0x10 基础算法

0x11 逆序对

```
int merge_sort(int q[], int l, int r)
{
    if (l >= r) return 0;

    int mid = l + r >> 1;
    int res = merge_sort(q, l, mid) + merge_sort(q, mid + 1, r);

    int i = l, j = mid + 1, k = 0;
    while (i <= mid && j <= r)
    {
        if (q[i] <= q[j]) tmp[k ++ ] = q[i ++ ];
        else
        {
            res += mid - i + 1;
            tmp[k ++ ] = q[j ++ ];
        }
    }
    while (i <= mid) tmp[k ++ ] = q[i ++ ];
    while (j <= r) tmp[k ++ ] = q[j ++ ];

    for (int i = l, j = 0; i <= r; i ++ , j ++ ) q[i] = tmp[j];

    return res;
}
```

0x12 ST 表

```
template<class T, class Cmp = less<T>>
struct ST
{
    const int n;
    const Cmp cmp;
    vector<vector<T>> f;

    ST(const vector<T> &init) : n(init.size() - 1), cmp(Cmp())
    {
        int k = log2(n);
        f.assign(n + 1, vector<T> (k + 1));
        for (int i = 1; i <= n; i ++ ) f[i][0] = init[i];

        for (int j = 1; j <= k; j ++ )
            for (int i = 1; i <= n - (1 << j) + 1; i ++ )
                f[i][j] = min(f[i][j - 1], f[i + (1 << (j - 1))][j - 1], cmp);
    }
```

```
    T RMQ(int l, int r)
    {
        int k = log2(r - l + 1);
        return min(f[l][k], f[r - (1 << k) + 1][k], cmp);
    }
};
```