确的，否则认为是错误的。
  - 输出假话总数。
- Restraints:
  - $1 \le n \le 5e4, 0 \le K \le 1e5$

T6「POJ1182」食物链

- 这道题稍有构造性。
- 对于每一个动物，我们建立三个点，表示该动物是A、该动物是B、该动物是C。
- 一个集合的点必须同时成立。初始时每个点单独形成一个集合。
- 遇到两种关系分别连边，如果同一个动物的任意两个点在同一个集合中，则该描述错误。

一道经典题

- Description:
  - 给出一张n个点m条边的无向图，每个点有一个权值，接下来有q次两种操作:
    - 删除一条边
    - 询问某个点所在连通块的最大点权
- Restraints:
  - $1 \le n, m, q \le 1e5$

一道经典题

- 逆向操作。
- 首先把所有要删的边删掉，反向处理询问。
- 只要加边求连通块最大权值。
- 直接并查集即可。

按秩合并并查集

更加灵活!

按秩合并并查集

- 所谓“秩”就是集合的大小，按秩合并就是指按照两集合大小合并。
- 我们不用路径压缩，而是维护$size(u)$表示集合大小，每一次合并时，将小的集合的父亲设为大的集合。
- 按秩合并并查集的深度为$O(\log n)$，它比路径压缩要慢些。
- 但是它能实现更多操作，比如撤销。（关于撤销的题目，都涉及到更为复杂的数据结构譬如LCT，今天先不讲，之后大家可以去看看这些题，比如BZOJ4025）
- 我们今天讲一道稍微简单一点的题目。

T7「BZOJ4668」冷战

- Description:
  - 给出n个点的图，初始时没有边。
  - 有q次两种操作：加边；询问两点之间最早在加入第几条边后联通（如果到这次询问仍未联通，输出0）
  - 强制在线。
- Restraints:
  - $1 \leq n, q \leq 5e5$

T7「BZOJ4668」冷战

- 我们直接用按秩合并的并查集维护。
- 在合并时，加一个边权表示时间戳（这是第几次加边）
- 询问时，我们要求u到v路径上的最小边权。
- 因为深度是$O(\log n)$的，所以可以暴力LCA。

Trie

前缀!

Trie

- Trie的朴素应用非常简单（插入字符串、查询字符串）。
- 一般不会把朴素应用当作一个知识点（因为太简单了，考了也不算考点）
- 所以Trie一般用于和前缀有关的题目中。
- 我们来看一道题。

T8「BZOJ1954」The xor – longest Path

- Description:
  - 给定一棵n个点的带权树，求树上最长的异或和路径。
- Restraints:
  - $1 \le n \le 1e5,$ 权值 $< 2^{31}$

T8「BZOJ1954」The xor – longest Path

- 异或具有两个性质，一是它可以按照二进制位分开考虑，二是同一个数字异或偶数次等于零。
- 运用第二个性质，对于每一个节点，我们算出它到根节点路径上的权值异或和$d(i)$，那么题目就是问$\max\limits_{i,j}\{d(i) \otimes d(j)\}$。
- 我们把所有$d(i)$按照二进制位从高到低插入Trie中。
- 接着运用第一个性质，对于每一个$d(i)$，到Trie中查找最大的，即每一位存在相反的方向就往该方向走。

一个民间流传的数据结构——二进制Trie

- 所谓二进制Trie，就是前一题中用到的这种类型的Trie。
- 为什么要单独拿出来讲？因为它几乎可以完全代替权值线段树。
- 它一样可以打标记、可持久化，也同样是动态开点。
- 只是不用每次算中点值（它的总值域是$[0,2^k-1]$，但这并不影响，因为它是动态开点的），常数更小。
- 以后大家可以试一试它。

Hash Table

$O(1)$!

Hash Table

- Hash表是实现集合或映射的一种数据结构。
- 它支持插入、删除、查询某元素（可以有多个键值）的操作。
- 我们将第一键值对某个质数取模（该质数一般比元素个数稍大）。
- 每个余数都挂一个链表。
- 时间复杂度$O(1)$。

Hash Table

- 将Hash表作为考点的题目真的少，大多涉及超过NOIP范围的考点。
- 所以我们讲一个小应用。

T9 离散对数

- Description:
  - 已知正整数a, b, m，且$\gcd(a, m) = 1$，求满足$a^x \equiv b (mod\ m)$的最小自然数x。
- Restraints:
  - $a, b, m \le 1e16$

T9 离散对数

- 我们设$c = \lceil\sqrt{m}\rceil$，若$a^x \equiv b(mod\ m)$，假设$x = pc - q(1 \le p \le c, 1 \le q \le c)$
- 那么，可以推出$a^{cp} \equiv ba^q(mod\ m)$，我们把右边所有数插入Hash表中，然后枚举左边的值到Hash表中查询。
- 这样就得到了我们需要的x。
- Time Complexity: $O(\sqrt{m})$

Sparse Table

离线高效算法!

Sparse Table

- Sparse Table又称ST表，是大家常用来解决RMQ问题的一种数据结构。
- 它支持查询区间最值的操作。
- 下面讲个关联不太大的题目。

T10 「BZOJ2006」[NOI2010]超级钢琴

- Description:
  - 给出一个长度为n的整数序列$\{a_n\}$，选出K个长度在$[L,R]$的不同的区间（可以相交），使得所有区间内的权值和最大。求最大值。
- Restraints:
  - $n, K \le 5e5, -1000 \le a_i \le 1000$

T10 「BZOJ2006」[NOI2010]超级钢琴

- 首先，我们把每一个右端点对应的最优左端点（用ST表求）放进优先队列。
- 然后每一次从优先队列中取出一个元素，之后加入对应右端点的次优左端点。
- 这个次优怎么求呢?
- 我们在优先队列中放的元素可以是四元组(a,b,r,val)，其中$[a,b]$为左端点可以取的范围，val为最优值。
- 然后每次取出来之后将区间分裂，放入两个元素就可以了。
- Time Complexity:$O(K\log n)$

树上倍增

将ST表扩展到树上！

树上倍增

- 简单来说，树上倍增就是将倍增这一套方法搬到了树上。
- 一个重要应用就是求LCA，接下来给大家讲解一下。

- 下面做一道简单题。

T11 「BZOJ3306」树

- Description:
  - 给定一颗n个点的树，支持以下操作:
    - 换根
    - 修改点权
    - 查询子树最小值
- Restraints:
  - $n, q \leq 1e5$

T11 「BZOJ3306」树

- 求出树的dfs序列，记录一下根，分类讨论一下情况（要用到LCA）
- 然后问题转化为单点修改，区间查询最小值。
- 线段树解决。

完结撒花~