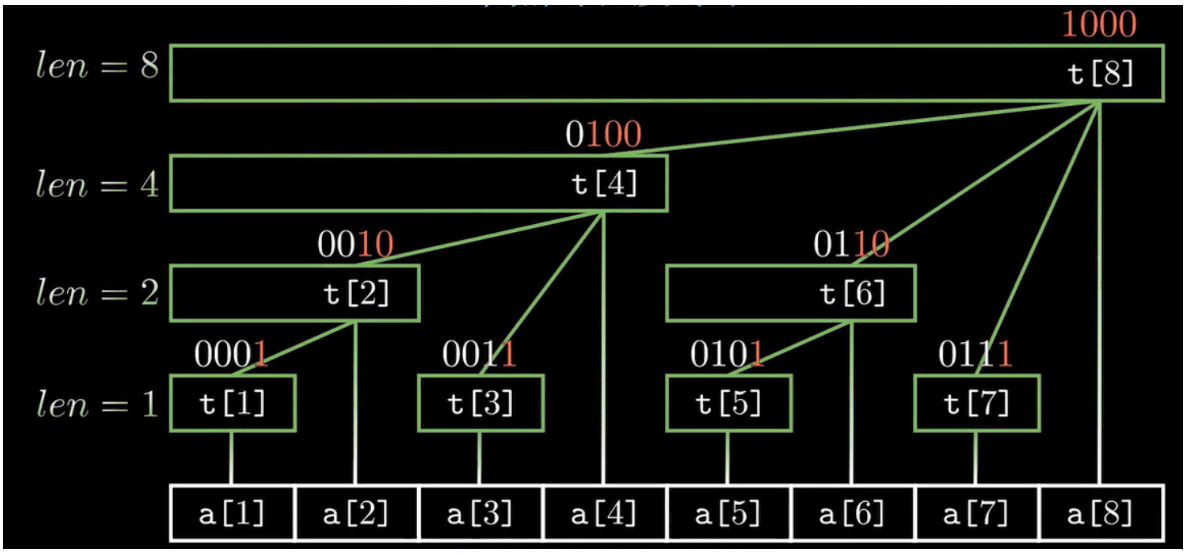
树状数组

树状数组是一个动态维护前缀和的工具



- `t[x]` 保存以 `x` 为根的子树中叶节点值的和
- `t[x]` 节点的长度等于 `lowbit(x)`
- `t[x]` 节点的父节点为 `t[x + lowbit(x)]`

总体上树状数组支持两个基本操作：

- 单点修改
- 查询前缀和

0. 初始化树状数组

> 时间复杂度O(NlogN)

```
int a[maxn], t[maxn];
for(int i = 1; i <= n; ++ i) add(x, a[i]);
```

> 时间复杂度O(N)

考虑到树状数组的本质，我们知道一个t[x]管辖的是[x−lowbit(x)+1,x]中所有数的和，所以我们可以对a[i]求一个前缀和，求前缀和的时候更新t[x]数组，即t[x]=sum[x]-sum[x-lowbit(x)]，这样就能线性构造了，时间复杂度O(n)。

```
int t[maxn], sum[maxn], a[maxn]; //a[]原数组 sum[]前缀和数组 t[]树状数组
for(int i = 1; i <= n; ++ i){
    sum[i] = sum[i - 1] + a[i];
    t[i] = sum[i] - sum[i - lowbit(i)];
}
```

1. 单点修改 区间查询

> 单点修改 查询前缀和

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

> 查询前缀和

```
int ask(int x){
    int ans = 0;
    for(; x; x -= x & -x) ans += t[x];
    return ans;
}
```

> 单点修改，单点查询

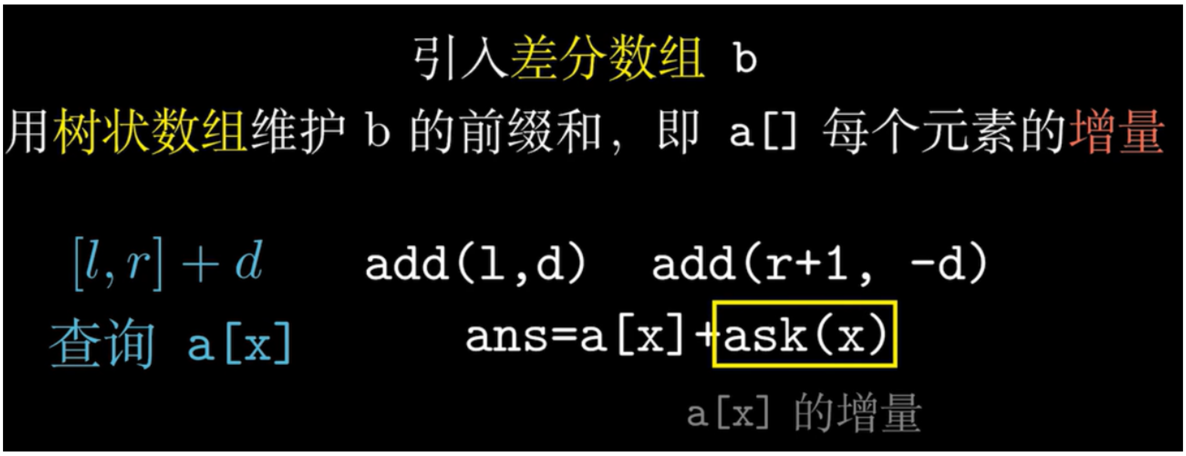
```
add(x, k)  //单点修改
ask(x) - ask(x - 1) //单点查询
```

> 单点修改，区间查询

```
add(x, k)  //单点修改
ask(r) - ask(l - 1) //区间查询
```

---

2. 区间修改 单点查询



- `t[x]` 维护的是差分数组b[]的前缀和

区间修改[l , r] 即单点修改 `差分数组b[x]` 的 `l` 和 `r + 1` 两点

单点查询 `a[x]` 时， `a[x]` 的增量为 `差分数组b[x]` 的前缀和

> 区间修改

```
//add、ask操作与单点修改、区间查询中相同
add(l,d);
add(r + 1, -d);
```

> 单点查询

```
a[x] + ask(x);
```

- 应用例题：`AcWing 242. 一个简单的整数问题`

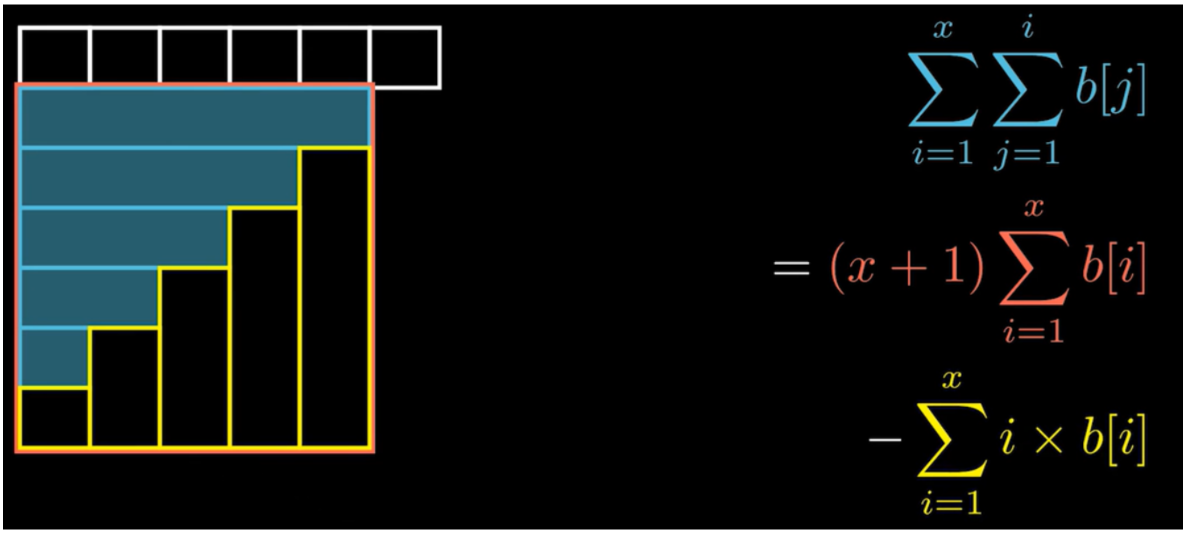
---

3. 区间修改 区间查询 (使用线段树更方便)

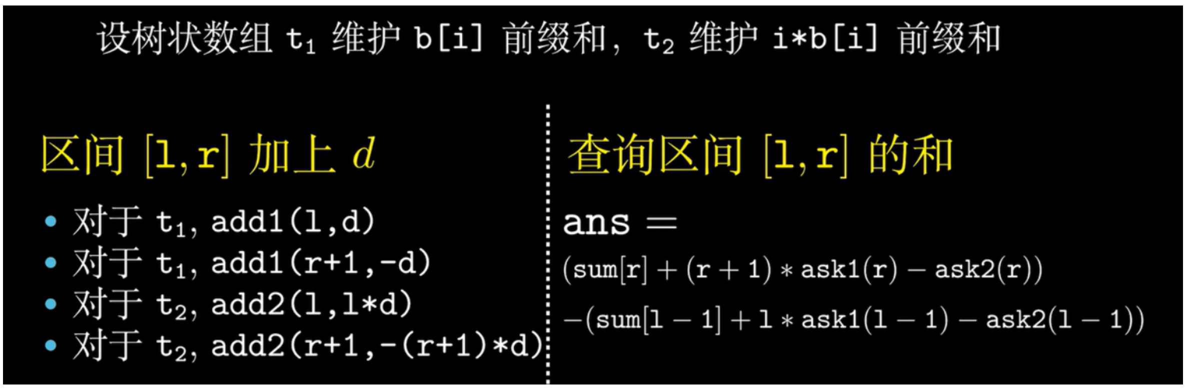
区间查询：`b[ ]为差分数组`

$$区间变化的和为：\triangle S=\sum_{i=1}^{x}\sum_{j=1}^{i}b[j]$$

$$区间和 = 前缀和 S + 区间变化的和 \triangle S$$



黄色部分需要另一个树状数组来维护 `i * b[i]` 的前缀和。



- 代码：

```
int t1[maxn], t2[maxn];
```

```
int sum[maxn]; //a[]数组的前缀和
void add1(int x, int k){
    for(; x <= n; x += x & -x) t1[x] += k;
}
int ask1(int x){
    int ans = 0;
    for(; x; x -= x & -x) ans += t1[x];
    return ans;
}
void add2(int x, int k){
    for(; x <= n; x += x & -x) t2[x] += k;
}
int ask2(int x){
    int ans = 0;
    for(; x; x -= x & -x) ans += t2[x];
    return ans;
}
```

> 区间修改

```
add1(l, d);
add1(r + 1, -d);
add2(l, l * d);
add2(r + 1, -(r + 1) * d);
```

> 区间查询

```
(sum[r] + (r + 1) * ask1(r) - ask2(r)) - (sum[l - 1] + l * ask1(l - 1) - ask2(l - 1));
```

- 应用例题: `AcWing 243. 一个简单的整数问题2`

---

4. 树状数组与逆序对

在 `数值范围` 上建立一个树状数组， ~~较难理解,具体看算法竞赛进阶指南0x42~~

时间复杂度为 `O((N + M)logM)`, `N` 为数组大小、 `M` 为数值范围的大小。

```
int a[maxn], t[maxm]; //maxm为a[]数组中的数据范围

for(int i = n; i; i --){
    ans += ask(a[i] - 1);
    add(a[i], 1);
}
```

- 整体代码

```
#include <bits/stdc++.h>
using namespace std;

const int maxn = 100005;
const int maxm = 100005;
```

```cpp
int a[maxn], t[maxm];
int n;

void add(int x, int val){
    for(; x <= n; x += x & -x) t[x] += val;
}
int ask(int x){
    int ans = 0;
    for(; x; x -= x & -x) ans += t[x];
    return ans;
}

int main(){
    cin >> n;
    for(int i = 1; i <= n; ++ i) cin >> a[i];
    int res = 0;
    for(int i = n; i; -- i){
        res += ask(a[i] - 1);
        add(a[i], 1);
    }
    cout << res << endl;
}
```

- 应用例题：`AcWing 241. 楼兰图腾`