

RMQ(Range Minimum Query)

线段树

设树高 h ，叶子节点 n 个，除去叶子节点后的节点个数为 $n - 1$ 。

```
#include<iostream>
#include<algorithm>
using namespace std;

const int N = 100, INF = 1e9;

int n;
int tree[N * N];

void init(int _n) {
    n = 1;
    while (n < _n) n <= 1; //将叶子层铺满方便计算
    for (int i = 0; i <= n; i++) tree[i] = INF;
}

void update(int k, int a) {
    k += n - 1; //从0开始对节点进行编号 l=2*k+1 ; r=2*k+2;
    tree[k] = a;
    while (k > 0) {
        k = (k - 1) >> 1;
        tree[k] = min(tree[(k << 1) + 1], tree[(k << 1) + 2]);
    }
}

//求[a , b)中的最小值
int query(int a, int b, int k, int l, int r) {
    if (r <= a || l >= b) return INF; //排除不要的区间
    if (a <= l && r <= b) return tree[k];
    else {
        int v1 = query(a, b, (k << 1) + 1, l, (l + r) >> 1);
        int v2 = query(a, b, (k << 1) + 2, (l + r) >> 1, r);
        return min(v1, v2);
    }
}

int main() {
    int a[] = { 5, 3, 7, 9, 6, 4, 1, 2 };
    init(8);
    for (int i = 0; i < 8; i++)
        update(i, a[i]);
    cout << query(2, 5, 0, 0, n); // [3, 6): 数组下标减1

    return 0;
}
```

稀疏表(Sparse Table)

```
#include<iostream>
#include<algorithm>
using namespace std;
```

```

const int N = 405;

int n;
int a[N];
int dp[N][N]; // dp[i][j] 表示 a[j] .. a(j+1<<i) 的最小值

int main() {
    /* input
    **8
    **5 3 7 9 6 4 1 2
    */
    int n;
    cin >> n;
    for (int i = 0; i < n; i++) {
        cin >> dp[0][i];
    }
    // (1 << i) 为区间长度 j 为区间起始
    for (int i = 0; (1 << i) <= n; i++) {
        for (int j = 0; j < n - (1 << i); j++) {
            dp[i + 1][j] = min(dp[i][j], dp[i][j + (1 << i)]);
        }
    }
    cout << endl;
    for (int i = 0; (1 << (i - 1)) < n; i++) {
        for (int j = 0; j < n; j++)
            cout << dp[i][j] << " ";
        cout << endl;
    }
    cout << dp[1][2]; // [2, 2 + 2]
    return 0;
}

```

树状数组(BIT)

能够完成的操作：给定一个初始值全为 0 的数列 a_1, a_2, \dots, a_n

①：给定 i ，计算 $a_1 + a_2 + \dots + a_n$

*从 i 开始将不断将 res 的加上 $bit[i]$ ，并从 i 中将 i 二进制最低非 0 位对应的幂减去。（二进制的最后一个 1，可以通过 $i \& -i$ 得到）

②：给定 i 和 x ，执行 $a_i += x$

*更新 BIT 的值：使第 i 项的值增加 x ，需要从 i 开始将不断将 $bit[i]$ 的加上 x ，并把 i 的二进制最低非 0 位对应的幂加到 i 上。（二进制的最后一个 1，可以通过 $i \& -i$ 得到）

bit 下标从 1 开始

实现

```

#include<iostream>
#include<algorithm>
using namespace std;

const int N = 405;
int bit[N], n;

```

```
int sum(int i) {
    int res = 0;
    while (i > 0) {
        res += bit[i];
        i -= (i & -i);
    }
    return res;
}

void add(int i, int x) {
    while (i <= n) {
        bit[i] += x;
        i += i & (-i);
    }
}

int main() {
    cin >> n;
    for (int i = 1; i <= n; i++) {
        int temp;
        cin >> temp;
        add(i, temp);
    }
    cout << sum(2);
    return 0;
}
```