线段树入门

by LittleRed

2017 年 1 月 X 日

Contents

Outline

关于数据结构

- 什么是数据结构?

关于数据结构

- 什么是数据结构？

关于数据结构

- 什么是数据结构?
- 通过一些方法来维护信息，以实现快速存储和查询的目的.

关于数据结构

- 什么是数据结构?

- 通过一些方法来维护信息，以实现快速存储和查询的目的.
- 在OI竞赛中通常是以各种形式的问题出现.

关于数据结构

- 什么是数据结构?

- 通过一些方法来维护信息，以实现快速存储和查询的目的.
- 在OI竞赛中通常是以各种形式的问题出现.
- 常见的数据结构题通常都需要经过几个步骤的约化.

关于数据结构

- 什么是数据结构?

- 通过一些方法来维护信息，以实现快速存储和查询的目的.
- 在OI竞赛中通常是以各种形式的问题出现.
- 常见的数据结构题通常都需要经过几个步骤的约化.
- 近年的OI竞赛很少出现不需要动脑子的数据结构题.

关于数据结构

- 有必要认识到数据结构是用来解决问题的工具而不是核心.

关于数据结构

- 有必要认识到数据结构是用来解决问题的工具而不是核心.
- 这是数据结构和算法的区别所在.

关于数据结构

- 有必要认识到数据结构是用来解决问题的工具而不是核心.
- 这是数据结构和算法的区别所在.
- 一个有多项式复杂度解的问题即使不使用数据结构，它仍然存在一个多项式复杂度的解.

关于数据结构

- 有必要认识到数据结构是用来解决问题的工具而不是核心.
- 这是数据结构和算法的区别所在.
- 一个有多项式复杂度解的问题即使不使用数据结构，它仍然存在一个多项式复杂度的解.
- 数据结构的任务是优化算法中的某些存储和查询过程，来做到更优秀的时空复杂度.

说明

- 笔者不保证这个课件中的所有内容都是正确的.

说明

- 笔者不保证这个课件中的所有内容都是正确的.
- 由于笔者自身水平极为有限，这里只能向你们介绍一些入门级的内容.

说明

- 笔者不保证这个课件中的所有内容都是正确的.
- 由于笔者自身水平极为有限，这里只能向你们介绍一些入门级的内容.
- 你们要学会自己去学习.

说明

- 笔者不保证这个课件中的所有内容都是正确的.
- 由于笔者自身水平极为有限，这里只能向你们介绍一些入门级的内容.
- 你们要学会自己去学习.
- 当有一天你们认为这个课件对你们已经没有帮助时，那笔者也就达到写这个课件的目的了.

常见术语

- 离线:

常见术语

- 离线:
- 预先读取所有操作，以达到改变回答询问顺序或是根据操作来预处理出某些信息的目的.

常见术语

- 离线:
- 预先读取所有操作，以达到改变回答询问顺序或是根据操作来预处理出某些信息的目的.
- 强制在线:

常见术语

- 离线:
- 预先读取所有操作，以达到改变回答询问顺序或是根据操作来预处理出某些信息的目的.
- 强制在线:
- 在构造数据时通过上次询问的答案来加密操作,以达到强制按给出询问的顺序来回答询问的目的，如果你回答某个询问的结果不正确，那你就无法知道接下来的操作是什么.

Outline

1. 概念
   - 数据结构
2. 线段树
   - 简介
   - 如何维护信息
   - 作为数据结构的线段树
3. 应用技巧
   - 一些简单的问题
   - 一些复杂的问题
4. 结语

线段树

- 在OI竞赛中极为实用的数据结构.

线段树

- 在OI竞赛中极为实用的数据结构.
- 学习这种数据结构只需要一点关于二叉树的知识就够了.

线段树

- 在OI竞赛中极为实用的数据结构.
- 学习这种数据结构只需要一点关于二叉树的知识就够了.
- 很多时候线段树都可以成为平衡树的代替品.

线段树

- 在OI竞赛中极为实用的数据结构.
- 学习这种数据结构只需要一点关于二叉树的知识就够了.
- 很多时候线段树都可以成为平衡树的代替品.
- 就OI竞赛而言，线段树的代码实现比各种平衡树都简单的多.

线段树

- 在OI竞赛中极为实用的数据结构.
- 学习这种数据结构只需要一点关于二叉树的知识就够了.
- 很多时候线段树都可以成为平衡树的代替品.
- 就OI竞赛而言，线段树的代码实现比各种平衡树都简单的多.
- 这也是它实用的众多原因之一.

基本概念

- 这是一种把区间（线段）划分成一些部分来维护信息的数据结构.

基本概念

- 这是一种把区间（线段）划分成一些部分来维护信息的数据结构.
- 树上的每一个节点都维护一个原序列的子区间.

基本概念

- 这是一种把区间（线段）划分成一些部分来维护信息的数据结构.
- 树上的每一个节点都维护一个原序列的子区间.
- 区间中的同一个位置的信息会被多个树上节点（其实是从根出发到叶子的一条链）维护.

基本概念

- 这是一种把区间（线段）划分成一些部分来维护信息的数据结构.
- 树上的每一个节点都维护一个原序列的子区间.
- 区间中的同一个位置的信息会被多个树上节点（其实是从根出发到叶子的一条链）维护.
- 这是为了加快查询速度.

基本概念

- 线段树是一颗平衡二叉树.

基本概念

- 线段树是一颗平衡二叉树.
- 在线段树中一个节点表示一段区间.

基本概念

- 线段树是一颗平衡二叉树.
- 在线段树中一个节点表示一段区间.
- 一个节点如果表示的是区间中的一段长度为1的区间，则它在线段树中没有儿子.

基本概念

- 线段树是一颗平衡二叉树.
- 在线段树中一个节点表示一段区间.
- 一个节点如果表示的是区间中的一段长度为1的区间，则它在线段树中没有儿子.
- 否则它至少有一个左儿子或右儿子.

基本概念

- 线段树是一颗平衡二叉树.
- 在线段树中一个节点表示一段区间.
- 一个节点如果表示的是区间中的一段长度为1的区间，则它在线段树中没有儿子.
- 否则它至少有一个左儿子或右儿子.
- 根节点表示整个区间.

基本概念

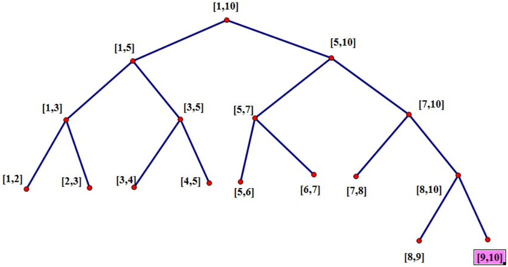
- 为了保证树高，我们让一个节点的左右儿子各代表它自己表示的区间从中间分开的两部分.

基本概念

- 为了保证树高，我们让一个节点的左右儿子各代表它自己表示的区间从中间分开的两部分.
- 这样每次区间长度都会缩小为原来的一半，如果我们要用一颗线段树维护一个长度为N的区间，线段树的树高就是$O(\log N)$级别的.

基本概念

- 为了保证树高，我们让一个节点的左右儿子各代表它自己表示的区间从中间分开的两部分.
- 这样每次区间长度都会缩小为原来的一半，如果我们要用一颗线段树维护一个长度为N的区间，线段树的树高就是$O(\log N)$级别的.
- 下图为一颗维护了长度为10的区间的线段树的形态.



重要性质1

- 线段树的节点总数是$O(N)$级别的.

重要性质1

- 线段树的节点总数是$O(N)$级别的.
- 实际上节点个数是一个类似于$\sum_{i=1}^{\log N} 2^{i-1}$的数量级.

重要性质1

- 线段树的节点总数是$O(N)$级别的.
- 实际上节点个数是一个类似于$\sum_{i=1}^{\log N} 2^{i-1}$的数量级.
- 把这个等比数列求和以后可以得到点数大概是2*N左右.

重要性质1

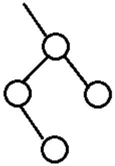
- 线段树的节点总数是$O(N)$级别的.
- 实际上节点个数是一个类似于$\sum_{i=1}^{\log N} 2^{i-1}$的数量级.
- 把这个等比数列求和以后可以得到点数大概是2*N左右.
- 为了方便，我们在建一颗满的线段树时，通常会时利用满二叉树的性质，直接令节点i的左儿子为$i*2$,右儿子为$i*2+1$.

一个值得注意的问题

- 我们要注意，当利用上述性质性质建树时，要把存储节点的数组开到$4 * N$.

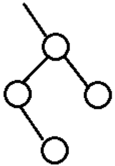
一个值得注意的问题

- 我们要注意，当利用上述性质性质建树时，要把存储节点的数组开到$4 * N$.
- 因为有可能出现下面这种情况:



一个值得注意的问题

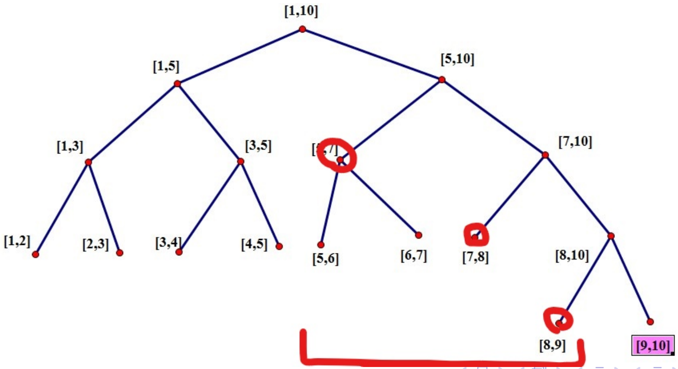
- 我们要注意，当利用上述性质性质建树时，要把存储节点的数组开到$4 * N$.
- 因为有可能出现下面这种情况:



- 此时编号最大的节点为$2^{(\log N)+2} - 3$.

重要性质2

- 一段连续的区间在线段树中可以被表示成不超过$O(\log N)$个节点.



作用

- 这意味着当我们需要查询一段区间时，只要把这些节点维护的信息合并就好了.

作用

- 这意味着当我们需要查询一段区间时，只要把这些节点维护的信息合并就好了.
- 如何找到这些节点？

作用

- 这意味着当我们需要查询一段区间时，只要把这些节点维护的信息合并就好了.
- 如何找到这些节点?
- 从根节点开始，对于每个节点分类讨论.

作用

- 这意味着当我们需要查询一段区间时，只要把这些节点维护的信息合并就好了.
- 如何找到这些节点?
- 从根节点开始，对于每个节点分类讨论.
- 如果这个节点所管辖的区间刚好被我们当前要找的区间包含，那么就说明我们找到了一个合法的节点.

作用

- 这意味着当我们需要查询一段区间时，只要把这些节点维护的信息合并就好了.
- 如何找到这些节点?
- 从根节点开始，对于每个节点分类讨论.
- 如果这个节点所管辖的区间刚好被我们当前要找的区间包含，那么就说明我们找到了一个合法的节点.
- 否则就观察我们要找的区间是否存在于左（右）儿子管辖的区间中，如果是的话就继续向下寻找.

复杂度证明

- 如果是向某个儿子单独递归显然不影响复杂度，因为树高是$O(\log N)$级别的.

复杂度证明

- 如果是向某个儿子单独递归显然不影响复杂度，因为树高是$O(\log N)$级别的.
- 考虑同时向左右儿子递归的情况。

复杂度证明

- 如果是向某个儿子单独递归显然不影响复杂度，因为树高是$O(\log N)$级别的.
- 考虑同时向左右儿子递归的情况。
- 这个时候原区间会被分裂为两个非常特殊的区间，它们都靠在线段树的某个区间的边界上.

复杂度证明

- 如果是向某个儿子单独递归显然不影响复杂度，因为树高是$O(\log N)$级别的.
- 考虑同时向左右儿子递归的情况。
- 这个时候原区间会被分裂为两个非常特殊的区间，它们都靠在线段树的某个区间的边界上.
- 我们发现这两个区间都会在$O(\log N)$的时间内被遍历完.

复杂度证明

- 如果是向某个儿子单独递归显然不影响复杂度，因为树高是$O(\log N)$级别的.
- 考虑同时向左右儿子递归的情况。
- 这个时候原区间会被分裂为两个非常特殊的区间，它们都靠在线段树的某个区间的边界上.
- 我们发现这两个区间都会在$O(\log N)$的时间内被遍历完.
- 因为在遍历这两个特殊区间时，如果出现左右同时向下递归的情况，其中有一个新状态会直接找到节点后退出.

复杂度证明

- 如果是向某个儿子单独递归显然不影响复杂度，因为树高是$O(\log N)$级别的.
- 考虑同时向左右儿子递归的情况。
- 这个时候原区间会被分裂为两个非常特殊的区间，它们都靠在线段树的某个区间的边界上.
- 我们发现这两个区间都会在$O(\log N)$的时间内被遍历完.
- 因为在遍历这两个特殊区间时，如果出现左右同时向下递归的情况，其中有一个新状态会直接找到节点后退出.
- 这样遍历的状态数得到了保证，最终找到的合法节点也不超过$O(\log N)$.

一种常见的问题形式

- 给出一个长度为N序列，Q次操作，每个操作要求支持修改某个数的权值，或统计某个区间中的权值和.

一种常见的问题形式

- 给出一个长度为N序列，Q次操作，每个操作要求支持修改某个数的权值，或统计某个区间中的权值和.
- $N, Q <= 100000.$

一种常见的问题形式

- 给出一个长度为N序列，Q次操作，每个操作要求支持修改某个数的权值，或统计某个区间中的权值和.
- $N, Q <= 100000$.

一种常见的问题形式

- 给出一个长度为N序列，Q次操作，每个操作要求支持修改某个数的权值，或统计某个区间中的权值和.
- $N, Q <= 100000$.

- 用线段树维护这个序列的权值和，每次询问就利用之前的性质合并标记上的信息，对于修改操作，我们发现修改某个点的权值只会影响维护它的那一条链。

一种常见的问题形式

- 给出一个长度为N序列，Q次操作，每个操作要求支持修改某个数的权值，或统计某个区间中的权值和.
- $N, Q <= 100000$.

- 用线段树维护这个序列的权值和，每次询问就利用之前的性质合并标记上的信息，对于修改操作，我们发现修改某个点的权值只会影响维护它的那一条链。
- 由于树高是$O(\log N)$级别的，所以暴力维护即可.

一种常见的问题形式

- 给出一个长度为N序列，Q次操作，每个操作要求支持修改某个数的权值，或统计某个区间中的权值和.
- $N, Q <= 100000$.

- 用线段树维护这个序列的权值和，每次询问就利用之前的性质合并标记上的信息，对于修改操作，我们发现修改某个点的权值只会影响维护它的那一条链。
- 由于树高是$O(\log N)$级别的，所以暴力维护即可.
- 时间复杂度$O(N + Q * \log N)$.

Lazy标记

- 给出一个长度为N序列，Q次操作，每个操作要求支持给某个区间的数一起加上k，或统计某个区间中的权值和.

Lazy标记

- 给出一个长度为N序列，Q次操作，每个操作要求支持给某个区间的数一起加上k，或统计某个区间中的权值和.
- $N, Q <= 100000$.

Lazy标记

- 给出一个长度为N序列，Q次操作，每个操作要求支持给某个区间的数一起加上k，或统计某个区间中的权值和.
- $N, Q <= 100000$.

Lazy标记

- 给出一个长度为N序列，Q次操作，每个操作要求支持给某个区间的数一起加上k，或统计某个区间中的权值和.
- $N, Q <= 100000$.

- 套用之前的方法可以解决询问，问题在于现在不能暴力维护所有被影响的点，因为区间加法最坏情况下会影响$O(N)$个节点.

Lazy标记

- 给出一个长度为N序列，Q次操作，每个操作要求支持给某个区间的数一起加上k，或统计某个区间中的权值和.
- $N, Q <= 100000$.

- 套用之前的方法可以解决询问，问题在于现在不能暴力维护所有被影响的点，因为区间加法最坏情况下会影响$O(N)$个节点.
- 不过我们可以使用Lazy标记来处理区间加法.

Lazy标记

- 根据之前的性质，我们知道某次区间加法影响的区间可以被表示为线段树上的$O(\log N)$个节点.

Lazy标记

- 根据之前的性质，我们知道某次区间加法影响的区间可以被表示为线段树上的$O(\log N)$个节点.
- 这次操作所影响的节点就在这些节点所代表的子树中.

Lazy标记

- 根据之前的性质，我们知道某次区间加法影响的区间可以被表示为线段树上的$O(\log N)$个节点.
- 这次操作所影响的节点就在这些节点所代表的子树中.
- 我们在这些节点上维护一个标记val，表示这颗子树所代表的区间被加了val.

Lazy标记

- 根据之前的性质，我们知道某次区间加法影响的区间可以被表示为线段树上的$O(\log N)$个节点.
- 这次操作所影响的节点就在这些节点所代表的子树中.
- 我们在这些节点上维护一个标记val，表示这颗子树所代表的区间被加了val.
- 这样在处理操作时通过节点信息维护标记信息，比如有size个点就给区间和加上$val * size$.

Lazy标记

- 根据之前的性质，我们知道某次区间加法影响的区间可以被表示为线段树上的$O(\log N)$个节点.
- 这次操作所影响的节点就在这些节点所代表的子树中.
- 我们在这些节点上维护一个标记val，表示这颗子树所代表的区间被加了val.
- 这样在处理操作时通过节点信息维护标记信息，比如有size个点就给区间和加上$val * size$.
- 所谓Lazy即是不访问就不处理，通过这种手段保证复杂度.

Lazy标记

- 根据之前的性质，我们知道某次操作只会遍历线段树上的$O(\log N)$个节点.

Lazy标记

- 根据之前的性质，我们知道某次操作只会遍历线段树上的$O(\log N)$个节点.
- 显然这个遍历点集如果包含a，则必然也包含a的所有祖先.

Lazy标记

- 根据之前的性质，我们知道某次操作只会遍历线段树上的$O(\log N)$个节点.
- 显然这个遍历点集如果包含a，则必然也包含a的所有祖先.
- 由于之前我们通过打标记来维护区间加法，线段树中有的节点上维护的信息是不正确的.

Lazy标记

- 根据之前的性质，我们知道某次操作只会遍历线段树上的$O(\log N)$个节点.
- 显然这个遍历点集如果包含a，则必然也包含a的所有祖先.
- 由于之前我们通过打标记来维护区间加法，线段树中有的节点上维护的信息是不正确的.
- 但我们只要把某个点的所有祖先上的标记都传下来就可以保证这个点上的信息是正确的.

Lazy标记

- 根据之前的性质，我们知道某次操作只会遍历线段树上的$O(\log N)$个节点.
- 显然这个遍历点集如果包含a，则必然也包含a的所有祖先.
- 由于之前我们通过打标记来维护区间加法，线段树中有的节点上维护的信息是不正确的.
- 但我们只要把某个点的所有祖先上的标记都传下来就可以保证这个点上的信息是正确的.
- 在遍历一个点之前处理掉它祖先上的标记即可.

Lazy标记

- 根据之前的性质，我们知道某次操作只会遍历线段树上的$O(\log N)$个节点.
- 显然这个遍历点集如果包含a，则必然也包含a的所有祖先.
- 由于之前我们通过打标记来维护区间加法，线段树中有的节点上维护的信息是不正确的.
- 但我们只要把某个点的所有祖先上的标记都传下来就可以保证这个点上的信息是正确的.
- 在遍历一个点之前处理掉它祖先上的标记即可.
- 时间复杂度$O(N + Q * \log N)$.

小结

- 线段树可以维护很多的信息，并支持区间修改、区间查询.

小结

- 线段树可以维护很多的信息，并支持区间修改、区间查询.
- 但有两个比较常见的前提：

小结

- 线段树可以维护很多的信息，并支持区间修改、区间查询.
- 但有两个比较常见的前提：

1:信息要支持快速合并，通常具有加法或乘法性质.

小结

- 线段树可以维护很多的信息，并支持区间修改、区间查询.
- 但有两个比较常见的前提：

1:信息要支持快速合并，通常具有加法或乘法性质.
2:如果要支持区间修改，那就要支持合并Lazy标记.

小结

- 线段树可以维护很多的信息，并支持区间修改、区间查询.
- 但有两个比较常见的前提：

  1:信息要支持快速合并，通常具有加法或乘法性质.
  2:如果要支持区间修改，那就要支持合并Lazy标记.

- 和、积便是两个用线段树来合并的很好的例子.

小结

- 线段树可以维护很多的信息，并支持区间修改、区间查询.
- 但有两个比较常见的前提：

  1:信息要支持快速合并，通常具有加法或乘法性质.

  2:如果要支持区间修改，那就要支持合并Lazy标记.

- 和、积便是两个用线段树来合并的很好的例子.
- 一些在节点上维护信息时会造成非常数级的空间占用的信息往往也是不能用线段树维护的.

权值线段树

- 我们之前提到的线段树都是以序列作为下标的.

权值线段树

- 我们之前提到的线段树都是以序列作为下标的.
- 但其实维护序列只是线段树的众多功能之一.

权值线段树

- 我们之前提到的线段树都是以序列作为下标的.
- 但其实维护序列只是线段树的众多功能之一.
- 我们可以以权值作为下标来实现一种维护集合的线段树.

权值线段树

- 我们之前提到的线段树都是以序列作为下标的.
- 但其实维护序列只是线段树的众多功能之一.
- 我们可以以权值作为下标来实现一种维护集合的线段树.
- 通常称之为权值线段树，这种线段树有着非常丰富的应用.

权值线段树

- 一颗满的线段树有$O(N)$个节点.

权值线段树

- 一颗满的线段树有$O(N)$个节点.
- 但如果要维护一个元素，只需要$O(h)$个节点.

权值线段树

- 一颗满的线段树有$O(N)$个节点.
- 但如果要维护一个元素，只需要$O(h)$个节点.
- 当线段树中元素个数很少时，我们不需要把整个线段树都建出来，而是只保存我们关心的点.

一种常见的问题形式

- 给出多个数集，支持在某个数集中插入一个元素或删除一个元素，支持查询某个数集中第K小的数.

一种常见的问题形式

- 给出多个数集，支持在某个数集中插入一个元素或删除一个元素，支持查询某个数集中第K小的数.

一种常见的问题形式

- 给出多个数集，支持在某个数集中插入一个元素或删除一个元素，支持查询某个数集中第K小的数.

一种常见的问题形式

- 给出多个数集，支持在某个数集中插入一个元素或删除一个元素，支持查询某个数集中第K小的数.
- 所有数集总大小$<= N <=$ 100000，权值范围-1e9到1e9.

一种常见的问题形式

- 给出多个数集，支持在某个数集中插入一个元素或删除一个元素，支持查询某个数集中第K小的数.
- 所有数集总大小$<= N <= 100000$，权值范围-1e9到1e9.

一种常见的问题形式

- 给出多个数集，支持在某个数集中插入一个元素或删除一个元素，支持查询某个数集中第K小的数.
- 所有数集总大小$<= N <= 100000$，权值范围-1e9到1e9.

一种比较暴力的方法

- 对于每个集合以权值为下标维护一棵线段树，每个节点上维护一个size表示这个区间内有几个数.

一种比较暴力的方法

- 对于每个集合以权值为下标维护一棵线段树，每个节点上维护一个size表示这个区间内有几个数.
- 插入、删除操作相当于修改一条链.

一种比较暴力的方法

- 对于每个集合以权值为下标维护一棵线段树，每个节点上维护一个size表示这个区间内有几个数.
- 插入、删除操作相当于修改一条链.
- 考虑如何实现查询第K小.

一种比较暴力的方法

- 对于每个集合以权值为下标维护一棵线段树，每个节点上维护一个size表示这个区间内有几个数.
- 插入、删除操作相当于修改一条链.
- 考虑如何实现查询第K小.
- 显然答案是单调的,我们二分一个答案key，然后通过线段树来判断key是否正确.

一种比较暴力的方法

- 对于每个集合以权值为下标维护一棵线段树，每个节点上维护一个size表示这个区间内有几个数.
- 插入、删除操作相当于修改一条链.
- 考虑如何实现查询第K小.
- 显然答案是单调的,我们二分一个答案key，然后通过线段树来判断key是否正确.
- 通过$size$数组在$O(h)$的时间内求出比key小的数有几个，相当于一次区间查询.

一种比较暴力的方法

- 对于每个集合以权值为下标维护一棵线段树，每个节点上维护一个size表示这个区间内有几个数.
- 插入、删除操作相当于修改一条链.
- 考虑如何实现查询第K小.
- 显然答案是单调的,我们二分一个答案key，然后通过线段树来判断key是否正确.
- 通过*size*数组在$O(h)$的时间内求出比key小的数有几个，相当于一次区间查询.
- 于是可以支持验证key是合法，以及偏大或是偏小.

一种比较暴力的方法

- 对于每个集合以权值为下标维护一棵线段树，每个节点上维护一个size表示这个区间内有几个数.
- 插入、删除操作相当于修改一条链.
- 考虑如何实现查询第K小.
- 显然答案是单调的,我们二分一个答案key，然后通过线段树来判断key是否正确.
- 通过$size$数组在$O(h)$的时间内求出比key小的数有几个，相当于一次区间查询.
- 于是可以支持验证key是合法，以及偏大或是偏小.
- 我们可以在$O(h^2)$的时间复杂度内得到结果.

一种更优秀的做法

- 是否存在更优秀的做法?

一种更优秀的做法

- 是否存在更优秀的做法?
- 我们从根节点开始搜索，如果左儿子的$size >= K$，那说明结果落在左儿子管辖的区间中.

一种更优秀的做法

- 是否存在更优秀的做法?
- 我们从根节点开始搜索，如果左儿子的$size >= K$，那说明结果落在左儿子管辖的区间中.
- 否则说明结果落在右儿子管辖的区间中.

一种更优秀的做法

- 是否存在更优秀的做法?
- 我们从根节点开始搜索，如果左儿子的$size >= K$，那说明结果落在左儿子管辖的区间中.
- 否则说明结果落在右儿子管辖的区间中.
- 我们可以在$O(h)$的时间复杂度内得到结果.

小结

- 这样做看上去得到了一种空间复杂度$O(N \log h)$的方法.

小结

- 这样做看上去得到了一种空间复杂度$O(N \log h)$的方法.
- 但由于在随着树中节点增多的同时，我们插入产生的新节点会之间减少.

小结

- 这样做看上去得到了一种空间复杂度$O(N\log h)$的方法.
- 但由于在随着树中节点增多的同时，我们插入产生的新节点会之间减少.
- 所以这个复杂度是不满的.

小结

- 这样做看上去得到了一种空间复杂度$O(N \log h)$的方法.
- 但由于在随着树中节点增多的同时，我们插入产生的新节点会之间减少.
- 所以这个复杂度是不满的.

小结

- 这样做看上去得到了一种空间复杂度$O(N\log h)$的方法.
- 但由于在随着树中节点增多的同时，我们插入产生的新节点会之间减少.
- 所以这个复杂度是不满的.

- 另外，这种做法启示我们只要能够把一个数集表达成线段树的形式，就可以支持快速查询第K小.

小结

- 这样做看上去得到了一种空间复杂度$O(N\log h)$的方法.
- 但由于在随着树中节点增多的同时，我们插入产生的新节点会之间减少.
- 所以这个复杂度是不满的.

- 另外，这种做法启示我们只要能够把一个数集表达成线段树的形式，就可以支持快速查询第K小.
- 通过一次二分还可以把第K小问题转换成查询数集中存在几个小于key的元素.

Outline

1. 概念
   - 数据结构
2. 线段树
   - 简介
   - 如何维护信息
   - 作为数据结构的线段树
3. 应用技巧
   - 一些简单的问题
   - 一些复杂的问题
4. 结语

Bin Operation

- 给出一个01序列，要求支持以下操作:

Bin Operation

- 给出一个01序列，要求支持以下操作:

Bin Operation

- 给出一个01序列，要求支持以下操作：
- 求出某个区间内最靠左/右的1的位置.

Bin Operation

- 给出一个01序列，要求支持以下操作:

- 求出某个区间内最靠左/右的1的位置.
- 求出一段区间内最长的连续的1的长度.

Bin Operation

- 给出一个01序列，要求支持以下操作:

- 求出某个区间内最靠左/右的1的位置.
- 求出一段区间内最长的连续的1的长度.
- 将某个区间内的数全部改为0或1.

Bin Operation

- 给出一个01序列，要求支持以下操作:

- 求出某个区间内最靠左/右的1的位置.
- 求出一段区间内最长的连续的1的长度.
- 将某个区间内的数全部改为0或1.
- 将某个区间内的0全部改为1，1全部改为0.

Bin Operation

- 给出一个01序列，要求支持以下操作:

- 求出某个区间内最靠左/右的1的位置.
- 求出一段区间内最长的连续的1的长度.
- 将某个区间内的数全部改为0或1.
- 将某个区间内的0全部改为1，1全部改为0.

Bin Operation

- 给出一个01序列，要求支持以下操作:

- 求出某个区间内最靠左/右的1的位置.
- 求出一段区间内最长的连续的1的长度.
- 将某个区间内的数全部改为0或1.
- 将某个区间内的0全部改为1，1全部改为0.

- 序列长度小于等于100000.

Bin Operation

- 显然前两个问题用线段树维护标记是可以$O(1)$合并的.

Bin Operation

- 显然前两个问题用线段树维护标记是可以$O(1)$合并的.
- 第三个操作直接Lazy.

Bin Operation

- 显然前两个问题用线段树维护标记是可以$O(1)$合并的.
- 第三个操作直接Lazy.
- 对于翻转操作，我们只要维护下0的情况就可以支持Lazy了.

History

- 给出一个序列，要求支持以下操作:

History

- 给出一个序列，要求支持以下操作:

History

- 给出一个序列，要求支持以下操作:
- 把某个区间的数都加上某个数.

History

- 给出一个序列，要求支持以下操作:
- 把某个区间的数都加上某个数.
- 把某个区间的数都改为某个数.

History

- 给出一个序列，要求支持以下操作:

- 把某个区间的数都加上某个数.
- 把某个区间的数都改为某个数.
- 求某个区间的最大值.

History

- 给出一个序列，要求支持以下操作:

- 把某个区间的数都加上某个数.
- 把某个区间的数都改为某个数.
- 求某个区间的最大值.
- 求某个区间的历史上出现过的最大值.

History

- 给出一个序列，要求支持以下操作：

- 把某个区间的数都加上某个数.
- 把某个区间的数都改为某个数.
- 求某个区间的最大值.
- 求某个区间的历史上出现过的最大值.

History

- 给出一个序列，要求支持以下操作:

- 把某个区间的数都加上某个数.
- 把某个区间的数都改为某个数.
- 求某个区间的最大值.
- 求某个区间的历史上出现过的最大值.

- 序列长度小于等于100000.

History

- 只要对于每种标记都多维护一个历史上的情况即可.

History

- 只要对于每种标记都多维护一个历史上的情况即可.
- 注意Lazy标记的下传顺序.

History

- 只要对于每种标记都多维护一个历史上的情况即可.
- 注意Lazy标记的下传顺序.

Abs Query

- 给出一个长度为N的序列（可正可负），要求支持以下操作:

Abs Query

- 给出一个长度为N的序列（可正可负），要求支持以下操作:

Abs Query

- 给出一个长度为N的序列（可正可负），要求支持以下操作:
- 把某个区间的数全部加上某个正整数.

Abs Query

- 给出一个长度为N的序列（可正可负），要求支持以下操作:
- 把某个区间的数全部加上某个正整数.
- 询问某个区间中的数的绝对值之和.

Abs Query

- 给出一个长度为N的序列（可正可负），要求支持以下操作:
- 把某个区间的数全部加上某个正整数.
- 询问某个区间中的数的绝对值之和.

Abs Query

- 给出一个长度为N的序列（可正可负），要求支持以下操作:

- 把某个区间的数全部加上某个正整数.
- 询问某个区间中的数的绝对值之和.

- 序列长度小于等于100000.

Abs Query

- 由于某次操作可能会改变区间中某些数的正负性，所以没有办法简单地Lazy维护.

Abs Query

- 由于某次操作可能会改变区间中某些数的正负性，所以没有办法简单地Lazy维护.
- 不过由于这种情况最多只会出现N次，我们可以在线段树上维护下区间中最大的负数.

Abs Query

- 由于某次操作可能会改变区间中某些数的正负性，所以没有办法简单地Lazy维护.
- 不过由于这种情况最多只会出现N次，我们可以在线段树上维护下区间中最大的负数.
- 当我们发现这个最大的负数被加以后会改变符号时，说明这个区间无法Lazy.

Abs Query

- 由于某次操作可能会改变区间中某些数的正负性，所以没有办法简单地Lazy维护.
- 不过由于这种情况最多只会出现N次，我们可以在线段树上维护下区间中最大的负数.
- 当我们发现这个最大的负数被加以后会改变符号时，说明这个区间无法Lazy.
- 此时暴力下传标记，直到遇到不会发生符号改 变的区间再Lazy.

Abs Query

- 这样操作复杂度有保证的原因是每次暴力下传就意味着某个负数会变为正数.

Abs Query

- 这样操作复杂度有保证的原因是每次暴力下传就意味着某个负数会变为正数.
- 而且由于加数的性质，它不会再变成负数了.

Abs Query

- 这样操作复杂度有保证的原因是每次暴力下传就意味着某个负数会变为正数.
- 而且由于加数的性质，它不会再变成负数了.

Abs Query

- 这样操作复杂度有保证的原因是每次暴力下传就意味着某个负数会变为正数.
- 而且由于加数的性质，它不会再变成负数了.

- 其实就是在利用某些变化的有限性.

Abs Query

- 这样操作复杂度有保证的原因是每次暴力下传就意味着某个负数会变为正数.
- 而且由于加数的性质，它不会再变成负数了.

- 其实就是在利用某些变化的有限性.
- 类似的问题还有给一个区间中所有的数开根号之类.

Other

- 其实还有很多想和你们分享的题.

Other

- 其实还有很多想和你们分享的题.
- 这里写不下了.

Other

- 其实还有很多想和你们分享的题.
- 这里写不下了.

PA2011 Journeys

- 给出一张n个点的图，每次给出一组a, b, c, d，表示对于任何一组x, y满足$a <= x <= b, c <= y <= d$，都建立一条边权为1的有向边.

PA2011 Journeys

- 给出一张n个点的图，每次给出一组a, b, c, d，表示对于任何一组x, y满足$a <= x <= b, c <= y <= d$，都建立一条边权为1的有向边.
- 求出点S到所有其他点的最短路（保证最短路存在）.

PA2011 Journeys

- 给出一张n个点的图，每次给出一组a, b, c, d，表示对于任何一组x, y满足$a <= x <= b, c <= y <= d$，都建立一条边权为1的有向边.
- 求出点S到所有其他点的最短路（保证最短路存在）.

PA2011 Journeys

- 给出一张n个点的图，每次给出一组a, b, c, d，表示对于任何一组x, y满足$a <= x <= b, c <= y <= d$，都建立一条边权为1的有向边.
- 求出点S到所有其他点的最短路（保证最短路存在）.

- n和四元组数量都保证$<= 100000$.

PA2011 Journeys

- 直接建图显然不太现实.

PA2011 Journeys

- 直接建图显然不太现实.
- 我们考虑根据每组的a,b来把c,d挂在线段树上.

PA2011 Journeys

- 直接建图显然不太现实.
- 我们考虑根据每组的a,b来把c,d挂在线段树上.
- 由于区间连续，一条边可以被拆成$O(\log N)$个线段树上的标记.

PA2011 Journeys

- 直接建图显然不太现实.
- 我们考虑根据每组的a,b来把c,d挂在线段树上.
- 由于区间连续，一条边可以被拆成$O(\log N)$个线段树上的标记.
- 尽管这些标记不支持合并，不过数量不多，我们直接暴力挂在线段树上.

PA2011 Journeys

- 我们考虑从起点开始Bfs，对于每个点我们在线段树上寻找那些从这个点出发的边.

PA2011 Journeys

- 我们考虑从起点开始Bfs，对于每个点我们在线段树上寻找那些从这个点出发的边.
- 这样问题转化为找到一个区间中还没有加入Bfs队列的点并将其入队.

PA2011 Journeys

- 我们考虑从起点开始Bfs，对于每个点我们在线段树上寻找那些从这个点出发的边.
- 这样问题转化为找到一个区间中还没有加入Bfs队列的点并将其入队.
- 可以再用一个线段树来找到这些点.

PA2011 Journeys

- 看上去似乎很暴力.

PA2011 Journeys

- 看上去似乎很暴力.
- 有一个很显然的性质是一条边只会被用一次.

PA2011 Journeys

- 看上去似乎很暴力.
- 有一个很显然的性质是一条边只会被用一次.
- 我们拓展过一条边之后就将其在维护边集的线段树节点上删除.

PA2011 Journeys

- 看上去似乎很暴力.
- 有一个很显然的性质是一条边只会被用一次.
- 我们拓展过一条边之后就将其在维护边集的线段树节点上删除.
- 很显然复杂度可以得到保证.

PA2011 Journeys

- 看上去似乎很暴力.
- 有一个很显然的性质是一条边只会被用一次.
- 我们拓展过一条边之后就将其在维护边集的线段树节点上删除.
- 很显然复杂度可以得到保证.
- 时空复杂度$O(N \log N + M \log N \log N)$.

PA2011 Journeys

- 看上去似乎很暴力.
- 有一个很显然的性质是一条边只会被用一次.
- 我们拓展过一条边之后就将其在维护边集的线段树节点上删除.
- 很显然复杂度可以得到保证.
- 时空复杂度$O(N\log N + M\log N\log N)$.
- 其实可以用并查集来找某个区间中最靠左存在的点，复杂度能再降低一个级别.

PA2011 Journeys

- 看上去似乎很暴力.
- 有一个很显然的性质是一条边只会被用一次.
- 我们拓展过一条边之后就将其在维护边集的线段树节点上删除.
- 很显然复杂度可以得到保证.
- 时空复杂度$O(N\log N + M\log N\log N)$.
- 其实可以用并查集来找某个区间中最靠左存在的点，复杂度能再降低一个级别.
- 这种直接把一些难以处理的操作挂在线段树上的思想也很常用.

BZOJ 4025

- 给出一个有N个点的图，N条边每条边只有在[L,R]的时刻才存在.

BZOJ 4025

- 给出一个有N个点的图，N条边每条边只有在[L,R]的时刻才存在.
- 现在询问每个时刻该图是否是二分图.

BZOJ 4025

- 给出一个有N个点的图，N条边每条边只有在[L,R]的时刻才存在.
- 现在询问每个时刻该图是否是二分图.
- N,M,L,R$<=$ 100000.

BZOJ 4025

- 类似地把边挂在线段树上.

BZOJ 4025

- 类似地把边挂在线段树上.
- 每次从根节点走下来时维护并查集即可.

BZOJ 4025

- 类似地把边挂在线段树上.
- 每次从根节点走下来时维护并查集即可.
- 此题还有一个使用LCT维护时间生成树的解法.

BZOJ 4025

- End or Start!