树状数组

树状数组、扩展功能

树状数组

树状数组（BIT，Binary Indexed Tree）

- 树状数组是常用的支持动态维护前缀和的数据结构
- 同等情况下树状数组的常数比线段树小得多（没有递归）

常用操作

- ask(k)：询问前k个数的前缀和
- add(x, k)：将第k个数的权值加上k

- 从基本功能上说，树状数组只支持前缀和查询，单点修改

树状数组

原理

- 树状数组从本质上讲是一种倍增算法
- 任何区间[1,n]都可以被表示为不超过log(n)个长度为2的幂的区间

证明

- 利用二进制分解即可，例如：$11=(1011)_2=2^3+2^1+2^0=8+2+1$
- 因此区间[1,11]可以拆分为[1,8]+[9,10]+[11,11]

树状数组

二进制分解

- lowbit运算：求x二进制最后一个1和后面的0
- lowbit(x) = x & (-x)
- 例如：lowbit(11)=1，lowbit(10)=2，lowbit(8)=8
- 记区间二进制分解为$[1,x_1]$, $[x_1+1,x_2]$, …, $[x_{k-1}+1,x_k]$
- 我们发现以下性质：$x_{k-1}=x_k$ - lowbit(x_k)（减法可以换成异或）
- 而总区间数量就等于x_k的二进制表示中1的个数

树状数组

区间表示

- 记Sum([l, r])表示区间[l,r]的和，树状数组的区间划分如下：

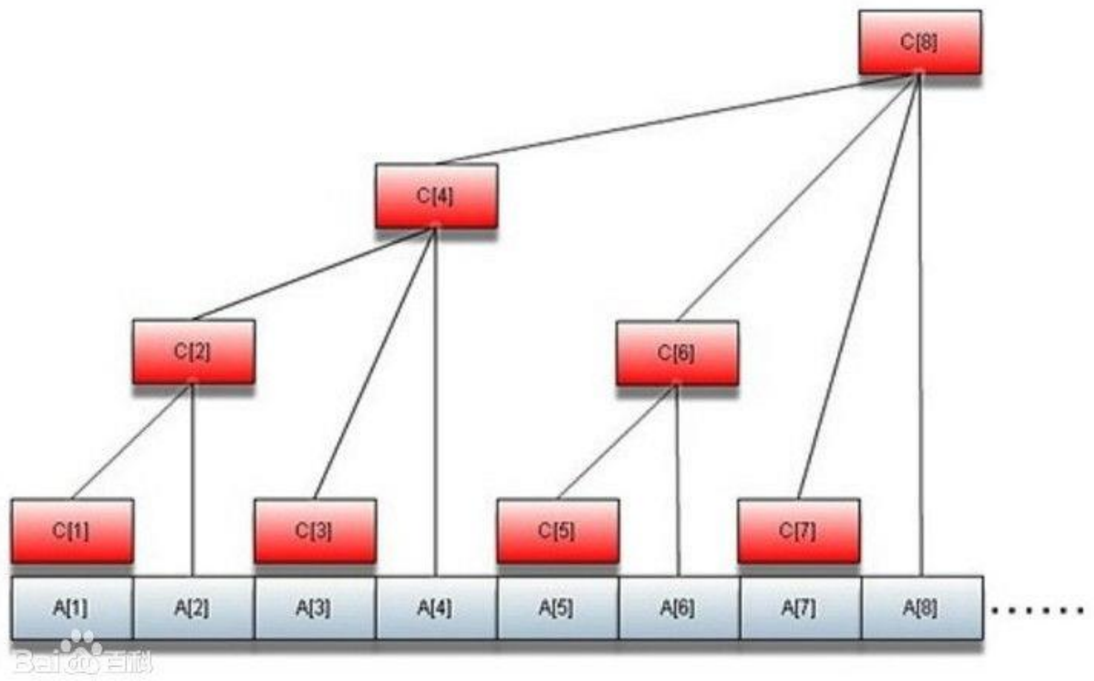
$$tree[n] = Sum([x - lowbit(x) + 1, x])$$

- 记x_0=n， x_{k+1}=x_k-lowbit(x_k)
- 那么有Sum([1,n])=Sum([1,x_k])+ Sum([x_k+1,x_{k-1}])+…+ Sum([x_1+1,x_0])

  =tree[x_k]+tree[x_{k-1}]+…+tree[x_0]
- 因此我们可以通过迭代求lowbit函数来求1到n的前缀和

树状数组

树状数组的结构



树状数组

询问（ask）

- 询问相当于通过lowbit运算不断向前跳求和的过程
- x每跳一次二进制数减少一个1，时间复杂度O(logn)

```
void ask(int x) {
    int ans = 0;
    for(; x; x ^= x & -x) ans += c[i];
    return ans;
}
```

树状数组

修改（add）

- 修改相当于通过lowbit运算不断向上更新区间和的过程
- x每跳一次二进制数减少一个1或左移1位，时间复杂度O(logn)

```
void add(int x, int y) {
    for(; x <= n; x += x & -x) c[x] += y;
}
```

树状数组

区间询问

- 区间[l,r]的和可以表示为两个前缀和的差，即：

$$Sum([l,r]) = ask(r) - ask(l-1)$$

树状数组

例题：逆序对

- 已知一个长度为n的数列a_n，求a_n中的逆序对数量。$n \le 10^6$

树状数组

例题：逆序对

- 已知一个长度为n的数列a_n，求a_n中的逆序对数量。$n \le 10^6$

- 首先将权值离散化，这样就a_1到a_n能就能对应1~n之间的整数了
- 如果我们将出现过的数权值记为1，那么树状数组能统计小于等于某个值的数有多少个
- 因为是求逆序对，所以我们从右到左将每个数加入树状数组
- 对于每个数ai，逆序对数量就是ask(rank(ai)-1)
- 总体的时间复杂度O(nlogn)

树状数组

拓展功能

- 树状数组的基本功能为单点修改，区间查询
- 树状数组还可以支持区间修改，单点查询
- 树状数组还可以支持区间修改，区间查询

树状数组

例题： A Tiny Problem with Integers

- N个数，Q组操作，区间修改，单点询问。N，Q≤10^5

树状数组

例题： A Tiny Problem with Integers

- N个数，Q组操作，区间修改，单点询问。N，Q≤10^5

- 用树状数组维护差分d[i]的前缀和
- 区间[l,r]修改相当于在r+1这一点减去d，在l这一点加上d
- 单点询问相当于求区间[1,x]的前缀和

树状数组

例题：A Simple Problem with Integers(poj3468)

- N个数，Q组操作，区间修改，区间询问。N，Q≤10^5
- 用线段树显然可以做，但请考虑树状数组的做法

树状数组

例题： A Simple Problem with Integers(poj3468)

- N个数，Q组操作，区间修改，区间询问。N，Q≤10^5

- 记原来的数组为a，差分为d，那么满足：

$$\sum_{i=1}^{n} \mathrm{a}[i] = \sum_{i=1}^{n}\sum_{j=1}^{i} d[i] = \sum_{i=1}^{n}(n-i+1)d[i]$$

$$= (x+1)\sum_{i=1}^{n} d[i] - \sum_{i=1}^{n} i * d[i]$$

树状数组

例题： A Simple Problem with Integers(poj3468)

- N个数，Q组操作，区间修改，区间询问。N，Q≤10^5

- 记原来的数组为a，差分为d
- 我们在前面的题目中已经维护了 $\sum_{i=1}^{n} d[i]$，记为树状数组c1
- 考虑如何维护$\sum_{i=1}^{n} i*d[i]$，记为树状数组c2
- 我们在l这一点上加上l*d，在r+1这一点上减去(r+1)*d

树状数组

例题：A Simple Problem with Integers(poj3468)

- N个数，Q组操作，区间修改，区间询问。N，$Q \le 10^5$

- 记原来的数组为a，差分为d，a的初始前缀和为sum
- 则修改后的两段区间的前缀和分别如下：
- 区间[1,r]的前缀和=sum[r]+(r+1)*ask(c1,r)-ask(c2,r)
- 区间[1,l-1]的前缀和= sum[l-1] +l*ask(c1,r-1)-ask(c2,l-1)

线段树

线段树、延迟标记、可持久化

线段树

线段树（segment tree）

- 线段树又叫区间树，线段树的核心思想是分治
- 线段树中父节点的儿子分别为他的左半区间和右半区间

操作

- build：建树
- change：修改
- ask：查询
- …

线段树

线段树入门

- 由于篇幅有限，这里只提供传送门

- 洛谷日报#4 [皎月半洒花] 浅谈线段树（Segment Tree）
https://pks-loving.blog.luogu.org/senior-data-structure-qian-tan-xian-duan-shu-segment-tree

线段树

部分细节问题

- 存储n个元素即n个叶节点的线段树至少需要多少空间?
- 假设根为1，父节点k的两个儿子分别为2*k和2*k+1

线段树

部分细节问题

- 存储n个元素即n个叶节点的线段树至少需要多少空间?
- 假设根为1，父节点k的两个儿子分别为2*k和2*k+1

- 大家可能会给出两个答案4n或8n

线段树

部分细节问题

- 存储n个元素即n个叶节点的线段树至少需要多少空间?
- 假设根为1，父节点k的两个儿子分别为2*k和2*k+1

- 情况一
- 假设父节点区间为[a,b]，两个儿子为[a,(a+b)/2]和[(a+b)/2+1,b]
- 这时右儿子不大于左儿子，右儿子的深度不会大于左儿子深度
- 考虑极端情况，n=2^k+1，左儿子一直比右儿子大1
- 这时深度为k+2，而$2^{k+2}<2^{k+2}+4=4n$，因此需要4n空间

线段树

部分细节问题

- 存储n个元素即n个叶节点的线段树至少需要多少空间?
- 假设根为1，父节点k的两个儿子分别为2*k和2*k+1

- 情况二
- 假设父节点区间为[a,b)，两个儿子为[a,(a+b)/2)和[(a+b)/2+1,b)
- 这时左儿子不大于右儿子，右儿子的深度不会小于左儿子深度
- 考虑极端情况，n=2^k+1，右儿子一直比左儿子大1
- 这时深度为k+2，而$2^{k+3}-1<2^{k+3}+8=8n$，因此需要8n空间

线段树

部分细节问题

- 存储n个元素即n个叶节点的线段树至少需要多少空间?
- 假设根为1，父节点k的两个儿子分别为2*k和2*k+1

- 在开区间的线段树中，将(a+b)/2向下取整该为(a+b+1)/2
- 假设父节点区间为[a,b)，两个儿子为[a,(a+b+1)/2)和[(a+b+1)/2+1,b)
- 此时有左儿子不小于右儿子，只需要申请4n空间就可以了

延迟标记

延迟标记

- 在进行区间修改时，无法完全更新区间[l,r]内的信息
- 更新信息相当于将值传到叶节点，时间复杂度会退化到O(n)
- 延迟标记的作用是暂时保存修改信息，不向下传递

延迟标记

带延迟的修改

```
void change(p, l, r, d) {
    if(l<=l(p) && r>=r(p)) { 修改并return; }
    down(p);
    if (l<=mid(p)) change(ls(p),...);
    if (r>mid(p)) change(rs(p),...);
    update(p);
}
```

延迟标记

带延迟的询问

```
int ask(p, l, r, d) {
    if(l<=l(p) && r>=r(p)) return 答案;
    down(p);
    int ans = 0;
    if (l<=mid(p)) ans+=ask (ls(p),...);
    if (r>mid(p)) ans+=ask(rs(p),...);
    return ans;
}
```

线段树

延迟标记

- 可以发现change中最多做2logn次down和2logn次up
- 可以发现ask中最多做2logn次down
- 也就是说延迟标记的引入只增加了O(logn)次操作
- 整体复杂度O(logn)

线段树

例题：Interval GCD

- N个数，M条指令，区间修改，区间求gcd。N，Q≤$2*10^5$

线段树

例题：Interval GCD

- N个数，M条指令，区间修改，区间求gcd。N，Q≤$2*10^5$

- 由gcd(x,y)=gcd(x,y-x)可知：
- $gcd(a_1,a_2,\ldots,a_n)=gcd(a_1,a_2-a_1,a_3-a_2,\ldots,a_n-a_{n-1})$
- 设a为原数列，d为差分数列
- 我们分别用线段树维护a的值和d的值，同时维护d的gcd
- 那么有gcd(a[l],…,a[r])=gcd(a[l],ask(1,l+1,r))