ST 表

Sparse Table

ST 表

Intro：解决可重复贡献问题的数据结构。所谓可重复贡献，即对于一种运算 opt，满足 $x \text{ opt } x = x$，例如 $\max(x, x) = x$、$\gcd(x, x) = x$；同时 opt 满足结合律，则该问题可以用 ST 表来解决。

Idea：倍增思想。以 $\max()$ 为例，$st[i][j]$ 表示包括 i 在内的连续 2^j 个数的最大值。

Complexity：$O(n \lg n)$ 初始化，$O(1)$ 查询。

Code：

```
void pre(){
    lg[1] = 0;
    lg[2] = 1;
    for(int i = 3; i <= n; i++)
        lg[i] = lg[i/2] + 1;
}
void init(){
    for(int j = 1; (1 << j) <= n; j++)
        for(int i = 1; i + (1 << j) - 1 <= n; i++)
            st[i][j] = max(st[i][j-1], st[i+(1<<(j-1))][j-1]);
}
int query(int l, int r){
    int k = lg[r - l + 1];
    return max(st[l][k], st[r-(1<<k)+1][k]);
}
```

二维 ST 表

Idea：ST 表扩展成二维，解决诸如二维 RMQ 问题等问题。

Complexity：$O(n^2 \lg^2 n)$ 初始化，$O(1)$ 查询。

Code：

```
int lg[N], st[N][N][9][9];
void pre(){
    lg[1] = 0, lg[2] = 1;
    for(int i = 3; i <= max(n, m); i++) lg[i] = lg[i/2] + 1;
}
void init(){
    for(int i = 1; i <= n; i++)
        for(int j = 1; j <= m; j++)
            st[i][j][0][0] = a[i][j];
    for(int ki = 0; (1 << ki) <= n; ki++){
        for(int kj = 0; (1 << kj) <= m; kj++){
            if(!ki && !kj)  continue;
            for(int i = 1; i + (1 << ki) - 1 <= n; i++){
                for(int j = 1; j + (1 << kj) - 1 <= m; j++){
                    if(!ki) st[i][j][ki][kj] = max(st[i][j][ki][kj-1], st[i][j+(1<<(kj-1))][ki][kj-1]);
                    else    st[i][j][ki][kj] = max(st[i][j][ki-1][kj], st[i+(1<<(ki-1))][j][ki-1][kj]);
                }
            }
        }
    }
}
int query(int u, int l, int d, int r){
    int k1 = lg[d - u + 1], k2 = lg[r - l + 1];
    return max(max(st[u][l][k1][k2], st[d-(1<<k1)+1][r-(1<<k2)+1][k1][k2]),
               max(st[d-(1<<k1)+1][l][k1][k2], st[u][r-(1<<k2)+1][k1][k2]));
}
```