模板

差分

```
/*
题目大意：
    第一行n，m
    第二行n个整数
    接下来m行，每行三个正整数l，r，c
    表示对区间[l, r]每个数加c
    输出操作后的数列
输入样例：
6 3
1 2 2 1 2 1
1 3 1
3 5 1
1 6 1
输出样例：
3 4 5 3 4 2
*/
#include <iostream>
using namespace std;
const int N = 100010;
int s[N], a[N];

void insert(int l, int r, int c)
{
```

```
    a[l] += c;
    a[r+1] -= c;

}

int main()
{
    int n, m;
    cin >> n >> m;
    for(int i=1;i<=n;i++) scanf("%d", &s[i]), insert(i, i, s[i]);

    while(m--)
    {
        int l, r, c;
        scanf("%d %d %d", &l, &r, &c);
        insert(l, r, c);
    }

    for(int i=1;i<=n;i++) a[i] += a[i-1];

    for(int i=1;i<=n;i++) printf("%d ", a[i]);
    return 0;
}

/*
题目大意：
    第一行n，m，q
    接下来n行m列矩阵
    接下来q行x1,y1, x2,y2, c
    表示对子矩阵每个数加c，输出操作后的矩阵
输入样例：
3 4 3
1 2 2 1
3 2 2 1
1 1 1 1
1 1 2 2 1
1 3 2 3 2
3 1 3 4 1
输出样例：
2 3 4 1
4 3 4 1
2 2 2 2
*/
#include <iostream>
using namespace std;
const int N = 1010;

int n, m, q;
int a[N][N], b[N][N];

void insert(int x1, int y1, int x2, int y2, int c)
{
    b[x1][y1] += c;
    b[x2+1][y1] -= c;
```

```
    b[x1][y2+1] -= c;
    b[x2+1][y2+1] += c;
}
int main()
{
    scanf("%d %d %d", &n, &m, &q);

    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= m; j++)
            scanf("%d", &a[i][j]), insert(i, j, i, j, a[i][j]);

    while(q--)
    {
        int x1, x2, y1, y2, c;
        scanf("%d %d %d %d %d", &x1, &y1, &x2, &y2, &c);
        insert(x1, y1, x2, y2, c);
    }
    for(int i=1;i<=n;i++)
        for(int j=1;j<=m;j++)
            b[i][j] += b[i-1][j] + b[i][j-1] - b[i-1][j-1];
    for(int i=1;i<=n;i++)
    {
        for(int j=1;j<=m;j++) printf("%d ", b[i][j]);
        puts("\b");
    }
    return 0;
}
```

KMP

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

const int N = 1e5+10, M = 1e6+10;
int n, m;
char p[N], s[M];
int ne[N];

int main()
{
    cin >> n >> p + 1 >> m >> s + 1;

    for(int i=2, j=0;i<=n;i++)
    {
        while(j&&p[i]!=p[j+1]) j = ne[j];
        if(p[i] == p[j+1]) j ++;
        ne[i] = j;
    }

    for(int i=1, j=0;i<=m;i++)
    {
```

```
        while(j&&s[i]!=p[j+1]) j = ne[j];
        if(s[i]==p[j+1]) j++;
        if(j==n)
        {
            printf("%d ", i - n);
            j = ne[j];
        }
    }
    return 0;
}
```

tire

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

const int N = 1e5+10;

int son[N][26], cnt[N], idx;
char str[N];

void insert(char str[])
{
    int p = 0;
    for(int i=0;str[i];i++)
    {
        int u = str[i] - 'a';
        if(!son[p][u]) son[p][u] = ++ idx;
        p = son[p][u];
    }
    cnt[p] ++;
}
int query(char str[])
{
    int p = 0;
    for(int i=0;str[i];i++)
    {
        int u = str[i] - 'a';
        if(!son[p][u]) return 0;
        p = son[p][u];
    }
    return cnt[p];

}


int main()
{
    int n;
    cin >> n;
    while(n--)
```

```
    {
        char op[2];
        scanf("%s%s", op, str);
        if(op[0]=='I') insert(str);
        else printf("%d\n", query(str));
    }
    return 0;
}
```

hash

```
//拉链法
#include <iostream>
#include <cstring>
using namespace std;
const int N = 1e5+3;

int h[N], e[N], ne[N], idx;

void insert(int x)
{
    int k = (x % N + N) % N;
    e[idx] = x;
    ne[idx] = h[k];
    h[k] = idx ++;

}

bool find(int x)
{
    int k = (x % N + N) % N;
    for(int i = h[k]; i != -1; i = ne[i])
        if (e[i] == x)
            return true;
    return false;

}

int main()
{
    int n;
    cin >> n;
    memset(h, -1, sizeof(h));
    while (n--)
    {
        string s;
        cin >> s;
        int x;
        cin >> x;
        if (s[0] == 'I') insert(x);
        else
```

```
        {
            if(find(x)) puts("Yes");
            else puts("No");
        }
    }

    return 0;
}
//开放寻址法
#include <iostream>
#include <cstring>
using namespace std;
const int N = 2e5+3, null = 0x3f3f3f3f;
int h[N];

int find(int x)
{
    int k = (x % N + N) % N;
    while (h[k] != null && h[k] != x)
    {
        k ++;
        if(k == N) k = 0;
    }
    return k;
}

int main()
{
    int n;
    cin >> n;
    memset(h, 0x3f, sizeof(h));
    while (n--)
    {
        string s;
        cin >> s;
        int x;
        cin >> x;

        int k = find(x);
        if(s[0] == 'I') h[k] = x;
        else
        {
            if (h[k] != null) puts("Yes");
            else puts("No");
        }
    }

    return 0;
}

//字符串哈希
#include <iostream>
using namespace std;
typedef unsigned long long ll;
```

```
const int N = 1e5+10, P = 131;

int n, m;
char str[N];
ll h[N], p[N];
ll get(int l, int r)
{
    return h[r] - h[l-1] * p[r-l+1];
}

int main()
{
    cin >> n >> m >> str + 1;
    p[0] = 1;
    for (int i = 1; i <= n; i++)
    {
        p[i] = p[i - 1] * P;
        h[i] = h[i - 1] * P + str[i];
    }
    while (m--)
    {
        int l1, r1, l2, r2;
        cin >> l1 >> r1 >> l2 >> r2;

        if(get(l1, r1) == get(l2, r2)) puts("Yes");
        else puts("No");
    }


    return 0;
}
```

最短路

```
//dijkstra
//O(n^2)
//n = 500, m = 1e5, 适合稠密图，无负权边
#include <iostream>
#include <cstring>
#include <algorithm>
using namespace std;
const int N = 510;
int n, m, g[N][N], dist[N];
bool st[N];

int dijkstra()
{
    memset(dist, 0x3f, sizeof(dist));
    dist[1] = 0;
    for (int i = 0; i < n; i++)
    {
```

```
        int t = -1;
        for (int j = 1; j <= n; j ++)
            if(!st[j] && (t == -1 || dist[t] > dist[j]))
                t = j;
        //if(t == n) break;
        st[t] = true;
        for (int j = 1; j<= n; j++)
            dist[j] = min(dist[j], dist[t]+g[t][j]);
    }
    if(dist[n] == 0x3f3f3f3f) return -1;
    return dist[n];
}
int main()
{
    scanf("%d %d", &n, &m);
    memset(g, 0x3f, sizeof(g));
    while (m--)
    {
        int a, b, c;
        scanf("%d %d %d", &a, &b, &c);
        g[a][b] = min(g[a][b], c);
    }
    int t = dijkstra();
    printf("%d\n", t);
    return 0;
}
//dijkstra
//O(mlogn)
//n, m = 1e5 , 适合稀疏图
#include <iostream>
#include <cstring>
#include <algorithm>
#include <queue>
using namespace std;
const int N = 1e5+10;
int n, m, h[N], e[N], ne[N], dist[N], w[N], idx;
bool st[N];
typedef pair<int, int> PII;
void add(int a, int b, int c)
{
    e[idx] = b;
    ne[idx] = h[a];
    w[idx] = c;
    h[a] = idx ++;
}

int dijkstra()
{
    memset(dist, 0x3f, sizeof(dist));
    dist[1] = 0;
    priority_queue<PII, vector<PII>, greater<PII>> heap;
    heap.push({0, 1});
    while(heap.size())
    {
```

```
        auto t = heap.top();
        heap.pop();
        int ver = t.second, distance = t.first;
        if(st[ver]) continue;
        for (int i=h[ver];i!=-1;i=ne[i])
        {
            int j = e[i];
            if(dist[j] >  distance + w[i])
            {
                dist[j] = distance + w[i];
                heap.push({dist[j], j});
            }
        }
        st[ver] = true;
    }
    if(dist[n] == 0x3f3f3f3f) return -1;
    return dist[n];
}
int main()
{
    scanf("%d %d", &n, &m);
    memset(h, -1, sizeof(h));
    while (m--)
    {
        int a, b, c;
        scanf("%d %d %d", &a, &b, &c);
        add(a, b, c);
    }
    int t = dijkstra();
    printf("%d\n", t);
    return 0;
}
//bellman_ford
//O(nm)
//可以处理负权边
#include <iostream>
#include <cstring>
#include <algorithm>
using namespace std;
const int N = 510, M = 10010;
int n, m, k;
int dist[N], backup[N];
struct Edge
{
    int a, b, w;
}edges[N];
int bellman_ford()
{
    memset(dist, 0x3f, sizeof(dist));
    dist[1] = 0;
    for (int i = 0; i < k; i++)
    {
        memcpy(backup, dist, sizeof(dist));
        for (int j = 0; j < m; j++)
```

```
        {
            int a = edges[j].a, b = edges[j].b, w = edges[j].w;
            dist[b] = min(dist[b], backup[a] + w);
        }
    }
    if (dist[n] > 0x3f3f3f3f / 2) return -1;
    return dist[n];

}
int main()
{
    scanf("%d %d %d", &n, &m, &k);
    for (int i = 0; i < m; i++)
    {
        int a, b, w;
        scanf("%d %d %d", &a, &b, &w);
        edges[i] = {a, b, w};
    }
    int t = bellman_ford();
    if (t == -1) puts("impossible");
    else printf("%d\n", t);

    return 0;
}
//spfa
//O(m)最坏O(nm)
#include <iostream>
#include <cstring>
#include <algorithm>
#include <queue>
using namespace std;
const int N = 1e5+10;
int n, m, h[N], e[N], ne[N], dist[N], w[N], idx;
bool st[N];
typedef pair<int, int> PII;
void add(int a, int b, int c)
{
    e[idx] = b;
    ne[idx] = h[a];
    w[idx] = c;
    h[a] = idx ++;
}

int spfa()
{
    memset(dist, 0x3f, sizeof(dist));
    dist[1] = 0;
    queue<int> q;
    q.push(1);
    st[1] = true;
    while (q.size())
    {
        int t = q.front();
        q.pop();
```

```
            st[t] = false;
            for (int i = h[t]; i != -1; i = ne[i])
            {
                int j = e[i];
                if (dist[j] > dist[t] + w[i])
                {
                    dist[j] = dist[t] + w[i];
                    if (!st[j])
                    {
                        q.push(j);
                        st[j] = true;
                    }
                }
            }
        }
        if (dist[n] == 0x3f3f3f3f) return -1;
        return dist[n];

    }
    int main()
    {
        scanf("%d %d", &n, &m);
        memset(h, -1, sizeof(h));
        while (m--)
        {
            int a, b, c;
            scanf("%d %d %d", &a, &b, &c);
            add(a, b, c);
        }
        int t = spfa();
        if (t == -1) puts("impossible");
        else printf("%d\n", t);
        return 0;
    }

    //spfa判负环
    //O(m)最坏O(nm)
    #include <iostream>
    #include <cstring>
    #include <algorithm>
    #include <queue>
    using namespace std;
    const int N = 1e5+10;
    int n, m, h[N], e[N], ne[N], dist[N], w[N], idx;
    int cnt[N];
    bool st[N];
    typedef pair<int, int> PII;
    void add(int a, int b, int c)
    {
        e[idx] = b;
        ne[idx] = h[a];
        w[idx] = c;
        h[a] = idx ++;
    }
```

```
bool spfa()
{


    queue<int> q;
    for(int i=1;i<=n;i++)
    {
        st[i] = true;
        q.push(i);
    }
    st[1] = true;
    while (q.size())
    {
        int t = q.front();
        q.pop();
        st[t] = false;
        for (int i = h[t]; i != -1; i = ne[i])
        {
            int j = e[i];
            if (dist[j] > dist[t] + w[i])
            {
                dist[j] = dist[t] + w[i];
                cnt[j] = cnt[t] + 1;

                if(cnt[j]>=n) return true;
                if (!st[j])
                {
                    q.push(j);
                    st[j] = true;
                }
            }
        }
    }
    return false;

}
int main()
{
    scanf("%d %d", &n, &m);
    memset(h, -1, sizeof(h));
    while (m--)
    {
        int a, b, c;
        scanf("%d %d %d", &a, &b, &c);
        add(a, b, c);
    }
    if(spfa()) puts("Yes");
    else puts("No");
    return 0;
}

//floyd
//O(n^3)
```

```cpp
#include <iostream>
using namespace std;
const int N = 210, INF = 1e9;
int n, m, Q;
int d[N][N];

void floyd()
{
    for(int k=1;k<=n;k++)
        for(int i=1;i<=n;i++)
            for(int j=1;j<=n;j++)
                d[i][j] = min(d[i][j], d[i][k]+d[k][j]);

}
int main()
{
    scanf("%d %d %d", &n, &m, &Q);

    for(int i=1;i<=n;i++)
        for(int j=1;j<=n;j++)
            if(i == j) d[i][j] = 0;
            else d[i][j] = INF;

    while (m--)
    {
        int a, b, w;
        scanf("%d %d %d", &a, &b, &w);

        d[a][b] = min(d[a][b], w);
    }
    floyd();

    while (Q--)
    {
        int a, b;
        scanf("%d %d", &a, &b);

        if(d[a][b] > INF / 2) puts("impossible");
        else printf("%d\n", d[a][b]);
    }
    return 0;

    return 0;
}

//prim
//O(n^2)
#define inf 0x3f3f3f3f
#include <iostream>
#include <cstring>
#include <algorithm>
using namespace std;
const int N = 510;
int n, m;
```

```
int g[N][N];
int dist[N];
bool st[N];

int prim()
{
    memset(dist, 0x3f, sizeof(dist));
    int res = 0;
    dist[1] = 0;
    for (int i = 0; i < n; i++)
    {
        int t = -1;
        for(int j=1;j<=n;j++)
            if(!st[j] && (t == -1 || dist[t] > dist[j]))
                t = j;
        if(dist[t] == inf) return inf;
        res += dist[t];
        for(int j=1;j<=n;j++) dist[j] = min(dist[j], g[t][j]);


        st[t] = true;

    }
    return res;
}

int main()
{
    scanf("%d %d", &n, &m);
    memset(g, 0x3f, sizeof(g));
    while (m--)
    {
        int a, b, c;
        scanf("%d %d %d", &a, &b, &c);
        g[a][b] = g[b][a] = min(g[a][b], c);
    }
    int t = prim();
    if(t == inf) puts("impossible");
    else cout << t << endl;
    return 0;
}

//kruskal
//O(mlogm)
#include <iostream>
#include <algorithm>
using namespace std;
const int N = 2e5+10;
int n, m;
int p[N];
int cnt, res;
struct Edge
{
    int a, b, w;
```

```
    bool operator< (const Edge &W) const
    {
        return w < W.w;
    }
}edges[N];
int find(int x)
{
    if(p[x] != x) p[x] = find(p[x]);
    return p[x];
}
int main()
{
    scanf("%d %d", &n, &m);
    for (int i = 0; i < m; i++)
    {
        int a, b, c;
        scanf("%d %d %d", &a, &b, &c);
        edges[i] = {a, b, c};
    }
    sort(edges, edges+m);
    for(int i=1;i<=n;i++) p[i] = i;
    for (int i = 0; i < m; i++)
    {
        int a = edges[i].a, b = edges[i].b, w = edges[i].w;
        a = find(a), b = find(b);
        if(a != b)
        {
            p[a] = b;
            res += w;
            cnt ++;
        }
    }
    if(cnt < n-1)puts("impossible");
    else cout << res << endl;
    return 0;
}
```

二分图

```
//O(n+m)
#include <iostream>
#include <cstring>
#include <algorithm>
using namespace std;
const int N = 1e5+10, M = 2e5+10;
int n, m;
int h[N], e[M], ne[M], idx;
int color[N];
```

```
void add(int a, int b)
{
    e[idx] = b, ne[idx] = h[a], h[a] = idx++;
}

bool dfs(int u, int c)
{
    color[u] = c;
    for(int i=h[u];~i;i=ne[i])
    {
        int j = e[i];
        if(!color[j])
        {
            if(!dfs(j, 3-c)) return false;
        }
        else if(color[j] == c) return false;
    }
    return true;
}

int main()
{
    scanf("%d %d", &n, &m);
    memset(h, -1, sizeof(h));

    while (m--)
    {
        int a, b;
        scanf("%d %d", &a, &b);
        add(a, b), add(b, a);
    }
    bool flag = true;
    for(int i=1;i<=n;i++)
    {
        if(!color[i])
        {
            if(!dfs(i, 1))
            {
                flag = false;
                break;
            }
        }
    }
    if(flag) puts("Yes");
    else puts("No");
    return 0;
}

//匈牙利算法
//O(mn)
#include <iostream>
#include <cstring>
using namespace std;
const int N = 510, M = 1e5+10;
```

```
int n1, n2, m;
int h[N], e[M], ne[M], idx;
int match[N];
bool st[N];

void add(int a, int b)
{
    e[idx] = b, ne[idx] = h[a], h[a] = idx ++;
}

bool find(int x)
{
    for(int i=h[x];~i;i=ne[i])
    {
        int j = e[i];
        if(!st[j])
        {
            st[j] = true;
            if(match[j] == 0 || find(match[j]))
            {
                match[j] = x;
                return true;
            }
        }
    }
    return false;
}

int main()
{
    scanf("%d %d %d", &n1, &n2, &m);
    memset(h, -1, sizeof(h));
    while (m--)
    {
        int a, b;
        scanf("%d %d", &a, &b);
        add(a, b);
    }
    int res = 0;
    for(int i=1;i<=n1;i++)
    {
        memset(st, false, sizeof st);
        if(find(i)) res ++;
    }
    cout << res << endl;

    return 0;
}
```

manacher

```
//acwing 3188
#include <iostream>
#include <cstring>
using namespace std;
const int N = 3e7+10;
char str[N], tmp[N/2];
int ne[N];
int main()
{
    scanf("%s", tmp);
    str[0] = '$', str[1] = '#';
    int length1 = 2;
    for(int i=0;tmp[i];i++)
    {
        str[length1++] = tmp[i];
        str[length1++] = '#';
    }
    str[length1++] = '^';
    int mr = 0, ml = 0;
    for(int i=1;str[i]!='^';i++)
    {
        if(i < mr) ne[i] = min(ne[2*ml-i], mr - i);
        else ne[i] = 1;
        while(str[i+ne[i]]==str[i-ne[i]]) ne[i] ++;
        if(ne[i]+i > mr)
        {
            mr = ne[i] + i;
            ml = i;
        }
    }
    int res = 0;
    for(int i=1;i<=length1;i++)
    {
        res = max(res, ne[i]);
    }
    printf("%d", res-1);

    return 0;
}
```

数学

质数筛

```
//线性筛
#include <iostream>
using namespace std;
const int N=1000010;

int cnt,primes[N];
```

```
bool st[N];

void get_prime(int n)
{
    for(int i=2;i<=n;i++)
    {
        if(!st[i])
        {
            primes[cnt++]=i;
        }
        for(int j=0;primes[j]<=n/i;j++)
        {
            st[primes[j]*i]=true;
            if(i% primes[j]==0) break;
        }
    }
}

int main()
{
    int n;
    cin>>n;

    get_prime(n);

    cout<<cnt<<endl;
    return 0;
}
```

约数个数

$$N = p_1^{\alpha _1}p_2^{\alpha _2}\cdots p_k^{\alpha _k}$$ N的约数个数 $$(\alpha _1+1)(\alpha_2+2)\cdots(\alpha_k+1) $$

约数之和

$$N = p_1^{\alpha _1}p_2^{\alpha _2}\cdots p_k^{\alpha _k}$$
$$(p_1^0+p_1^1+\cdots+p_1^{\alpha_1})\cdots(p_k^0+p_k^1+\cdots+p_k^{\alpha_k}) $$

欧拉函数

```
#include <iostream>

using namespace std;
#define ll long long

const int mod=1e9+7;

int main()
{
    int t;
```

```
    cin>>t;
    while(t--)
    {
        int n;
        cin>>n;
        int res=n;
        for(int i=2;i<=n/i;i++)
        {
            if(n%i==0)
            {
                res=res/i*(i-1);
                while(n%i==0) n/=i;
            }
        }
        if(n>1) res=res/n*(n-1);

        cout<<res<<endl;
    }
    return 0;
}

//欧拉函数定义：对于整数n，在1~n-1中与n互质的数的个数
#include <iostream>
using namespace std;

#define ll long long
const int N=1000010;

int cnt,primes[N];
int phi[N];
bool st[N];

ll get_eulers(int n)
{
    phi[1]=1;
    for(int i=2;i<=n;i++)
    {
        if(!st[i])
        {
            primes[cnt++]=i;
            phi[i]=i-1;//质数p的欧拉函数为p-1
        }
        for(int j=0;primes[j]<=n/i;j++)
        {
            st[primes[j]*i]=true;
            if(i% primes[j]==0)
            {
                phi[i*primes[j]]=primes[j]*phi[i];
                break;
            }
            phi[i*primes[j]]=phi[i]*(primes[j]-1);
        }
    }
    ll res=0;
```

```
    for(int i=1;i<=n;i++) res+=phi[i];
    return res;
}

int main()
{
    int n;
    cin>>n;

    cout<<get_eulers(n)<<endl;

    return 0;
}
```

快速幂

```
#include <iostream>

using namespace std;

#define ll long long

ll quick_pow(ll a,ll k,ll p)
{
    ll res=1%p;
    while(k)
    {
        if(k&1) res=res*a%p;
        k>>=1;
        a=a*a%p;
    }
    return res;
}

int main()
{
    int t;
    cin>>t;
    while(t--)
    {
        ll a,b,c;
        cin>>a>>b>>c;
        cout<<quick_pow(a,b,c)<<endl;
    }
    return 0;
}
```

高斯消元

```
#include <cmath>
#include <iostream>
#include <algorithm>
using namespace std;
const int N = 110;
const double eps = 1e-6;
int n;
double a[N][N];
int gauss()
{
    int c, r;
    for(c=0, r=0;c<n;c++)
    {
        int t = r;
        for (int i=r;i<n;i++)
            if(fabs(a[i][c]) > fabs(a[t][c]))
                t = i;//找到绝对值最大的一列

        if(fabs(a[t][c]) < eps) continue;//是零就下一列

        for(int i=c;i<=n;i++) swap(a[t][i], a[r][i]);//交换绝对值最大的行与当前
行
        for(int i=n;i>=c;i--) a[r][i] /= a[r][c];//当前行第一个数变成1
        for(int i=r+1;i<n;i++)
            if(fabs(a[i][c]) > eps)
                for(int j=n;j>=c;j--)
                    a[i][j] -= a[r][j] * a[i][c];//把c下面每个数变成0，就是减去
当前行的a[i][c]倍

        r++;
    }
    if(r < n)
    {
        for(int i=r;i<n;i++)
            if(fabs(a[i][n]) > eps)
                return 2;//无解
        return 1; //无穷多解
    }
    for(int i=n-1;i>=0;i--)
        for(int j=i+1;j<n;j++)
            a[i][n] -= a[i][j] * a[j][n];//算出答案
    return 0;//唯一解
}
int main()
{
    cin >> n;
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n+1; j++)
            cin >> a[i][j];

    int t = gauss();
    if(t == 0) for (int i=0;i<n;i++)
    {
```

```
            double ans = a[i][n];
            if(fabs(ans) < eps) ans = 0;
            printf("%.2f\n", ans);
        }
        else if(t == 1) puts("Infinite group solutions");
        else puts("No solution");

        return 0;
    }
```

st表，树状数组与线段树

st表

```
#include <iostream>
#include <cmath>
#define lowbit(x) (x & -x)
using namespace std;
const int N = 1e5+10;
int f[N][25], g[N][25];
int lg[N];
int n, q;
int main()
{
    lg[1] = 0, lg[2] = 1;
    for(int i=3;i<N;i++) if(lowbit(i) == i) lg[i] = lg[i-1] + 1; else lg[i]
= lg[i-1];
    cin >> n >> q;
    for(int i=1;i<=n;i++)
    {
        int x;
        scanf("%d", &x);
        f[i][0] = g[i][0] = x;
    }
    int ed = 25;
    for(int j=1;j<=ed;j++)
        for(int i=1;i+(1<<j)-1 <= n;i++)
        {
            f[i][j] = max(f[i][j-1], f[i+(1<<(j-1))][j-1]);

            g[i][j] = min(g[i][j-1], g[i+(1<<(j-1))][j-1]);
        }

    while (q--)
    {
        int a, b;
        scanf("%d %d", &a, &b);
        int s = lg[b-a+1];
        printf("%d\n", max(f[a][s], f[b-(1<<s)+1][s]) - min(g[a][s], g[b-
(1<<s)+1][s]));
    }
```

```
    return 0;
}
```

树状数组

```
#include <iostream>
#include <cstring>
#define lowbit(x) (x & -x)
using namespace std;
const int N = 1e5+10;
int tr[N];
int n;
void add(int x, int c)
{
    for(int i=x;i<=n;i+=lowbit(i)) tr[i] += c;
}
char str[20];
int sum(int x)
{
    int res = 0;
    for(int i=x;i;i-=lowbit(i)) res += tr[i];
    return res;
}
int main()
{
    int t, cnt = 0;
    cin >> t;
    while (t--)
    {
        cnt ++;
        printf("Case %d:\n", cnt);
        memset(tr, 0, sizeof(tr));
        scanf("%d", &n);
        for(int i=1;i<=n;i++)
        {
            int x;
            scanf("%d", &x);
            add(i, x);
        }
        while(1)
        {
            scanf("%s", str);
            char op = str[0];
            if(op == 'E') break;
            int i, j;
            scanf("%d %d", &i, &j);
            if(op == 'A') add(i, j);
            else if(op == 'S') add(i, -j);
            else printf("%d\n", sum(j)-sum(i-1));
        }
```

```
    }

    return 0;
}
```

线段树

```
#include <iostream>
using namespace std;
const int N = 2e5+10;
int m, p;
struct Node
{
    int l, r;
    int v;
}tr[N*4];

void pushup(int u)
{
    tr[u].v = max(tr[u<<1].v, tr[u<<1|1].v);
}

void build(int u, int l, int r)
{
    tr[u] = {l, r};
    if(l == r) return;
    int mid = l + r >> 1;
    build(u<<1, l, mid), build(u<<1|1, mid+1, r);
}
int query(int u, int l, int r)
{
    if(tr[u].l >= l && tr[u].r <= r) return tr[u].v;
    int mid = tr[u].l + tr[u].r >> 1;
    int v = 0;
    if(l <= mid) v = query(u<<1, l, r);
    if(r > mid) v = max(v, query(u<<1|1, l, r));


    return v;
}
void modify(int u, int x, int v)
{
    if(tr[u].l == x && tr[u].r == x) tr[u].v = v;
    else
    {
        int mid = tr[u].l + tr[u].r >> 1;
        if(x <= mid) modify(u<<1, x, v);
        else modify(u<<1|1, x, v);
        pushup(u);
    }
}
```

```
int main()
{
    int n = 0, last = 0;
    scanf("%d %d", &m, &p);
    build(1, 1, m);

    int x;
    char op[2];
    while (m--)
    {
        scanf("%s %d", op, &x);
        if(*op == 'Q')
        {
            last = query(1, n-x+1, n);
            printf("%d\n", last);
        }
        else
        {
            modify(1, n+1, (last+x)%p);
            n ++;
        }
    }

    return 0;
}

//AcWing 243
#include <iostream>
#include <cstdio>
#include <algorithm>
#include <iostream>
#define int long long
using namespace std;
const int N = 1e5+10;
int n, m;
int w[N];
struct Node
{
    int l, r;
    int sum, add;
}tr[N * 4];
void pushdown(int u)
{
    auto &root = tr[u], &left = tr[u << 1], &right = tr[u << 1 | 1];
    if(root.add)
    {
        left.add += root.add, left.sum += (left.r - left.l + 1) * root.add;
        right.add += root.add, right.sum += (right.r - right.l + 1) *
root.add;
        root.add = 0;
    }
}
void pushup(int u)
{
```

```
    tr[u].sum = tr[u << 1].sum + tr[u << 1 | 1].sum;
}
void build(int u, int l, int r)
{
    if(l == r) tr[u] = {l, r, w[r], 0};
    else
    {
        tr[u] = {l, r};
        int mid = l + r >> 1;
        build(u << 1, l, mid), build(u << 1 | 1, mid + 1, r);
        pushup(u);
    }
}
void modify(int u, int l, int r, int d)
{
    if(tr[u].l >= l && tr[u].r <= r)
    {
        tr[u].sum += (tr[u].r - tr[u].l + 1) * d;
        tr[u].add += d;
    }
    else
    {
        pushdown(u);
        int mid = tr[u].l + tr[u].r >> 1;
        if(l <= mid) modify(u << 1, l, r, d);
        if(r > mid) modify(u << 1 | 1, l, r, d);
        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 >> 1;
    int sum = 0;
    if (l <= mid) sum = query(u << 1, l, r);
    if (r > mid) sum += query(u << 1 | 1, l, r);
    return sum;
}
signed main()
{
    scanf("%lld %lld", &n, &m);
    for(int i=1;i<=n;i++) scanf("%lld", &w[i]);
    build(1, 1, n);
    char op[2];
    int l, r, d;
    while (m--)
    {
        scanf("%s %lld %lld", op, &l, &r);
        if(*op == 'C')
        {
            scanf("%lld", &d);
            modify(1, l, r, d);
        }
```

```
            else printf("%lld\n", query(1, l, r));
        }

        return 0;
    }
```