

# 综合混练

## 人员

叶乐山、陈瀚霄、张皓宁 到课

## 上周作业检查

<https://vjudge.net/contest/764809>

开始时间 : 2025-11-08 08:30 CST		☆ 🎉 2025-1108 五队上课 (整除分块)						结束时间 : 2026-06-04 16:30 CST					
		进行中											
概览		题目	提交状态	排名 (7:09:10:27)		讨论				设置	克隆	更新	删除
Rank	Team	Score	Penalty	A 10 / 14	B 10 / 10	C 9 / 20	D 8 / 9	E 6 / 6	F 0 / 0				
1	☆ 🎉 lxw123bc (卢新闻)	5	359	0:28:03 (-2)	0:33:39	0:46:35 (-1)	1:19:00	1:52:00					
2	☆ 🎉 123zhw (战鹤文)	5	836	0:24:54	0:35:52	0:50:12 (-1)	1:28:07	10:17:47					
3	☆ 🎉 two_tiger (卢炫佑)	5	2371	7:21:34	7:30:52	7:43:48	7:55:00	8:59:58					
4	☆ 🎉 FEILANYU (于飞岗)	5	2463	7:34:09	7:36:03	7:45:57	8:47:55 (-1)	8:58:58					
5	☆ 🎉 zhn123bc (张皓宁)	5	2464	7:26:29 (-1)	7:33:03	7:44:57 (-1)	8:42:23	8:57:44					
6	☆ 🎉 cxx123bc (曹承贤)	4	1893	7:25:47	7:34:21	7:53:58	8:39:38						
7	☆ 🎉 chx123bc (陈瀚霄)	4	1954	7:35:00 (-1)	7:42:29	7:55:27	9:01:11						
8	☆ 🎉 Terry_MC (叶乐山)	4	1982	7:32:49	7:42:51	8:03:28 (-3)	8:43:15						
9	☆ 🎉 ikunTLE (方冠霖)	4	10995	7:36:22	7:42:59	8:03:17 (-5)		6:14:12:31					
10	☆ 🎉 WangYanzhen (王彦臻)	2	19035	6:14:32:47	6:14:42:41								

## 本周作业

<https://vjudge.net/contest/766697> (课上讲了 A ~ C 这些题, 课后作业是 D 题)

## 课堂表现

今天的第二题是个线段树, 稍微一转化之后就变成一个线段树的板子题了

大部分同学课上都没写出来, 课下一定要着重补一补调一调这个题

## 课堂内容

### P8016 [COCI 2013/2014 #4] ČOKOLADE (上周作业)

每个数最多  $\sqrt{1e8}$  段, 一共  $n$  个数, 总共  $n * \sqrt{1e8}$  种情况

针对每种情况, 可以 O(n) 求是否符合题意

总时间复杂度:  $n * n * \sqrt{1e8}$

```
#include <bits/stdc++.h>

using namespace std;

const int N = 100 + 5, M = 1e8 + 5;
int w[N], ans[N];

int main()
{
    int n, maxx = 0; cin >> n;
    for (int i = 1; i <= n; ++i) cin >> w[i], maxx = max(maxx, w[i]);

    memset(ans, -1, sizeof(ans));
    map<int, int> mp;
    for (int i = 1; i <= maxx+1; ) {
        mp.clear();
        int minn_j = 1e9+10;
        for (int k = 1; k <= n; ++k) {
            int v = w[k] / i;
            if (v) minn_j = min(minn_j, w[k] / v);
            mp[v]++;
        }
        for (auto it : mp) {
            if (ans[it.second] == -1) ans[it.second] = i;
        }
        i = minn_j + 1;
    }

    for (int i = 1; i <= n; ++i) cout << ans[i] << endl;
    return 0;
}
```

### AT\_abc300\_e [ABC300E] Dice Product 3

直接记忆化搜索即可, 时间复杂度其实就是  $n$  的因数级别的

```
#include <bits/stdc++.h>
#define int long long

using namespace std;

const int mod = 998244353;

int qmod(int a, int k) {
    int res = 1;
    while (k) {

```

```

    if (k&1) res = res*a % mod;
    a = a*a % mod;
    k >>= 1;
}
return res;
}
int inv(int x) { return qmod(x, mod-2); }

int inv_5;

map<int, int> mp;
int dfs(int n) {
    if (mp.count(n)) return mp[n];

    int res = 0;
    for (int i = 2; i <= 6; ++i) {
        if (n%i == 0) res += dfs(n/i)*inv_5, res %= mod;
    }
    mp[n] = res;

    return mp[n];
}

signed main()
{
    int n; cin >> n;
    inv_5 = inv(5); mp[1] = 1;
    cout << dfs(n) << endl;
    return 0;
}

```

## AT\_abc371\_f [ABC371F] Takahashi in Narrow Road

先让所有  $w[i]$  减去  $i$ , 这样后续就是让所有  $w[i]$  保持单调不降

如果第  $i$  个数从原值  $v_1$  变为  $v_2$ , 其实就是把一段连续得数全变为  $v_2$ , 这个过程可以用 线段树 来进行维护

```

#include <bits/stdc++.h>
#define int long long

using namespace std;

const int maxn = 2e5 + 5;
struct node {
    int l, r;
    int flag;
    int sum;
} tr[maxn*4];
int w[maxn];

void pushup(int u) { tr[u].sum = tr[u*2].sum + tr[u*2+1].sum; }

```

```
void pushdown(int u) {
    if (tr[u].flag != -1) {
        tr[u*2].flag = tr[u].flag, tr[u*2+1].flag = tr[u].flag;
        tr[u*2].sum = tr[u*2].flag * (tr[u*2].r - tr[u*2].l + 1);
        tr[u*2+1].sum = tr[u*2+1].flag * (tr[u*2+1].r - tr[u*2+1].l + 1);
        tr[u].flag = -1;
    }
}

void build(int u, int l, int r) {
    tr[u] = {l, r, -1, 0};
    if (l != r) {
        int mid = (l + r) / 2;
        build(u*2, l, mid), build(u*2+1, mid+1, r);
    }
}

void modify(int u, int l, int r, int k) {
    if (tr[u].l>=l && tr[u].r<=r) {
        tr[u].flag = k, tr[u].sum = k * (tr[u].r - tr[u].l + 1);
        return;
    }

    pushdown(u);
    int mid = (tr[u].l + tr[u].r) / 2;
    if (l <= mid) modify(u*2, l, r, k);
    if (r > mid) modify(u*2+1, l, r, k);
    pushup(u);
}

int query(int u, int l, int r) {
    if (tr[u].l>=l && tr[u].r<=r) return tr[u].sum;

    pushdown(u);
    int mid = (tr[u].l + tr[u].r) / 2, res = 0;
    if (l <= mid) res += query(u*2, l, r);
    if (r > mid) res += query(u*2+1, l, r);
    return res;
}

int find_pos(int n, int k) { // 返回最后一个 <=k 的位置
    int l = 1, r = n;
    while (l <= r) {
        int mid = (l + r) / 2;
        if (query(1, mid, mid) <= k) l = mid+1;
        else r = mid-1;
    }
    return r;
}

signed main()
{
    int n; cin >> n; build(1, 1, n);
```

```

for (int i = 1; i <= n; ++i) cin >> w[i], modify(1, i, i, w[i]-i);

int T; cin >> T;
int res = 0;
while (T -- ) {
    int last = query(1, 1, n);
    int id, pos; cin >> id >> pos;

    int id2 = find_pos(n, pos-id);
    if (id <= id2) modify(1, id, id2, pos-id);
    else modify(1, id2+1, id, pos-id);

    int now = query(1, 1, n);
    res += abs(now - last);
}
cout << res << endl;
return 0;
}

```

## AT\_abc375\_f [ABC375F] Road Blocked

考虑删边操作是比较复杂的, 因为删完边后点与点之间的最短路变化可能变化很大

因此, 可以把整个问题反过来思考, 考虑加边操作

每次加完一条边后, 最短路的更新只可能通过这条边进行更新, 所以最短路的更新是  $O(n^2)$  级别的

题目保证, 最多添加 300 条边, 那么这个题就解决了

```

#include <bits/stdc++.h>
#define int long long

using namespace std;

const int N = 300 + 5, M = 2e5 + 5;
const int inf = 0x3f3f3f3f3f3f3f3f;
struct Edge {
    int a, b, c;
} w[N*N];
bool st[N*N];
int f[N][N];

struct node {
    int op, id, x, y;
} q[M];

signed main()
{
    int n, m, Q; cin >> n >> m >> Q;
    for (int i = 1; i <= m; ++i) cin >> w[i].a >> w[i].b >> w[i].c, st[i] = true;
    for (int i = 1; i <= Q; ++i) {
        cin >> q[i].op;
    }
}

```

```
if (q[i].op == 1) cin >> q[i].id, st[q[i].id] = false;
else cin >> q[i].x >> q[i].y;
}

memset(f, 0x3f, sizeof(f));
for (int i = 1; i <= n; ++i) f[i][i] = 0;
for (int i = 1; i <= m; ++i) {
    int a = w[i].a, b = w[i].b, c = w[i].c;
    if (st[i]) f[a][b] = f[b][a] = c;
}

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

vector<int> ans;
for (int i = Q; i >= 1; --i) {
    int op = q[i].op, id = q[i].id, x = q[i].x, y = q[i].y;
    if (op == 1) {
        int a = w[id].a, b = w[id].b, c = w[id].c;
        f[a][b] = min(f[a][b], c);
        for (int i = 1; i <= n; ++i) {
            for (int j = 1; j <= n; ++j) {
                int value = min(f[i][a]+f[b][j], f[i][b]+f[a][j]) + f[a][b];
                f[i][j] = min(f[i][j], value);
            }
        }
    } else {
        int dis = (f[x][y]==inf ? -1 : f[x][y]);
        ans.push_back(dis);
    }
}

reverse(ans.begin(), ans.end());
for (int i : ans) cout << i << endl;
return 0;
}
```