General Techniques

// Line Sweep

signed main()

{

    cin>>n;

    for (int i=0; i<n; i++)

    {

        cin>>h[i];

        s[i]={h[i], i};

    }

    sort(s, s+n);

    for (int i=0; i<n; i++)

    {

        if (s[i].sec==0 || h[s[i].sec-1]<s[i].fir) cur--;

        else if (h[s[i].sec-1]>s[i].fir) cur++;

        if (s[i].sec==n-1 || h[s[i].sec+1]<s[i].fir) cur--;

        else if (h[s[i].sec+1]>s[i].fir) cur++;

        if (i==n-1 || s[i].fir!=s[i+1].fir) ans=max(ans, cur/2);

    }

    cout<<ans<<endl;

    return 0;

}

// Linear Scan

void scan(int i)

{

    sort(st[i&1].begin(), st[i&1].end());

    for (auto b:st[i&1]) dp[i&1][b]=1e18;

    int k, minn;

    // scan left

    k=0, minn=1e18;

    for (auto j:st[i&1])

    {

        while (k<st[1-i&1].size() && st[1-i&1][k]<=j)

        {

            minn=min(minn, dp[1-i&1][st[1-i&1][k]]-(st[1-i&1][k]-\*st[1-i&1].begin())\*s[i]);

            k++;

        }

        if (minn!=1e18) dp[i&1][j]=min(dp[i&1][j], minn+(j-st[1-i&1][0])\*s[i]);

    }

    // scan right

    k=st[1-i&1].size()-1, minn=1e18;

    for (int \_=st[i&1].size()-1; \_>=0; \_--)

    {

        int j=st[i&1][\_];

        while (k>=0 && st[1-i&1][k]>=j)

        {

            minn=min(minn, dp[1-i&1][st[1-i&1][k]]-(\*st[1-i&1].rbegin()-st[1-i&1][k])\*s[i]);

            k--;

        }

        if (minn!=1e18) dp[i&1][j]=min(dp[i&1][j], minn-(j-\*st[1-i&1].rbegin())\*s[i]);

    }

}

// Mo's Algorithm

void add(int pos)

{

    cnt[s[pos]]++;

    if (cnt[s[pos]]==1) cur++;

}

void del(int pos)

{

    cnt[s[pos]]--;

    if (cnt[s[pos]]==0) cur--;

}

    sort(q.begin(), q.end(), [](node x, node y)

    {

        if (x.l/B!=y.l/B) return x.l<y.l;

        else if ((x.l/B)&1) return x.r<y.r;

        else return x.r>y.r;

    });

// Monotonic Queue

signed main()

{

    cin>>n>>m;

    for (int i=0; i<n; i++) cin>>s[i];

    deque<int> q; q.push\_back(0);

    for (int i=1; i<n; i++)

    {

        while (q.size() && q.front()<=i-m) q.pop\_front();

        while (q.size() && s[q.back()]>s[i]) q.pop\_back();

        q.push\_back(i);

        if (i+1>=m) cout<<s[q.front()]<<' ';

    }

    cout<<endl;

    return 0;

}