# 树状数组，倍增和树上问题

2024 年 7 月

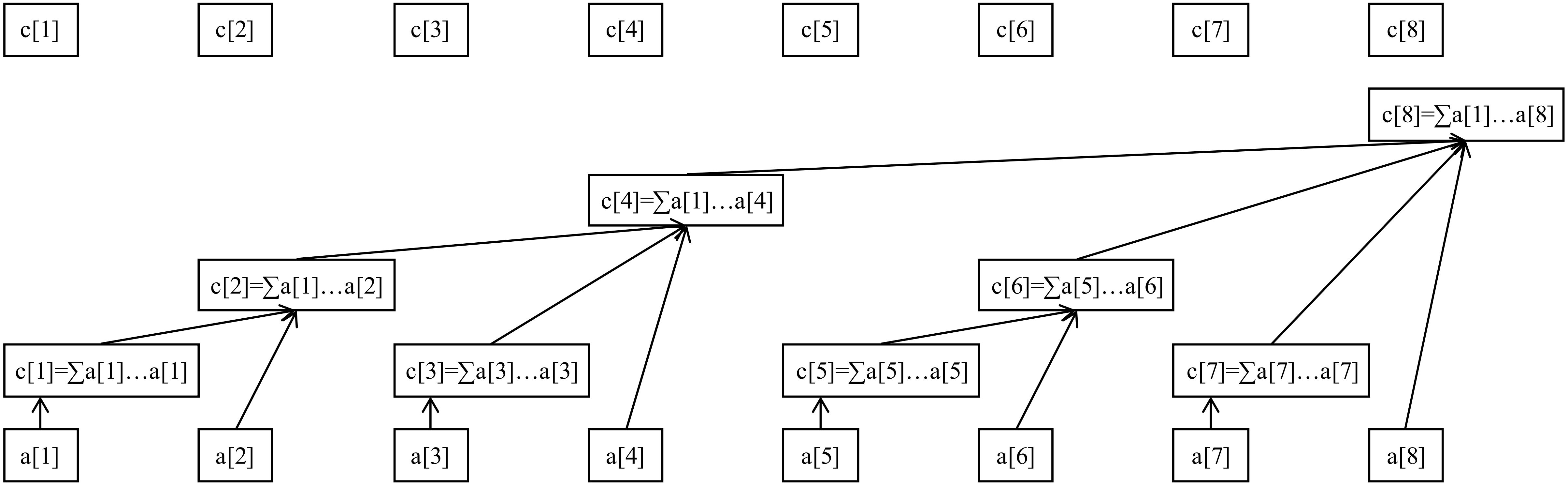
feecle8146

# 树状数组

## 单点修改区间求和

### 树状数组的结构

下图是一棵长度为 8 的树状数组。



# 树状数组



树状数组的循环不变式

树状数组上，$c_i$ 维护的是原序列中 $(i - lowbit(i), i]$ 这些位置的和。

由于没有标记，所以它的循环不变式就是：$\forall i$，$c_i = \sum_{k=i-lowbit(i)+1}^{i} a_k$。

思考：如何由 $a$ 构建树状数组 $c$？
*前缀和*

思考：由该循环不变式，你能不能直接推出树状数组单点修改和区间求和的具体过程？

# 树状数组

## 单点修改区间求和

思考：若传入的 $x$ 为 0 会发生什么？

*会死循环，所以使用树状数组前必须确认下标是正数。*

### 单点修改

修改 $a_x$ 为 $y$ 后，我们需要知道哪些 $c_i$ 发生了改变。也就是，哪些 $i$ 满足 $x \in (i - lowbit(i), i]$。

事实上，这些 $i$ 就是所有把 $x$ 为 0 的位改成 1，并把之后的位改成 0 得到的数。例如选择橙色的位，

$$x = 10010110$$
$$i = 10001000$$

据此写出代码：

```cpp
int c[N + 5];
void Upd(int x, int y) { // a[x] += y
    while (x <= n) c[x] += y, x += x & -x;
    // here, x & -x = lowbit(x) always holds
}
```

# 树状数组

## 单点修改区间求和

区间求和

你也许已经发现，树状数组是一棵“前倾”而不对称的树，这使得前缀和后缀在树状数组上的简单程度不相同。

所以，我们通常把区间求和拆成前缀和相减来处理。

1 到 $x$ 的前缀，可以拆成哪些 $(i - lowbit(i), i]$?

从低位到高位依次剥离 $x$ 的 1 即可。
据此写出代码：

```
int Query(int x) {
    int res = 0;
    while (x) res += c[x], x -= x & -x;
    return res;
}
```

# 树状数组

## 单点修改区间求和

树状数组维护信息、支持修改的性质

我们昨天已经学到，线段树维护的信息需要可合并，修改标记需要封闭。

树状数组不支持（一般性的）区间修改，所以只需要讨论信息的性质和单点修改需要满足的要求就行了。

信息：可合并性 +（如果要求区间的话）可减性

修改：容易计算变化量（因为不能重新 pushup）

橙色部分是相比线段树的不同之处。

# 树状数组

单点修改区间求和

**LIS**

无来源

给你一个数组，求最长严格上升子序列。

$$n \le 500000, a_i \le 10^9$$

首先将 $a_i$ 离散化。

dp 式为 $f(i) = \max\limits_{j<i, a_j<a_i} f(j) + 1$。

*使用数据结构优化其它算法时，*
*需要想清楚其中存储的数据的含义：*
*时间？下标？存储内容？合并方式？*
*满不满足我想用的数据结构的要求？*

维护树状数组 $c$，其上在循环到 $i$ 之前存储了 $f(1), \dots f(i-1)$ 以 $a_j$ 为下标的最大值信息，那么 $f(i)$ 只需要求前缀最大值。

同时，同一 $a_i$ 对应的 $f$ 值不减，因此变化量也容易计算。

# 树状数组

## 单点修改区间求和

**逆序对**

无来源

给你一个数组，求逆序对数。

$$n \le 500000, a_i \le 10^9$$

首先将 $a_i$ 离散化。

维护树状数组 $c$，其上在循环到 $i$ 之前存储 $a_1, \ldots, a_{i-1}$ 以 $a_j$ 为下标、1 为值的和信息，那么 $i$ 处逆序对数只需要求后缀和。

练习：P1637

# 倍增



## $2^k$ 级祖先

回顾“树”的定义：$n$ 个点、$n-1$ 条边的连通无向图，可以选一个点作为根，从而定义“深度”“父亲”“祖先”“子树”等。

设数组 $p_{i,j}$ 表示 $i$ 的 $2^j$ 级祖先，我们有 $p_{i,j}=p_{p_{i,j-1},j-1}$，若不存在定义为 0。这样的好处是不需要特判，因为全局变量本来初值就是 0。

需要注意 $p$ 求出的顺序：可以按照 $j$ 从小到大再 $i$ 从小到大，也可以 dfs 过程中顺便求出。

# 倍增



## LCA

定义 $x, y$ 的最近公共祖先 LCA 为两者深度最深的公共祖先。

性质：$x, y$ 在树上的路径可以拆分为 $x \to LCA \to y$。

LCA 的求法分两步：

1. 使 $x, y$ 中较深的一方爬到另一方的深度。
2. $x, y$ 同时爬树，直到两者相遇。

思考：两步分别怎么实现？

```
int LCA(int x, int y) {
    if (d[x] < d[y]) swap(x, y);
    for (int i = 20; i >= 0; i--)
        if (d[p[x][i]] >= d[y]) x = p[x][i];
    if (x == y) return x; // 注意
    for (int i = 20; i >= 0; i--) {
        if (p[x][i] ^ p[y][i]) {
            x = p[x][i], y = p[y][i];
        }
    }
    return p[x][0];
}
```

# 倍增



## 树上路径信息

对于有**可减性**的信息，可以将 $x \to y$ 的路径拆分成 $O(1)$ 条到根链。
需要特别注意 LCA 周围。

例 1：多次询问两点间路径上的边权和。

$$dis(x,y) = dis_x + dis_y - 2dis_{lca}$$

例 2：多次询问两点间路径上的点权和。

$$dis(x,y) = dis_x + dis_y - 2dis_{lca} + a_{lca}$$

# 倍增



## 树上路径信息

对于没有可减性但有可合并性的信息，可以将 $x \to y$ 的路径拆分成 $O(\log n)$ 条直上直下的、长度为 $2^k$ 的链，有时需要注意顺序问题。

这里，信息设计的方法和线段树是一样的。

例 1：询问 $x \to y$ 路径上的点点权依次拼接形成数组的最大子段和。
维护四元组。

例 2：点权保证是 0~9 的整数，询问 $x \to y$ 路径上的点点权依次拼接形成整数的值。
维护路径长度和权值。需要对上和下维护两种倍增数组。

# 倍增



幂等信息

某个信息 $x$ 是幂等的，就是说 $x, x$ 合并之后还是 $x$，例如：路径最大值、路径 gcd 等都是幂等信息。

对于幂等信息，原先合并要求“不重不漏、顺序不变”，现在只需要“不漏”就够了。

可以把任意一条直上直下的长为 $len$ 路径拆成两条长为 $2^k$ 的路径，其中 $2^k \le len, 2^{k+1} > len$：它们可能会重复，但没关系。这样，直上直下的路径只需要一次合并，一般路径只需要 $3 = O(1)$ 次合并。

# 倍增



幂等信息

把该思想用到不带修序列信息维护上，有时能取得比线段树更优的询问复杂度，称此时的倍增数组为 ST 表。

换句话说，ST 表就是特化到序列上的树上倍增，用于维护幂等信息。

为确保序列上的查询是 $O(1)$ 的，需要预处理 2 的幂次。

练习：P3865, P2880

```
int Query(int l, int r) {
    int k = lg2[r - l + 1];
    return min(f[l][k], f[r - (1 << k) + 1][k]);
}

lg2[0] = -1;
for (int i = 1; i <= n; i++) lg2[i] = lg2[i >> 1] + 1;
```

# 倍增

## 小结

倍增维护可合并的信息，对信息的操作与线段树别无二致。同时，需要注意：

1. 树上信息变化更加复杂，时常需要结合树的形态分类讨论；
2. 幂等信息只需不漏。

# 处理树上问题的方法



## dfs 序

树上问题总体是不如序列问题直观的，因此我们希望将树上问题转化为序列问题。

按照右图的方法得到 dfn 数组，则 $x$ 子树的 dfn 恰好构成了区间 $[dfn_x, dfn_x + sz_x)$。这就把子树限制转化为了区间限制。

```
int sz[N + 5], dfn[N + 5], sign;
void dfs(int x) {
    sz[x] = 1, dfn[x] = ++sign;
    for (int y : g[x]) {
        dfs(y), sz[x] += sz[y];
    }
}
```

# 处理树上问题的方法



**经典问题** 无来源

给出一棵树，$q$ 次操作，每次操作都是链加边权/点权。
求操作完后每条边/每个点的权值。

要求 $O(n+q)$。

树上差分是树上前缀和的逆操作。类比序列问题，我们希望把链加单点差化为单点加查询“前缀和”：在树上，一般认为这里的前缀和是子树和。

首先，链加可以看作 $O(1)$ 个 $x$ 的到根加。怎么转为单点加子树和？

直接令 $b_x$ 加上 $v$，再求子树和，就达到了到根加效果。

# 处理树上问题的方法



**经典问题** 无来源

给出一棵树，$q$ 次操作，每次操作都是链加边权/点权。
求操作完后每条边/每个点的权值。

要求 $O(n+q)$。

对于链加点权，一般写成 $x,y$ 加，$lca, fa_{lca}$ 减。

对于链加边权，一般令点权等于 $x, fa_x$ 间的边权，$x,y$ 加，$lca$ 减两倍。