Standard Code Library

Your TeamName

Your School

August 2, 2025

Contents

一切的开始

宏定义

- 需要 C++11

```cpp
#include <bits/stdc++.h>
using namespace std;
using LL = long long;
#define FOR(i, x, y) for (decay<decltype(y)>::type i = (x), _##i = (y); i < _##i; ++i)
#define FORD(i, x, y) for (decay<decltype(x)>::type i = (x), _##i = (y); i > _##i; --i)
#ifdef DEBUG
#ifndef ONLINE_JUDGE
#define zerol
#endif
#endif
#ifdef zerol
#define dbg(x...) do { cout << "\033[32;1m" << #x << " -> "; err(x); } while (0)
void err() { cout << "\033[39;0m" << endl; }
template<template<typename...> class T, typename t, typename... A>
void err(T<t> a, A... x) { for (auto v: a) cout << v << ' '; err(x...); }
template<typename T, typename... A>
void err(T a, A... x) { cout << a << ' '; err(x...); }
#else
#define dbg(...)
#define err(...)
#endif
// ------------------------------------------------------------------
```

- 调试时添加编译选项 -DDEBUG，提交时注释
- 注意检查判题系统编译选项，修改 `#ifndef ONLINE_JUDGE`
- FOR ++ 循环 FOR(循环变量名称，循环变量起始值，循环变量结束值（不含）)
- FORD --循环
- err() 调试时输出（支持单层迭代）
- dbg() 变色输出变量名和变量值（支持单层迭代）
- 黄色 33, 蓝色 34, 橙色 31

对拍

- Linux

```sh
#!/usr/bin/env bash
g++ -o r main.cpp -O2 -std=c++11
g++ -o std std.cpp -O2 -std=c++11
while true; do
    python gen.py > in
    ./std < in > stdout
    ./r < in > out
    if test $? -ne 0; then
        exit 0
    fi
    if diff stdout out; then
        printf "AC\n"
    else
        printf "GG\n"
        exit 0
    fi
done
```

- Windows

```bat
@echo off
setlocal enabledelayedexpansion

g++ -o r main.cpp -O2 -std=c++11
g++ -o std std.cpp -O2 -std=c++11

:loop
python gen.py > in
if !errorlevel! neq 0 exit /b
```

```

std.exe < in > stdout
if !errorlevel! neq 0 exit /b

r.exe < in > out
if !errorlevel! neq 0 exit /b

fc /b stdout out > nul
if !errorlevel! equ 0 (
    echo AC
) else (
    echo GG
    exit /b
)

goto loop
```

快速编译运行（配合无插件 VSC）

- Linux

```
#!/bin/bash
g++ $1.cpp -o $1 -O2 -std=c++14 -Wall -Dzerol -g
if $? -eq 0; then
    ./$1
fi
```

- Windows

```
@echo off
:: 参数为文件名（不含.cpp后缀）
g++ %1.cpp -o %1 -O2 -std=c++14 -Wall -Dzerol -g
if %errorlevel% equ 0 (
    %1.exe
)
```

数据结构

ST 表

- 一维

```
#define M 10

struct RMQ {
    int f[22][M];
    inline int highbit(int x) { return 31 - __builtin_clz(x); }
    void init(int* v, int n) {
        FOR (i, 0, n) f[0][i] = v[i];
        FOR (x, 1, highbit(n) + 1)
        FOR (i, 0, n - (1 << x) + 1)
        f[x][i] = min(f[x - 1][i], f[x - 1][i + (1 << (x - 1))]);
    }
    int get_min(int l, int r) {
        assert(l <= r);
        int t = highbit(r - l + 1);
        return min(f[t][l], f[t][r - (1 << t) + 1]);
    }
};
```

- 二维

```
#define maxn 10
LL n, m, a[maxn][maxn];

struct RMQ2D{
    int f[maxn][maxn][10][10];
    inline int highbit(int x) { return 31 - __builtin_clz(x); }
```

```
    inline int calc(int x, int y, int xx, int yy, int p, int q) {
        return max(
            max(f[x][y][p][q], f[xx - (1 << p) + 1][yy - (1 << q) + 1][p][q]),
            max(f[xx - (1 << p) + 1][y][p][q], f[x][yy - (1 << q) + 1][p][q])
            );
    }
    void init() {
        FOR (x, 0, highbit(n) + 1)
        FOR (y, 0, highbit(m) + 1)
        FOR (i, 0, n - (1 << x) + 1)
        FOR (j, 0, m - (1 << y) + 1) {
            if (!x && !y) { f[i][j][x][y] = a[i][j]; continue; }
            f[i][j][x][y] = calc(
                i, j,
                i + (1 << x) - 1, j + (1 << y) - 1,
                max(x - 1, 0), max(y - 1, 0)
                );
        }
    }
    inline int get_max(int x, int y, int xx, int yy) {
        return calc(x, y, xx, yy, highbit(xx - x + 1), highbit(yy - y + 1));
    }
};
```

线段树

朴素线段树

- 默认为最大值，可自行修改 `struct Q`struct P`P operator &`
- 注意建树时的下标问题 (1-based)

```
const LL INF = LONG_LONG_MAX;
#define maxn 10
LL n;

namespace SGT {
    struct Q {
        LL setv;
        explicit Q(LL setv = -1): setv(setv) {}
        void operator += (const Q& q) { if (q.setv != -1) setv = q.setv; }
    };
    struct P {
        LL max;
        explicit P(LL max = -INF): max(max) {}
        void up(Q& q) { if (q.setv != -1) max = q.setv; }
    };
    template<typename T>
    P operator & (T&& a, T&& b) {
        return P(max(a.max, b.max));
    }
    P p[maxn << 2];
    Q q[maxn << 2];
#define lson o * 2, l, (l + r) / 2
#define rson o * 2 + 1, (l + r) / 2 + 1, r
    void up(int o, int l, int r) {
        if (l == r) p[o] = P();
        else p[o] = p[o * 2] & p[o * 2 + 1];
        p[o].up(q[o]);
    }
    void down(int o, int l, int r) {
        q[o * 2] += q[o]; q[o * 2 + 1] += q[o];
        q[o] = Q();
        up(lson); up(rson);
    }
    template<typename T>
    void build(T&& f, int o = 1, int l = 1, int r = n) {
        if (l == r) q[o] = f(l);
        else { build(f, lson); build(f, rson); q[o] = Q(); }
        up(o, l, r);
    }
    P query(int ql, int qr, int o = 1, int l = 1, int r = n) {
```

```
        if (ql > r || l > qr) return P();
        if (ql <= l && r <= qr) return p[o];
        down(o, l, r);
        return query(ql, qr, lson) & query(ql, qr, rson);
    }
    void update(int ql, int qr, const Q& v, int o = 1, int l = 1, int r = n) {
        if (ql > r || l > qr) return;
        if (ql <= l && r <= qr) q[o] += v;
        else {
            down(o, l, r);
            update(ql, qr, v, lson); update(ql, qr, v, rson);
        }
        up(o, l, r);
    }
}

// ------------------------------------------------------------------
void solve(){
    vector<LL> arr = {1, 5, 7, 4, 2, 8 ,3, 6, 10, 9};
    n = arr.size();
    SGT::build([&](int idx){
        return SGT::Q(arr[idx-1]);
    });
    for(LL i=1; i<=n; i++){
        dbg(SGT::query(1, i).max);
    }
    SGT::update(2, 4, SGT::Q(-3));
    cout << "MODIFIED\n";
    for(LL i=1; i<=n; i++){
        dbg(SGT::query(1, i).max);
    }
}
```

- 区间修改，区间累加，查询区间和、最大值、最小值。

```
#define maxn 100005
#define INF LONG_LONG_MAX
LL a[maxn], n;

struct IntervalTree {
#define ls o * 2, l, m
#define rs o * 2 + 1, m + 1, r
    static const LL M = maxn * 4, RS = 1E18 - 1;
    LL addv[M], setv[M], minv[M], maxv[M], sumv[M];
    void init() {
        memset(addv, 0, sizeof addv);
        fill(setv, setv + M, RS);
        memset(minv, 0, sizeof minv);
        memset(maxv, 0, sizeof maxv);
        memset(sumv, 0, sizeof sumv);
    }
    void maintain(LL o, LL l, LL r) {
        if (l < r) {
            LL lc = o * 2, rc = o * 2 + 1;
            sumv[o] = sumv[lc] + sumv[rc];
            minv[o] = min(minv[lc], minv[rc]);
            maxv[o] = max(maxv[lc], maxv[rc]);
        } else sumv[o] = minv[o] = maxv[o] = 0;
        if (setv[o] != RS) { minv[o] = maxv[o] = setv[o]; sumv[o] = setv[o] * (r - l + 1); }
        if (addv[o]) { minv[o] += addv[o]; maxv[o] += addv[o]; sumv[o] += addv[o] * (r - l + 1); }
    }
    void build(LL o, LL l, LL r) {
        if (l == r) addv[o] = a[l];
        else {
            LL m = (l + r) / 2;
            build(ls); build(rs);
        }
        maintain(o, l, r);
    }
    void pushdown(LL o) {
        LL lc = o * 2, rc = o * 2 + 1;
        if (setv[o] != RS) {
```

```
            setv[lc] = setv[rc] = setv[o];
            addv[lc] = addv[rc] = 0;
            setv[o] = RS;
        }
        if (addv[o]) {
            addv[lc] += addv[o]; addv[rc] += addv[o];
            addv[o] = 0;
        }
    }
    void update(LL p, LL q, LL o, LL l, LL r, LL v, LL op) {
        if (p <= r && l <= q){
            if (p <= l && r <= q) {
                if (op == 2) { setv[o] = v; addv[o] = 0; }
                else addv[o] += v;
            } else {
                pushdown(o);
                LL m = (l + r) / 2;
                update(p, q, ls, v, op); update(p, q, rs, v, op);
            }
        }
        maintain(o, l, r);
    }
    void query(LL p, LL q, LL o, LL l, LL r, LL add, LL& ssum, LL& smin, LL& smax) {
        if (p > r || l > q) return;
        if (setv[o] != RS) {
            LL v = setv[o] + add + addv[o];
            ssum += v * (min(r, q) - max(l, p) + 1);
            smin = min(smin, v);
            smax = max(smax, v);
        } else if (p <= l && r <= q) {
            ssum += sumv[o] + add * (r - l + 1);
            smin = min(smin, minv[o] + add);
            smax = max(smax, maxv[o] + add);
        } else {
            LL m = (l + r) / 2;
            query(p, q, ls, add + addv[o], ssum, smin, smax);
            query(p, q, rs, add + addv[o], ssum, smin, smax);
        }
    }
    // 简化接口
    void build(int n) {
        build(1, 1, n);
    }

    void range_add(int l, int r, int val) {
        update(l, r, 1, 1, n, val, 1);
    }

    void range_set(int l, int r, int val) {
        update(l, r, 1, 1, n, val, 2);
    }

    void range_query(int l, int r, LL& sum, LL& min_val, LL& max_val) {
        sum = 0;
        min_val = INF;
        max_val = -INF;
        query(l, r, 1, 1, n, 0, sum, min_val, max_val);
    }
} IT;
// --------------------------------------------------------------------
void solve(){
    IT.init();

    n = 5;
    vector<int> data = {1, 3, 5, 7, 9};
    for (int i = 0; i < n; i++) {
        a[i + 1] = data[i];  // 注意: 线段树从 1 开始索引
    }

    IT.build(n);

```

```
    LL sum, min_val, max_val;
    IT.range_query(1, 5, sum, min_val, max_val);
    cout << " " << sum << " " << min_val << " " << max_val << endl;

    IT.range_add(2, 4, 2);
    IT.range_query(1, 5, sum, min_val, max_val);
    cout << " " << sum << " " << min_val << " " << max_val << endl;

    IT.range_set(3, 5, 10);
    IT.range_query(1, 5, sum, min_val, max_val);
    cout << " " << sum << " " << min_val << " " << max_val << endl;

    IT.range_query(2, 4, sum, min_val, max_val);
    cout << " " << sum << " " << min_val << " " << max_val << endl;
}
```

树状数组

- 单点修改，区间查询
- 频次统计下的 k 小值
- 维护差分数组时的区间修改，单点查询

```
#define M 100005

namespace BIT {
    LL c[M]; // 注意初始化开销
    inline int lowbit(int x) { return x & -x; }
    void add(int x, LL v) { // 单点加
        for (int i = x; i < M; i += lowbit(i))
            c[i] += v;
    }
    LL sum(int x) { // 前缀和
        LL ret = 0;
        for (int i = x; i > 0; i -= lowbit(i))
            ret += c[i];
        return ret;
    }
    int kth(LL k) { // 频次统计下从小到大第 k 个，详见应用
        int p = 0;
        for (int lim = 1 << 20; lim; lim /= 2)
            if (p + lim < M && c[p + lim] < k) {
                p += lim;
                k -= c[p];
            }
        return p + 1;
    }
    LL sum(int l, int r) { return sum(r) - sum(l - 1); } // 区间和
    // 区间加（此时树状数组为差分数组，sum(x) 为第 x 个数的值）
    void add(int l, int r, LL v) { add(l, v); add(r + 1, -v); }
}
// ------------------------------------------------------------------------
void solve(){
    vector<LL> a={9, 9, 9, 9, 5, 3, 3, 3, 1, 1};
    LL n = a.size(), i;
    for(i=1; i<=n; i++) BIT::add(a[i-1], 1);
    // 1 1 3 3 3 5 9 9 9 9
    for(i=1; i<=n; i++) cout << BIT::kth(i) << ' ';
}
```

- 区间修改、区间查询

```
#define maxn 100005

namespace BIT {
    int n;
    int c[maxn], cc[maxn];
    inline int lowbit(int x) { return x & -x; }
    void init(int siz){ // 初始化
        n = siz;
        for(LL i=0; i<=n; i++){
            c[i] = cc[i] = 0;
```

```
        }
    }
    void add(int x, int v) { // 不要用这个
        for (int i = x; i <= n; i += lowbit(i)) {
            c[i] += v; cc[i] += x * v;
        }
    }
    void add(int l, int r, int v) { add(l, v); add(r + 1, -v); } // 区间修改
    int sum(int x) { // 前缀和
        int ret = 0;
        for (int i = x; i > 0; i -= lowbit(i))
            ret += (x + 1) * c[i] - cc[i];
        return ret;
    }
    int sum(int l, int r) { return sum(r) - sum(l - 1); } // 区间和
}
// ------------------------------------------------------------------
void solve(){
    LL i, n=8;
    BIT::init(n);
    BIT::add(2, 4, 2);
    for(i=1; i<=n; i++) cout << BIT::sum(i, i) << ' ';
    cout << '\n';
    cout << BIT::sum(5) << '\n';
    cout << BIT::sum(2, 3) << '\n';
}
```

- 三维

```
#define maxn 105

namespace BIT{
    int n;
    LL c[maxn][maxn][maxn];
    inline int lowbit(int x) { return x & -x; }
    void init(int siz){
        n = siz;
        for(int i=0; i<=n; i++){
            for(int j=0; j<=n; j++){
                for(int k=0; k<=n; k++){
                    c[i][j][k] = 0;
                }
            }
        }
    }
    void update(int x, int y, int z, int d) {
        for (int i = x; i <= n; i += lowbit(i))
            for (int j = y; j <= n; j += lowbit(j))
                for (int k = z; k <= n; k += lowbit(k))
                    c[i][j][k] += d;
    }
    LL query(int x, int y, int z) {
        LL ret = 0;
        for (int i = x; i > 0; i -= lowbit(i))
            for (int j = y; j > 0; j -= lowbit(j))
                for (int k = z; k > 0; k -= lowbit(k))
                    ret += c[i][j][k];
        return ret;
    }
    LL solve(int x, int y, int z, int xx, int yy, int zz) {
        return    query(xx, yy, zz)
        - query(xx, yy, z - 1)
        - query(xx, y - 1, zz)
        - query(x - 1, yy, zz)
        + query(xx, y - 1, z - 1)
        + query(x - 1, yy, z - 1)
        + query(x - 1, y - 1, zz)
        - query(x - 1, y - 1, z - 1);
    }
}
```

数学

图论

计算几何

字符串

杂项