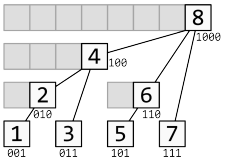
树状数组

Fenwick Tree / Binary Indexed Tree

树状数组

Idea：通过神奇的 `lowbit()` 操作使得 $c[x]$ 包含了其前一系列 $c[]$ 的和，管理一系列 $a[]$。



例如，$c[4]$ 管理了 $a[1...4]$，$c[6]$ 管理了 $a[5...6]$，$c[7]$ 只管理了 $a[7]$。

Complexity：$O(\lg n)$

Code：

```
int c[N];
inline int lowbit(int x){ return x & -x; }
inline int querySum(int x){
    int res = 0;
    for(; x; x -= lowbit(x))    res += c[x];
    return res;
}
inline void add(int x, int v){
    for(; x <= n; x += lowbit(x))   c[x] += v;
}
```

二维树状数组

Idea：树状数组扩展成二维，解决二维的单点/区间修改/求和问题。

Complexity：$O(\lg^2 n)$

Code：

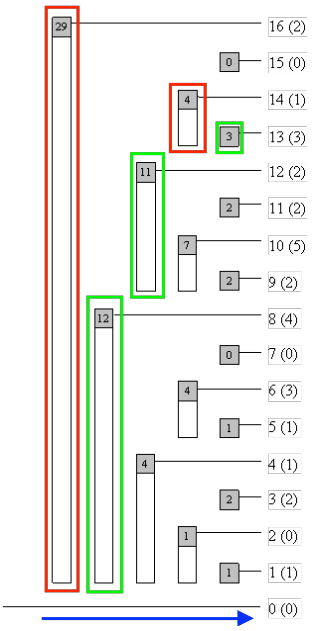
```
int c[N][N];
inline int lowbit(int x){ return x & -x; }
inline void add(int x, int y, int val, int c[][N]){
    for(int i = x; i <= n; i += lowbit(i))
        for(int j = y; j <= m; j += lowbit(j))
            c[i][j] += val;
}
inline int sum(int x, int y, int c[][N]){
    int res = 0;
    for(int i = x; i; i -= lowbit(i))
        for(int j = y; j; j -= lowbit(j))
            res += c[i][j];
    return res;
}
```

树状数组倍增

Problem：给定某个序列，要求维护的操作有：单点修改，求前缀和，搜索某个前缀和（类似于在前缀和数组上求 `lower_bound`）。

Idea：假设我们想要搜索前缀和为 val 的地方，设定一个 `pos` 指针，它初始为 0，最终将指向最大的前缀和小于 val 的位置；再设置一个变量 `sum`，存储 `pos` 处的前缀和；设置倍增的长度 `i`，最初为 $\lg n$（为了代码方便，一般取 20 即可），在倍增的过程中不断减小至 0。每一个状态（`pos`，`sum`，`i`）表示我们现在考虑的是位置 `pos+(1<<i)` 的前缀和，这个前缀和的值是 `sum+c[pos+(1<<i)]`，如果它大于等于了 val，那么我们减小倍增的长度 `i`；否则，我们把 `pos` 提到 `pos+(1<<i)` 处。

树状数组



各变量变化情况

Step	pos	sum	bit[pos + 2^i]	sum + bit	Lift
0	0	0	29	29	No
1	0	0	12	12	Yes
2	8	12	11	23	Yes
3	12	23	4	27	No
4	12	23	3	26	Yes

What's more：只要我们维护的信息具有**单调性**，就可以用这个方法。

Compelxity：$O(n \lg n)$

Code：

```
int search(int val){
    int pos = 0, sum = 0;
    for(int i = 20; i >= 0; i--)
        if(pos + (1<<i) <= n && sum + c[pos+(1<<i)] < val)
            pos += (1<<i), sum += c[pos];
    return pos + 1;
}
```