*int* a[N];

*int* b[N];

*void* merge(*int* l1, *int* r1, *int* l2, *int* r2) {

*int* i = l1, j = l2;

*int* id = l1;

    ll ans = 0;

*while* (i <= r1 || j <= r2) {

*if* (i == r1 + 1) {

            b[id++] = a[j++];

*continue*;

        }

*if* (j == r2 + 1) {

            b[id++] = a[i++];

*continue* ;

        }

*if* (a[i] <= a[j]) {

            b[id++] = a[i++];

        }

*else* {

*//            cout << l1 << " " << r1 << " " << i << " " << r1 - i + 1 << endl;*

            b[id++] = a[j++];

        }

    }

*for* (*int* i = l1; i <= r2; i++) a[i] = b[i];

*return*;

}

*void* dfs(*int* l, *int* r) {

*if* (l >= r) *return*;

*int* mid = (l + r) / 2;

    dfs(l , mid);

    dfs(mid + 1, r);

    merge(l, mid, mid + 1, r);

*//    cout << l << " " << r << " " << ans << endl;*

*return*;

}

首先是最标准的归并排序的写法，注意这种写法的变量类型，以及mid的计算公式。如果mid超限，可以换成l + (r - l) / 2这种写法。这个归并排序中我们声明了两个全局变量数组a和b数组a是排序前的数组，数组b是排序后的数组。默认是从小到大排序，如果想要从大到小排序改一下关键比较即可。注意merge中对a数组的修改，归并排序的性质要确保合并后的区间是排好序的。

*int* a[N];

*int* b[N];

ll merge(*int* l1, *int* r1, *int* l2, *int* r2) {

*int* i = l1, j = l2;

*int* id = l1;

    ll ans = 0;

*while* (i <= r1 || j <= r2) {

*if* (i == r1 + 1) {

            b[id++] = a[j++];

*continue*;

        }

*if* (j == r2 + 1) {

            b[id++] = a[i++];

*continue* ;

        }

*if* (a[i] <= a[j]) {

            b[id++] = a[i++];

        }

*else* {

*//            cout << l1 << " " << r1 << " " << i << " " << r1 - i + 1 << endl;*

            ans += r1 - i + 1;

            b[id++] = a[j++];

        }

    }

*for* (*int* i = l1; i <= r2; i++) a[i] = b[i];

*return* ans;

}

ll dfs(*int* l, *int* r) {

*if* (l >= r) *return* 0;

*int* mid = (l + r) / 2;

    ll ans = 0;

    ans += dfs(l , mid);

    ans += dfs(mid + 1, r);

    ans += merge(l, mid, mid + 1, r);

*//    cout << l << " " << r << " " << ans << endl;*

*return* ans;

}

这是归并排序求逆序数的写法，注意只需要在dfs中加入ans记录答案，在merge中出现逆序的情况时增加ans即可。原理为当ai > bj时ai后面知道r1的元素也一定大于bj所以都会产生逆序数。