浅谈维护多维数组的方法在数据结构题中的应用

梁泽宇

合肥一中

April 20, 2014

引入：可修改区间第K小问题

给出一个正整数序列$A[1 \mathinner{..} n]$，两种操作：

引入：可修改区间第K小问题

给出一个正整数序列$A[1\,..\,n]$，两种操作：

将$A[i]$修改为x

引入：可修改区间第K小问题

给出一个正整数序列$A[1\,..\,n]$，两种操作：

- 将$A[i]$修改为x
- 询问$A[l\,..\,r]$中的第K小值

引入：可修改区间第K小问题

给出一个正整数序列$A[1 \mathinner{..} n]$，两种操作：

- 将$A[i]$修改为x
- 询问$A[l \mathinner{..} r]$中的第K小值

设$A[i]$值的范围为$[1 \mathinner{..} W]$，操作总数为q

要求$O((n+q)\log n \log W)$的算法。

所谓的常见解法

系列题解之一：权值线段树套平衡树

所谓的常见解法

系列题解之一：权值线段树套平衡树

系列题解之二：权值树状数组套平衡树

所谓的常见解法

系列题解之一：权值线段树套平衡树

系列题解之二：权值树状数组套平衡树

系列题解之三：重量平衡树套线段树

所谓的常见解法

系列题解之一：权值线段树套平衡树

系列题解之二：权值树状数组套平衡树

系列题解之三：重量平衡树套线段树

系列题解之四：权值块状数组套平衡树

所谓的常见解法

系列题解之一：权值线段树套平衡树

系列题解之二：权值树状数组套平衡树

系列题解之三：重量平衡树套线段树

系列题解之四：权值块状数组套平衡树

都需要嵌套数据结构，代码量太大，无法忍受。

所谓的常见解法

系列题解之一：权值线段树套平衡树

系列题解之二：权值树状数组套平衡树

系列题解之三：重量平衡树套线段树

系列题解之四：权值块状数组套平衡树

都需要嵌套数据结构，代码量太大，无法忍受。

要想新方法！！

权值二维数组

建立$n * W$的数组

权值二维数组

建立$n * W$的数组

若$A[i] = j$，则该数组$[i][j]$元素为1

权值二维数组

建立$n*W$的数组

若$A[i]=j$，则该数组$[i][j]$元素为1

其它元素均为0

权值二维数组

建立$n * W$的数组

若$A[i] = j$，则该数组$[i][j]$元素为1

其它元素均为0

如$n = 5, W = 8$，序列$A[] = (4, 5, 7, 1, 3)$

权值二维数组

建立$n*W$的数组

若$A[i]=j$，则该数组$[i][j]$元素为1

其它元素均为0

如$n=5, W=8$，序列$A[]=(4,5,7,1,3)$

0	0	0	1	0	0	0	0
0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0
1	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0

权值二维数组

建立$n * W$的数组

若$A[i] = j$，则该数组$[i][j]$元素为1

其它元素均为0

如$n = 5, W = 8$，序列$A[] = (4, 5, 7, 1, 3)$

0	0	0	1	0	0	0	0
0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0
1	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0

对于任意$1 \le W_r \le W$，该数组中$[l .. r][1 .. W_r]$内元素的和

权值二维数组

建立$n * W$的数组

若$A[i] = j$，则该数组$[i][j]$元素为1

其它元素均为0

如$n = 5, W = 8$，序列$A[] = (4, 5, 7, 1, 3)$

0	0	0	1	0	0	0	0
0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0
1	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0

对于任意$1 \le W_r \le W$，该数组中$[l .. r][1 .. W_r]$内元素的和

就是$A[l .. r]$中值不超过W_r的元素个数。

权值二维数组

建立$n*W$的数组

若$A[i]=j$，则该数组$[i][j]$元素为1

其它元素均为0

如$n=5, W=8$，序列$A[]=(4,5,7,1,3)$

0	0	0	1	0	0	0	0
0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0
1	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0

权值二维数组

建立$n * W$的数组

若$A[i] = j$，则该数组$[i][j]$元素为1

其它元素均为0

如$n = 5, W = 8$，序列$A[] = (4, 5, 7, 1, 3)$

0	0	0	1	0	0	0	0
0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0
1	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0

用二维树状数组维护

可以通过二分将本题的询问操作转化为求二维区间和

用二维树状数组维护

可以通过二分将本题的询问操作转化为求二维区间和

用二维树状数组维护

用二维树状数组维护

- 可以通过二分将本题的询问操作转化为求二维区间和
- 用二维树状数组维护
- 修改操作直接在二维树状数组里改，时间复杂度$O(\log n \log W)$

用二维树状数组维护

可以通过二分将本题的询问操作转化为求二维区间和

用二维树状数组维护

修改操作直接在二维树状数组里改，时间复杂度$O(\log n \log W)$

每次询问，需要二分$O(\log W)$次

用二维树状数组维护

可以通过二分将本题的询问操作转化为求二维区间和

用二维树状数组维护

修改操作直接在二维树状数组里改，时间复杂度$O(\log n \log W)$

每次询问，需要二分$O(\log W)$次

每次二分求区间和需要涉及二维树状数组中的$O(\log n \log W)$个结点

用二维树状数组维护

可以通过二分将本题的询问操作转化为求二维区间和

用二维树状数组维护

修改操作直接在二维树状数组里改，时间复杂度$O(\log n \log W)$

每次询问，需要二分$O(\log W)$次

每次二分求区间和需要涉及二维树状数组中的$O(\log n \log W)$个结点

总时间复杂度$O(\log n \log^2 W)$......

用二维树状数组维护

- 可以通过二分将本题的询问操作转化为求二维区间和
- 用二维树状数组维护
- 修改操作直接在二维树状数组里改，时间复杂度$O(\log n \log W)$
- 每次询问，需要二分$O(\log W)$次
- 每次二分求区间和需要涉及二维树状数组中的$\cancel{O(\log n \log W)}$个结点
- 总时间复杂度$\cancel{O(\log n \log^2 W)}$......
- 按位从左到右二分W_r，每次在第二维（$[1..W]$这一维）相对于上一次只增加一个结点

用二维树状数组维护

- 可以通过二分将本题的询问操作转化为求二维区间和
- 用二维树状数组维护
- 修改操作直接在二维树状数组里改，时间复杂度$O(\log n \log W)$
- 每次询问，需要二分$O(\log W)$次
- 每次二分求区间和需要涉及二维树状数组中的$\cancel{O(\log n \log W)}$个结点
- 总时间复杂度$\cancel{O(\log n \log^2 W)}$......
- 按位从左到右二分W_r，每次在第二维（$[1..W]$这一维）相对于上一次只增加一个结点
- 每次二分只增加$O(\log W)$个结点，询问时间复杂度$O(\log n \log W)$......

如何存储二维数组

这个二维树状数组结点太多，以至于无法完全存下来......

如何存储二维数组

这个二维树状数组结点太多，以至于无法完全存下来……

注意到一次修改操作只会修改$O(\log n \log W)$个结点

如何存储二维数组

这个二维树状数组结点太多，以至于无法完全存下来......

注意到一次修改操作只会修改$O(\log n \log W)$个结点

也就是q次修改操作之后，树状数组中值不为0的结点只有$O((n+q)\log n \log W)$个。

如何存储二维数组

这个二维树状数组结点太多，以至于无法完全存下来......

注意到一次修改操作只会修改$O(\log n \log W)$个结点

也就是q次修改操作之后，树状数组中值不为0的结点只有$O((n+q)\log n \log W)$个。

用hash表或STL set存储这些不为0的结点即可。

分层块状数组

除了用二维树状数组外，还可以用分层块状数组维护区间和（同样只存储值不为0的结点）

分层块状数组

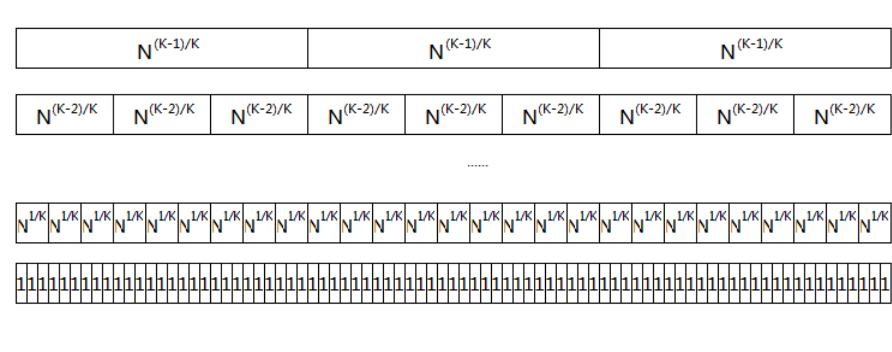
除了用二维树状数组外，还可以用分层块状数组维护区间和（同样只存储值不为0的结点）

一维情况，K层的块状数组：

分层块状数组

除了用二维树状数组外，还可以用分层块状数组维护区间和（同样只存储值不为0的结点）

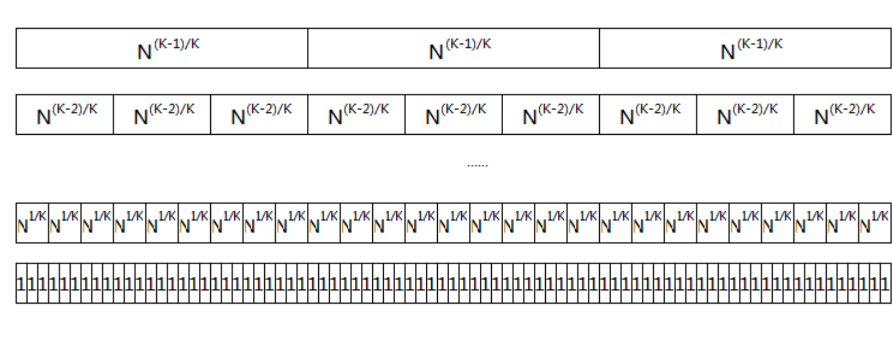
一维情况，*K*层的块状数组：



分层块状数组

除了用二维树状数组外，还可以用分层块状数组维护区间和（同样只存储值不为0的结点）

一维情况，K层的块状数组：

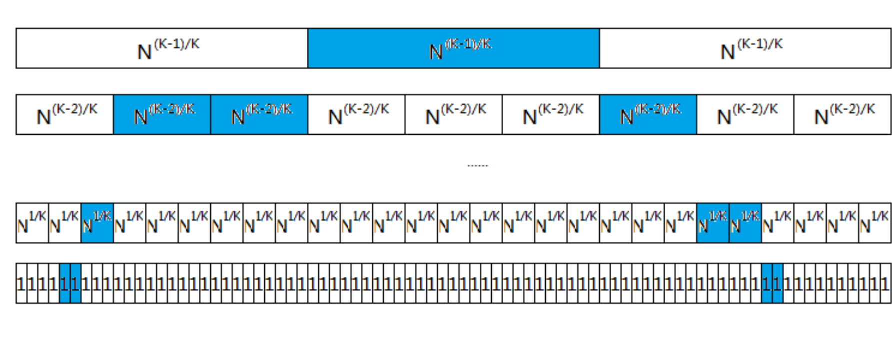


每个区间可以拆成每层各$O(n^{\frac{1}{K}})$个区间的并，共$O(Kn^{\frac{1}{K}})$个。

分层块状数组

除了用二维树状数组外，还可以用分层块状数组维护区间和（同样只存储值不为0的结点）

一维情况，K层的块状数组：

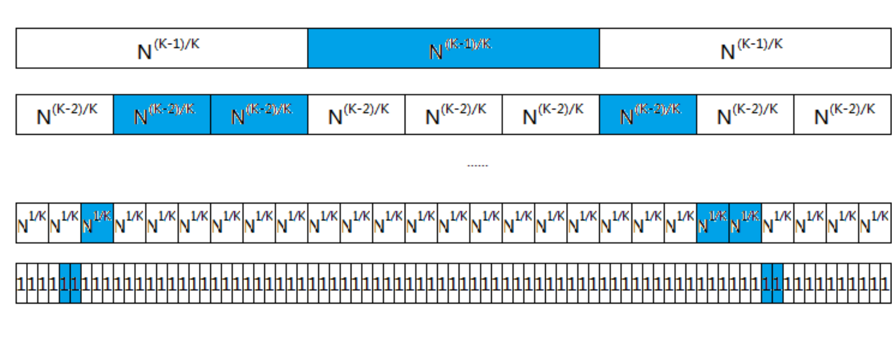


每个区间可以拆成每层各$O(n^{\frac{1}{K}})$个区间的并，共$O(Kn^{\frac{1}{K}})$个。

分层块状数组

除了用二维树状数组外，还可以用分层块状数组维护区间和（同样只存储值不为0的结点）

一维情况，K层的块状数组：



每个区间可以拆成每层各$O(n^{\frac{1}{K}})$个区间的并，共$O(Kn^{\frac{1}{K}})$个。

二维及以上的分层块状数组

对每一维分别建立一维分层块状数组

二维及以上的分层块状数组

对每一维分别建立一维分层块状数组

一个区间在每一维上分别拆成$O(Kn^{\frac{1}{K}})$个区间的并

二维及以上的分层块状数组

对每一维分别建立一维分层块状数组

一个区间在每一维上分别拆成$O(Kn^{\frac{1}{K}})$个区间的并

设有D维，共$O(K^D n^{\frac{D}{K}})$个

二维及以上的分层块状数组

对每一维分别建立一维分层块状数组

一个区间在每一维上分别拆成$O(Kn^{\frac{1}{K}})$个区间的并

设有D维，共$O(K^D n^{\frac{D}{K}})$个

每个元素关联K^D个块。

二维及以上的分层块状数组

对每一维分别建立一维分层块状数组

一个区间在每一维上分别拆成$O(Kn^{\frac{1}{K}})$个区间的并

设有D维，共$O(K^D n^{\frac{D}{K}})$个

每个元素关联K^D个块。

为了保证修改和询问的复杂度平衡，一维数组K取$3\sim4$，二维数组K取$5\sim6$效果最好。

动态序列的多维数组维护方法

有的题目需要维护一个动态（有插入删除操作的）序列

动态序列的多维数组维护方法

有的题目需要维护一个动态（有插入删除操作的）序列

使用动态标号法，每次插入均摊修改$O(\log n)$个标号

动态序列的多维数组维护方法

有的题目需要维护一个动态（有插入删除操作的）序列

使用动态标号法，每次插入均摊修改$O(\log n)$个标号

将标号作为下标，建立权值二维数组

动态序列的多维数组维护方法

有的题目需要维护一个动态（有插入删除操作的）序列

使用动态标号法，每次插入均摊修改$O(\log n)$个标号

将标号作为下标，建立权值二维数组

标号这一维用线段树，值这一维用树状数组维护区间和

动态序列的多维数组维护方法

有的题目需要维护一个动态（有插入删除操作的）序列

使用动态标号法，每次插入均摊修改$O(\log n)$个标号

将标号作为下标，建立权值二维数组

标号这一维用线段树，值这一维用树状数组维护区间和

复杂度不变，代码量大幅减小......

动态序列的多维数组维护方法

有的题目需要维护一个动态（有插入删除操作的）序列

使用动态标号法，每次插入均摊修改$O(\log n)$个标号

将标号作为下标，建立权值二维数组

标号这一维用线段树，值这一维用树状数组维护区间和

复杂度不变，代码量大幅减小......

（嵌套数据结构在某种意义下已成为时代的眼泪......）

树的处理

借助DFS序，这种方法同样可以应用于树上

树的处理

借助DFS序，这种方法同样可以应用于树上

一般的DFS序可以处理子树，但不能处理路径

树的处理

借助DFS序，这种方法同样可以应用于树上

一般的DFS序可以处理子树，但不能处理路径

定义 (欧拉DFS序)

*DFS*遍历整棵树，每个结点在入栈的时候记录一次，出栈的时候再记录一次，得到的序列称为该树的**欧拉DFS序**，又叫**括号序**。设$in[i]$为结点i的入栈记录在序列中出现的位置（又叫**左括号位置**），$out[i]$为结点i的出栈记录在序列中出现的位置（又叫**右括号位置**）。

树的处理

关于欧拉DFS序有两个很重要的性质。

树的处理

关于欧拉DFS序有两个很重要的性质。

性质 (1)

对于一棵树上任意两个无前后代关系的结点u和v，
若$out[u] <= in[v]$，则该树的欧拉DFS序中的$[out[u] .. in[v]]$区间内所有**出现且仅出现一次**的结点以及$\mathrm{LCA}(u, v)$，就是树上路径$u \to v$中的所有结点。

性质 (2)

对于一棵树上的两个结点u和v，若u是v的祖先，则该树的欧拉DFS序中的$[in[u] .. in[v]]$区间内所有**出现且仅出现一次**的结点，就是树上路径$u \to v$中的所有结点。

树的处理

根据上述性质，树上的一条路径对应其欧拉DFS序中的一个区间

树的处理

根据上述性质，树上的一条路径对应其欧拉DFS序中的一个区间

路径上的结点为这个区间内所有出现且仅出现一次的结点

树的处理

根据上述性质，树上的一条路径对应其欧拉DFS序中的一个区间

路径上的结点为这个区间内所有出现且仅出现一次的结点

对于LCA可以特判

树的处理

根据上述性质，树上的一条路径对应其欧拉DFS序中的一个区间

路径上的结点为这个区间内所有出现且仅出现一次的结点

对于LCA可以特判

如果操作可反，可以使用**反操作抵消**的方法，即一个结点第二次出现时就抵消它第一次出现的效果值

树的处理

根据上述性质，树上的一条路径对应其欧拉DFS序中的一个区间

路径上的结点为这个区间内所有出现且仅出现一次的结点

对于LCA可以特判

如果操作可反，可以使用**反操作抵消**的方法，即一个结点第二次出现时就抵消它第一次出现的效果值

可以利用性质(2)，将路径$u \to v$拆成

$$[1, in[u]] + [1, in[v]] - [1, in[LCA(u, v)]] - [1, in[LCA(u, v).pr]]$$

其中$x.pr$表示结点x的父结点。

例1：糖果公园(WC2013)

给出一棵有n个结点的树，每个结点i上有一个权值$V[i]$，另外给出一个序列$W[1..n]$。两种操作

例1：糖果公园(WC2013)

给出一棵有n个结点的树，每个结点i上有一个权值$V[i]$，另外给出一个序列$W[1..n]$。两种操作

- 修改一个点的权值；

例1：糖果公园(WC2013)

给出一棵有n个结点的树，每个结点i上有一个权值$V[i]$，另外给出一个序列$W[1..n]$。两种操作

修改一个点的权值；

询问树上的路径$u \to v$上所有结点的效果值之和，对于该路径上的一个结点i，如果它的权值是在该路径上第k次出现的（即前面已有$k-1$个结点的权值等于它），则它的效果值为$V[i]W[k]$。

例1：糖果公园(WC2013)

给出一棵有n个结点的树，每个结点i上有一个权值$V[i]$，另外给出一个序列$W[1..n]$。两种操作

- 修改一个点的权值；
- 询问树上的路径$u \to v$上所有结点的效果值之和，对于该路径上的一个结点i，如果它的权值是在该路径上第k次出现的（即前面已有$k-1$个结点的权值等于它），则它的效果值为$V[i]W[k]$。

$1 \leq n$、询问次数$Q \leq 10^5$，时限30s。

例1：糖果公园(WC2013)

先求出欧拉DFS序

例1：糖果公园(WC2013)

先求出欧拉DFS序

建立数组$C[1..2n][1..2n]$，$C[i][j](i \le j)$表示欧拉**DFS**序中位置i的存在对位置j的效果值的影响（增量）

例1：糖果公园(WC2013)

先求出欧拉DFS序

建立数组$C[1..2n][1..2n]$，$C[i][j](i \le j)$表示**欧拉DFS序中位置i的存在对位置j的效果值的影响（增量）**

设$V'[i]$为欧拉DFS序位置i对应结点的权值，当且仅当$i \le j$且$V'[i] = V'[j]$时，$C[i][j]$才有意义。

例1：糖果公园(WC2013)

先求出欧拉DFS序

建立数组$C[1..2n][1..2n]$，$C[i][j](i \le j)$表示欧拉**DFS**序中位置i的存在对位置j的效果值的影响（增量）

设$V'[i]$为欧拉DFS序位置i对应结点的权值，当且仅当$i \le j$且$V'[i] = V'[j]$时，$C[i][j]$才有意义。

对于树中出现次数超过$n^{\frac{1}{3}}$的权值，询问时特殊处理

例1：糖果公园(WC2013)

先求出欧拉DFS序

建立数组$C[1..2n][1..2n]$，$C[i][j](i \le j)$表示**欧拉DFS序中位置i的存在对位置j的效果值的影响（增量）**

设$V'[i]$为欧拉DFS序位置i对应结点的权值，当且仅当$i \le j$且$V'[i] = V'[j]$时，$C[i][j]$才有意义。

对于树中出现次数超过$n^{\frac{1}{3}}$的权值，询问时特殊处理

其它权值在C中算出对应值，有以下几种情况（假设$W[0] = 0$）：

例1：糖果公园(WC2013)

若位置i和j对应相同结点，则$C[i][j]=-2V'[i]W[1]$，因为i和j的第一次出现的效果值同时被抵消；

例1：糖果公园(WC2013)

若位置i和j对应相同结点，则$C[i][j] = -2V'[i]W[1]$，因为i和j的第一次出现的效果值同时被抵消；

若在区间$[i+1..j]$中存在某位置与j对应相同结点（即位置j对应结点在$[i+1..j]$中出现两次），则$C[i][j] = 0$，因为此时位置j的效果值已被完全抵消，为0，所以位置i的出现并不会改变其效果值；

例1：糖果公园(WC2013)

若位置i和j对应相同结点，则$C[i][j]=-2V'[i]W[1]$，因为i和j的第一次出现的效果值同时被抵消；

若在区间$[i+1..j]$中存在某位置与j对应相同结点（即位置j对应结点在$[i+1..j]$中出现两次），则$C[i][j]=0$，因为此时位置j的效果值已被完全抵消，为0，所以位置i的出现并不会改变其效果值；

若在区间$[i..j]$中不存在与位置j对应相同结点的位置（即位置j对应结点在$[i..j]$中出现一次），设在区间$[i..j]$中有K个权值为$V'[i]$的结点出现且仅出现一次，则当i在区间$[i..j]$中出现一次时，$C[i][j]=V'[i](W[K]-W[K-1])$，即加上位置$i$对应结点的影响，当$i$在区间$[i..j]$中出现两次时，$C[i][j]=V'[i](W[K-1]-W[K])$，即抵消位置$i$对应结点的影响。

例1：糖果公园(WC2013)

- 若位置i和j对应相同结点，则$C[i][j] = -2V'[i]W[1]$，因为i和j的第一次出现的效果值同时被抵消；
- 若在区间$[i+1..j]$中存在某位置与j对应相同结点（即位置j对应结点在$[i+1..j]$中出现两次），则$C[i][j] = 0$，因为此时位置j的效果值已被完全抵消，为0，所以位置i的出现并不会改变其效果值；
- 若在区间$[i..j]$中不存在与位置j对应相同结点的位置（即位置j对应结点在$[i..j]$中出现一次），设在区间$[i..j]$中有K个权值为$V'[i]$的结点出现且仅出现一次，则当i在区间$[i..j]$中出现一次时，$C[i][j] = V'[i](W[K] - W[K-1])$，即加上位置$i$对应结点的影响，当$i$在区间$[i..j]$中出现两次时，$C[i][j] = V'[i](W[K-1] - W[K])$，即抵消位置$i$对应结点的影响。

例1：糖果公园(WC2013)

这样就将询问操作转化成了区间和，只需求出$C[l..r][l..r]$（或$C[1..r][l..2n]$）的区间和，即为询问的答案。

例1：糖果公园(WC2013)

这样就将询问操作转化成了区间和，只需求出$C[l..r][l..r]$（或$C[1..r][l..2n]$）的区间和，即为询问的答案。

对于修改操作，如果涉及的权值出现次数不超过$n^{\frac{1}{3}}$，则C中所有该权值对应的元素全部暴力重算，一共$O(n^{\frac{2}{3}})$个元素

例1：糖果公园(WC2013)

这样就将询问操作转化成了区间和，只需求出$C[l..r][l..r]$（或$C[1..r][l..2n]$）的区间和，即为询问的答案。

对于修改操作，如果涉及的权值出现次数不超过$n^{\frac{1}{3}}$，则C中所有该权值对应的元素全部暴力重算，一共$O(n^{\frac{2}{3}})$个元素

如果出现次数超过$n^{\frac{1}{3}}$，特殊处理即可。

例1：糖果公园(WC2013)

这样就将询问操作转化成了区间和，只需求出$C[l..r][l..r]$（或$C[1..r][l..2n]$）的区间和，即为询问的答案。

对于修改操作，如果涉及的权值出现次数不超过$n^{\frac{1}{3}}$，则C中所有该权值对应的元素全部暴力重算，一共$O(n^{\frac{2}{3}})$个元素

如果出现次数超过$n^{\frac{1}{3}}$，特殊处理即可。

时间复杂度：如果使用二维树状数组维护数组C，则一次操作时间复杂度为$O(n^{\frac{2}{3}}\log^2 n)$

例1：糖果公园(WC2013)

这样就将询问操作转化成了区间和，只需求出$C[l..r][l..r]$（或$C[1..r][l..2n]$）的区间和，即为询问的答案。

对于修改操作，如果涉及的权值出现次数不超过$n^{\frac{1}{3}}$，则C中所有该权值对应的元素全部暴力重算，一共$O(n^{\frac{2}{3}})$个元素

如果出现次数超过$n^{\frac{1}{3}}$，特殊处理即可。

时间复杂度：如果使用二维树状数组维护数组C，则一次操作时间复杂度为$O(n^{\frac{2}{3}}\log^2 n)$

如果使用分层块状数组，假设指数取3（本题中指数取3比取大于3的整数效果更好），则一次操作的时间复杂度为$O(9 \cdot n^{\frac{2}{3}})$

例1：糖果公园(WC2013)

这样就将询问操作转化成了区间和，只需求出$C[l..r][l..r]$（或$C[1..r][l..2n]$）的区间和，即为询问的答案。

对于修改操作，如果涉及的权值出现次数不超过$n^{\frac{1}{3}}$，则C中所有该权值对应的元素全部暴力重算，一共$O(n^{\frac{2}{3}})$个元素

如果出现次数超过$n^{\frac{1}{3}}$，特殊处理即可。

时间复杂度：如果使用二维树状数组维护数组C，则一次操作时间复杂度为$O(n^{\frac{2}{3}}\log^2 n)$

如果使用分层块状数组，假设指数取3（本题中指数取3比取大于3的整数效果更好），则一次操作的时间复杂度为$O(9 \cdot n^{\frac{2}{3}})$

显然用分层块状数组的方法更优。

例2：Just for fun EXT（陈立杰）

给出n个结点（一开始无边），每个结点有一个非负整数权值，四种可能的操作：

例2：Just for fun EXT（陈立杰）

给出n个结点（一开始无边），每个结点有一个非负整数权值，四种可能的操作：

在结点u和v间加一条边，若u, v已连通则无视该操作；

例2：Just for fun EXT（陈立杰）

给出n个结点（一开始无边），每个结点有一个非负整数权值，四种可能的操作：

- 在结点u和v间加一条边，若u, v已连通则无视该操作；
- 修改一个结点的权值；

例2：Just for fun EXT（陈立杰）

给出n个结点（一开始无边），每个结点有一个非负整数权值，四种可能的操作：

- 在结点u和v间加一条边，若u, v已连通则无视该操作；
- 修改一个结点的权值；
- 询问u到v的路径上权值第k大的结点，若u, v尚未连通则无视该操作；

例2：Just for fun EXT（陈立杰）

给出n个结点（一开始无边），每个结点有一个非负整数权值，四种可能的操作：

- 在结点u和v间加一条边，若u, v已连通则无视该操作；
- 修改一个结点的权值；
- 询问u到v的路径上权值第k大的结点，若u, v尚未连通则无视该操作；
- 询问u到v的路径上，所有权值小于等于z的结点的权值之积mod 28256292的余数，若u, v尚未连通则无视该操作。

例2：Just for fun EXT（陈立杰）

给出n个结点（一开始无边），每个结点有一个非负整数权值，四种可能的操作：

在结点u和v间加一条边，若u, v已连通则无视该操作；

修改一个结点的权值；

询问u到v的路径上权值第k大的结点，若u, v尚未连通则无视该操作；

询问u到v的路径上，所有权值小于等于z的结点的权值之积mod 28256292的余数，若u, v尚未连通则无视该操作。

$1 \le n \le 2 * 10^5, 1 \le$操作数$q \le 4 * 10^5$。强制在线。

例2：Just for fun EXT（陈立杰）

先考虑允许离线的情况。

例2：Just for fun EXT（陈立杰）

先考虑允许离线的情况。

求欧拉DFS序，并建立5个权值二维数组

例2：Just for fun EXT（陈立杰）

先考虑允许离线的情况。

求欧拉DFS序，并建立5个权值二维数组

其中一个元素值为1和-1，树状数组维护区间和，用于回答权值第k大的询问

例2：Just for fun EXT（陈立杰）

先考虑允许离线的情况。

求欧拉DFS序，并建立5个权值二维数组

其中一个元素值为1和-1，树状数组维护区间和，用于回答权值第k大的询问

另三个的元素值分别表示对应权值中2、3、784897的因数个数，及其相反数，当然只存储非0的，树状数组维护区间和

例2：Just for fun EXT（陈立杰）

先考虑允许离线的情况。

求欧拉DFS序，并建立5个权值二维数组

其中一个元素值为1和-1，树状数组维护区间和，用于回答权值第k大的询问

另三个的元素值分别表示对应权值中2、3、784897的因数个数，及其相反数，当然只存储非0的，树状数组维护区间和

最后一个的元素值为对应权值本身去掉所有2、3、784897因数后的值及其mod 28256292的逆元，树状数组维护区间积

例2：Just for fun EXT（陈立杰）

先考虑允许离线的情况。

求欧拉DFS序，并建立5个权值二维数组

其中一个元素值为1和-1，树状数组维护区间和，用于回答权值第k大的询问

另三个的元素值分别表示对应权值中2、3、784897的因数个数，及其相反数，当然只存储非0的，树状数组维护区间和

最后一个的元素值为对应权值本身去掉所有2、3、784897因数后的值及其mod 28256292的逆元，树状数组维护区间积

以上4个数组用于回答权值积的询问。

例2：Just for fun EXT（陈立杰）

本题的难点在于在线合并

例2：Just for fun EXT（陈立杰）

本题的难点在于在线合并

使用启发式合并的方法，将较小树拆掉后插入较大树

例2：Just for fun EXT（陈立杰）

本题的难点在于在线合并

使用启发式合并的方法，将较小树拆掉后插入较大树

合并欧拉DFS序，只要将较小树重建有根树，求出欧拉DFS序后插入较大树的欧拉DFS序当中即可

例2：Just for fun EXT（陈立杰）

本题的难点在于在线合并

使用启发式合并的方法，将较小树拆掉后插入较大树

合并欧拉DFS序，只要将较小树重建有根树，求出欧拉DFS序后插入较大树的欧拉DFS序当中即可

合并权值二维数组，可以当成动态序列，用动态标号法

例2：Just for fun EXT（陈立杰）

本题的难点在于在线合并

使用启发式合并的方法，将较小树拆掉后插入较大树

合并欧拉DFS序，只要将较小树重建有根树，求出欧拉DFS序后插入较大树的欧拉DFS序当中即可

合并权值二维数组，可以当成动态序列，用动态标号法

也可以将其分块后插入。

例3：Version Controlled IDE(UVA 12538)

维护一个字符串，三种操作：

例3：Version Controlled IDE(UVA 12538)

维护一个字符串，三种操作：

- 在某个位置插入一个长度不超过100的字符串；

例3：Version Controlled IDE(UVA 12538)

维护一个字符串，三种操作：

- 在某个位置插入一个长度不超过100的字符串；
- 删除一个子串；

例3：Version Controlled IDE(UVA 12538)

维护一个字符串，三种操作：

- 在某个位置插入一个长度不超过100的字符串；
- 删除一个子串；
- 询问时间tm（即第tm次插入删除操作之前）该字符串的第l到第r个字符组成的子串。

例3：Version Controlled IDE(UVA 12538)

维护一个字符串，三种操作：

- 在某个位置插入一个长度不超过100的字符串；
- 删除一个子串；
- 询问时间tm（即第tm次插入删除操作之前）该字符串的第l到第r个字符组成的子串。

$1 \leq$操作数$q \leq 5 * 10^5$，插入的字符串总长度不超过10^6，询问中输出的字符串总长度不超过$2 * 10^5$。

例3：Version Controlled IDE(UVA 12538)

这种问题一般使用伸展树(Splay Tree）进行维护

例3：Version Controlled IDE(UVA 12538)

这种问题一般使用伸展树(Splay Tree）进行维护

一个子串在伸展树中可以通过伸展操作形成一棵完整的子树，因此插入删除操作其实就是插入或一棵子树。

例3：Version Controlled IDE(UVA 12538)

这种问题一般使用伸展树(Splay Tree）进行维护

一个子串在伸展树中可以通过伸展操作形成一棵完整的子树，因此插入删除操作其实就是插入或一棵子树。

用数组存储这棵伸展树，下标表示结点

例3：Version Controlled IDE(UVA 12538)

这种问题一般使用伸展树(Splay Tree）进行维护

一个子串在伸展树中可以通过伸展操作形成一棵完整的子树，因此插入删除操作其实就是插入或一棵子树。

用数组存储这棵伸展树，下标表示结点

以时间轴为第一维，下标为第二维建立二维数组，存储每个结点的修改记录

例3：Version Controlled IDE(UVA 12538)

这种问题一般使用伸展树(Splay Tree）进行维护

一个子串在伸展树中可以通过伸展操作形成一棵完整的子树，因此插入删除操作其实就是插入或一棵子树。

用数组存储这棵伸展树，下标表示结点

以时间轴为第一维，下标为第二维建立二维数组，存储每个结点的修改记录

若下标为j的结点在时刻i被修改，则该数组的元素$[i][j]$为该结点修改后的样子。

例3：Version Controlled IDE(UVA 12538)

这样，任意一个结点在任意一个时刻的样子都可以用$O(\texttt{log}\,q)$时间，从该二维数组中找到

例3：Version Controlled IDE(UVA 12538)

这样，任意一个结点在任意一个时刻的样子都可以用$O(\texttt{log}\, q)$时间，从该二维数组中找到

可持久化就这样实现了......

例3：Version Controlled IDE(UVA 12538)

这样，任意一个结点在任意一个时刻的样子都可以用$O(\log q)$时间，从该二维数组中找到

可持久化就这样实现了......

问题在于伸展树是均摊平衡的，现在由于可持久化，要将其改为每次操作后严格平衡

例3：Version Controlled IDE(UVA 12538)

这样，任意一个结点在任意一个时刻的样子都可以用$O(\log q)$时间，从该二维数组中找到

可持久化就这样实现了......

问题在于伸展树是均摊平衡的，现在由于可持久化，要将其改为每次操作后严格平衡

只要在每次插入删除之后人为地进行一些伸展操作来进行调整（比如每次将深度最大的结点伸展到根），保证深度即可。

例3：Version Controlled IDE(UVA 12538)

这样，任意一个结点在任意一个时刻的样子都可以用$O(\log q)$时间，从该二维数组中找到

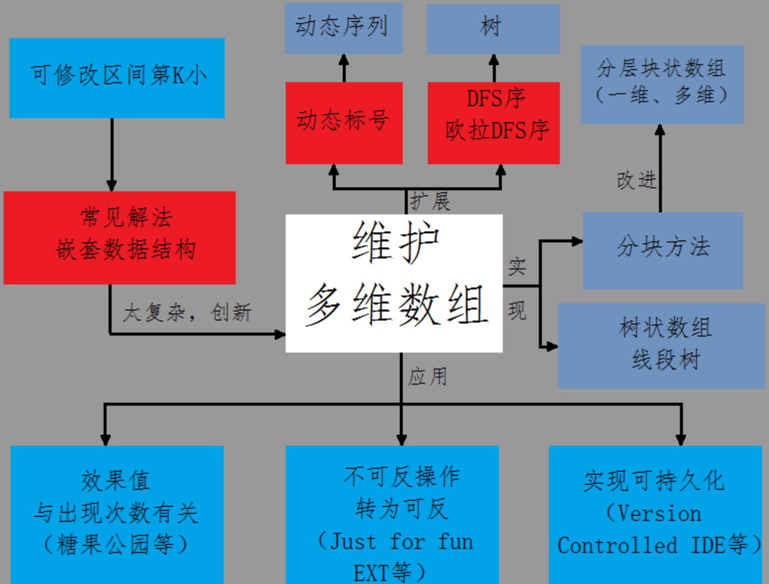
可持久化就这样实现了......

问题在于伸展树是均摊平衡的，现在由于可持久化，要将其改为每次操作后严格平衡

只要在每次插入删除之后人为地进行一些伸展操作来进行调整（比如每次将深度最大的结点伸展到根），保证深度即可。

这些操作引起的结点修改自然也要在二维数组中记录。

总结


动态序列
树
可修改区间第K小
分层块状数组
（一维、多维）
动态标号
DFS序
欧拉DFS序
改进
扩展
常见解法
嵌套数据结构
维护
多维数组
实
现
分块方法
太复杂，创新
树状数组
线段树
应用
效果值
与出现次数有关
（糖果公园等）
不可反操作
转为可反
（Just for fun
EXT等）
实现可持久化
（Version
Controlled IDE等）

总结

我们要充分发挥人类智慧，探索、测试、改进解决方案的能力……

参考文献

[1]陈立杰，《可持久化数据结构研究》，2012。

[2]许昊然，《数据结构漫谈》，2012。

[3]List Order Maintenance && Monotonic List Labeling Density Maintenance.

[4]http://en.wikipedia.org/wiki/Euler_tour_technique

感谢

感谢CCF提供了这次交流的机会。

感谢四十五中张老师对我的培养。

感谢陈立杰、顾昱洲等对我的引导和榜样作用。

感谢黄嘉泰、许昊然、罗干、董宏华、陶润洲等在我完成本文的过程中对我的帮助。

感谢多年来为我解答问题、给我提供资料和其它帮助的所有人，没有你们，我现在就不会在这里。

最后，感谢AHOI2012，磨练了我的意志，让我变得更坚强。