# 单调队列及其应用

## POJ2823

【题意】

移动区间(长度固定)最值问题。

【分析】 这类思想在单调队列优化思想中尤其重要：区间长度为k，求区间内的最大值，考虑第i个数和第j个数，j-i<k，若a[i]<a[j]，那么a[i]将毫无用处。直觉上理解，因为窗口的移动，a[i]要比a[j]先移出去，无论如何，区间的最大值都不可能是a[i]。 这样，考虑构造一个单调递增的队列，存放相应的序号，当a[队尾]>=要入队数据a[i]，删除队尾元素；当队头<=i-k时，删除队头元素。

/\*\*

单调队列：

加入找最小数，考虑顺序a,b(b在a的后面)，若b<a,当b入队列后，a不可能称为最小值(a比b先出)，删去。

每个元素出队列和入队列一次，时间复杂度为O(n)

\*/

#include <iostream>

#include <cstdio>

#include <cstring>

using namespace std;

const int N = 1100000;

int n,k;

int a[N];

int DanDiao\_Que[N]; //单调递减队列(最大),单调递增队列(最小)

int head,tail;

//递增

void Min()

{

int i;

int head = 1;

int tail = 0;

for(i = 0; i < k-1; i++)

{

while(head<=tail && a[DanDiao\_Que[tail]]>=a[i]) tail--;

tail++;

DanDiao\_Que[tail] = i;

}

for(i = k-1; i < n; i++)

{

while(head<=tail && a[DanDiao\_Que[tail]]>=a[i]) tail--;

tail++;

DanDiao\_Que[tail] = i;

while(DanDiao\_Que[head]< i-k+1) head++;

printf("%d",a[DanDiao\_Que[head]]);

printf("%c",i==n-1?'\n':' ');

}

}

//递减

void Max()

{

int i;

int head = 1;

int tail = 0;

for(i = 0; i < k-1; i++)

{

while(head<=tail && a[DanDiao\_Que[tail]]<=a[i]) tail--;

tail++;

DanDiao\_Que[tail] = i;

}

for(i = k-1; i < n; i++)

{

while(head<=tail && a[DanDiao\_Que[tail]]<=a[i]) tail--;

tail++;

DanDiao\_Que[tail] = i;

while(DanDiao\_Que[head]< i-k+1) head++;

printf("%d",a[DanDiao\_Que[head]]);

printf("%c",i==n-1?'\n':' ');

}

}

int main()

{

scanf("%d %d",&n,&k);

for(int i = 0 ;i < n; i++)

scanf("%d",&a[i]);

Min();

Max();

return 0;

}

## poj-3250-Bad Hair Day

题意：有 n 头牛头朝东站成一列。每头牛有一定的高度，并且只能看到它右边比他矮的牛的牛头，直到被某头高度大于等于它的高度的牛所挡住。计算每头牛能看到的牛头顶的数量之和

Sample Input

6

10

3

7

4

12

2

Sample Output

5

#include <iostream>

#include <cstdio>

#include <cstring>

#define MAXN 80050

#define LL long long

using namespace std;

LL q[MAXN];

int main()

{

int n;

scanf("%d",&n);

LL a;

LL ans=0;

int rear=-1,front=0;

for(int i=0;i<n;i++){

scanf("%I64d",&a);

while(rear>=front&&q[rear]<=a)rear--;

q[++rear]=a;

ans+=rear;

}

cout<<ans<<endl;

return 0;

}

## HDU 3415 Max Sum of Max-K-sub-sequence（单调队列）

题意:给你一个循环序列{An}，让你找出长度不大于K的连续子序列，使这个子序列和最大。

解题思路：由于子序列连续，所以它的和可以通过两个前缀和作差得到。（即sum[i~j]=sum[j]-sum[i-1](j-i+1<=k)）。而对于同一个序列终点来说，起点左边那个元素的前缀和越小，所得到的和越大。所以，题目又变成了求滚动区间的最小值问题。

#include <stdio.h>

#include <limits.h>

const int M = 100001<<1;

int sum[M],q[M];

int main()

{

int z,n,k;

scanf("%d",&z);

while(z--)

{

int start,end,max;

int head,rear;

scanf("%d%d",&n,&k);

for(int i=1;i<=n;i++)

{

scanf("%d",&sum[i]);

sum[i+n] = sum[i];

}

for(int i=2;i< n+k;i++)

sum[i] += sum[i-1];

head = rear = 0;

q[head] = 0;

max = INT\_MIN;

for(int i=1;i< n+k;i++) //枚举每个区间终点

{

while(head<=rear && sum[i-1]<=sum[ q[rear] ])

rear--;

q[++rear] = i-1;

if(q[head]+1 < i-k+1) //起点如果大于区间范围，删除队首

head++;

if(sum[i] - sum[ q[head] ] > max)

{

start = q[head]+1;

end = i;

max = sum[i] - sum[ q[head] ];

}

}

end = end>n ? end%n : end;

printf("%d %d %d\n",max,start,end);

}

return 0;

}

## hdu 4193 单调队列

将一个序列形成一个环，序列中的每个数字轮流作为起始端，问有多少个起点，可以使前1项，前两项……前n项加和全都为非负数。

既然数组是循环的，自然想到先复制一遍，然后求sum[i],枚举每一个长度为n的区间，看最小值是否满足即可。

#include <stdio.h>

#define N 2000002

int a[N],sum[N],que[N];

int main()

{

int i,n;

while(scanf("%d",&n) == 1 && n)

{

for(i = 1; i <= n; ++i)

{

scanf("%d",a + i);

a[i + n] = a[i];

}

int nn = n << 1;

for(i = 1; i <= nn; ++i)

sum[i] = sum[i-1] + a[i];

int head = 0,tail = 0;

int res = 0;

for(i = 1; i < nn; ++i)

{

while(head < tail && sum[i] < sum[que[tail-1]])

--tail;

que[tail++] = i;

if(i >= n && sum[que[head]] >= sum[i-n])

++res;

if(que[tail-1] - que[head] >= n)

++head;

}

printf("%d\n",res);

}

return 0;

}

## HDU3474

/\*\*

题意: 有一环形项链，项链上有两种珍珠分别用Ｃ，Ｊ来表示．现在问是否存在这样的切割点，满足以下条件：

沿着该切割后的项链的一端依次取出珍珠，但手里的C珍珠数必须时刻不少于手里的J珍珠数．问这样的切割点有几个．

由n(10^6)个数据组成的圆环,数据为1和-1,问从一个点开始顺时针或逆时针,能遍历完所有点,并且保证中间过程中sum>=0。

分析:首先暴力O(n^2)是不可行的。

假设从i点开始,这里仅考虑向左,必须保证sum(j,i)>=0, i-n <j <= i.

设sum[i]表示从0到i点的和,即保证sum[i]-sum[j]>=0,即sum[i] - max(sum[j])>=0.

要求区间[i-n,i-1]最大值,维护单调递减队列即可。

\*/

#include<stdio.h>

#include<string.h>

#include<algorithm>

using namespace std;

#define N 1000005

int T,sum[2\*N],lab[2\*N];

char le[2\*N];

bool vis[2][2\*N];

int front,rear;

void solve(int n,int d)

{

int i;

sum[0]=0; front=rear=0;

for(i=1;i<=2\*n;i++) sum[i]=sum[i-1]+(le[i]=='C'?1:-1);

for(i=1;i<n;i++)

{

while(front<rear&&sum[lab[rear-1]]>=sum[i]) rear--;

lab[rear++]=i;

}

for(i=n;i<=2\*n;i++)

{

while(front<rear&&lab[front]<=i-n) front++;

while(front<rear&&sum[lab[rear-1]]>=sum[i]) rear--;

lab[rear++]=i;

vis[d][i-n]=(sum[lab[front]]>=sum[i-n]);

}

}

int main()

{

int i,j,k,len;

scanf("%d",&T);

for(int end=1;end<=T;end++)

{

scanf("%s",le+1);

len=strlen(le+1);

for(i=1;i<=len;i++) le[i+len]=le[i];

solve(len,0);

reverse(le+1,le+1+2\*len);

solve(len,1);

for(k=0,i=1;i<=len;i++)

k+=vis[0][i]|vis[1][len-i];

printf("Case %d: %d\n",end,k);

}

return 0;

}

## 广告印刷

【问题描述】

最近，afy决定给TOJ印刷广告，广告牌是刷在城市的建筑物上的，城市里有紧靠着的N(N<=400000)个建筑。afy决定在上面找一块尽可能大的矩形放置广告牌。我们假设每个建筑物都有一个高度，从左到右给出每个建筑物的高度H1,H2…HN，0<Hi<=1,000,000,000，并且我们假设每个建筑物的宽度均为1。要求输出广告牌的最大面积。

【输入样例】

6

5 8 4 4 8 4

【输出样例】

24

【分析】考虑构造一个单调非递减队列，从左至右，依次加入到队列中，肯定会有元素出队列，设当前要插入的数为a，要出队列的数为b，必有b>=a，则b向右能到达的最远距离就是b-a。注意在求解时，让0先入队列，这样保证每个数据都会出队列。同理，左极限也可求出。

#include <iostream.h>

#define MAXN 1000000

int h[MAXN+5]; //建筑物的高度

int n; //建筑物的数目

int mq[MAXN+5]; //单调队列，对内元素为建筑物高度的下标

int left[MAXN+5]; //left[i]：在第i个建筑物左侧，不比它的高度小的建筑物数量

int right[MAXN+5]; //right[i]：在第i个建筑物右侧，不比它的高度小的建筑物数量

void CalcLeft()

{

mq[0] = 0;

int front = 0, rear = 1;

int i;

for (i = 1; i <= n; i++)

{

while (front < rear && h[i] <= h[mq[rear-1]])

{

rear--;

}

left[i] = i - mq[rear-1] - 1;

mq[rear++] = i;

}

}

void CalcRight()

{

mq[0] = n + 1;

int front = 0, rear = 1;

int i;

for (i = n; i >= 1; i--)

{

while (front < rear && h[i] <= h[mq[rear-1]])

{

rear--;

}

right[i] = mq[rear-1] - i - 1;

mq[rear++] = i;

}

}

int MaxRectArea()

{

int maxArea = -1;

int i;

for (i = 1; i <= n; i++)

{

int area = (left[i] + right[i] + 1) \* h[i];

if (area > maxArea)

{

maxArea = area;

}

}

return maxArea;

}

int main()

{

while (cin >> n)

{

int i;

for (i = 1; i <= n; i++)

{

cin >> h[i];

}

h[0] = h[n+1] = -1;

CalcLeft();

CalcRight();

for (i = 1; i <= n; i++)

{

cout << left[i] << ' ';

}

cout << endl;

for (i = 1; i <= n; i++)

{

cout << right[i] << ' ';

}

cout << endl;

cout << MaxRectArea() << endl;

}

return 0;

}

/\*

6

5 8 4 4 8 4

6

9 5 8 2 8 8

6

9 6 8 2 8 8

\*/

## HDU 3530 Subsequence（单调队列）

题意：给一个长度为 n 的序列，找出最长的子序列满足 子序列中的最大值和最小值 的差在区间 [m,k] 范围内。

解题思路：维护两个单调队列，分别存最小值和最大值。

利用 two pointer 的思想，用 j 指向正在考虑中的区间左端点位置，出队时不断右移左指针 j，对于每个 i 不断更新 ans。

#include <stdio.h>

#include <string.h>

#include <algorithm>

using namespace std;

const int N = 1e5 + 5;

int a[N],qmin[N],qmax[N];

int main()

{

int n,m,k;

while(~scanf("%d%d%d",&n,&m,&k))

{

for(int i=0;i<n;i++)

scanf("%d",&a[i]);

int head1 = 0 , rear1 = -1;

int head2 = 0 , rear2 = -1;

int ans = 0;

for(int i=0,j=0;i<n;i++)

{

while(head1 <= rear1 && a[ qmin[rear1] ] >= a[i])

rear1--;

qmin[++rear1] = i;

while(head2 <= rear2 && a[ qmax[rear2] ] <= a[i])

rear2--;

qmax[++rear2] = i;

while(a[ qmax[head2] ] - a[ qmin[head1] ] > k)

{

if(j == qmin[head1])

++head1;

if(j == qmax[head2])

++head2;

j++;

}

if(j <= i && a[ qmax[head2] ] - a[ qmin[head1] ] >= m)

ans = max(ans,i-j+1);

}

printf("%d\n",ans);

}

return 0;

}

## FZU- Problem 2080 最大差值

题意：给出一个n\*m的矩阵，要求求出它所有r\*c子矩阵里面的元素的最大值减最小值的差的最大值。

思路：二维的单调队列维护，时间复杂度为O（n^2）。

#include<cstdio>

#include<map>

#include<cstring>

#include<vector>

#include<queue>

#include<algorithm>

#include<cmath>

#include<set>

using namespace std;

#define N 1005

int head,tail,mat[N][N],vmax[N][N],vmin[N][N]; //vmax[i][j]表示(i,j)到(i+d-1,j+d-1)中的最大值,vmin[i][j]表示(i,j)到(i+d-1,j+d-1)中的最小值。

struct node

{

int num,id;

};

node que[N];

void add\_max(int num,int id)

{

while(head<tail&&que[tail-1].num<=num)tail--; //单调队列维护

que[tail].id=id;

que[tail++].num=num;

}

void add\_min(int num,int id)

{

while(head<tail&&que[tail-1].num>=num)tail--;

que[tail].id=id;

que[tail++].num=num;

}

int get\_num(int id)

{

while(head<tail&&que[head].id<id)head++;

return que[head].num;

}

int main()

{

//freopen("a.txt","r",stdin);

int n,m,r,c;

while(scanf("%d%d%d%d",&n,&m,&r,&c)!=EOF)

{

for(int i=0;i<n;i++)

{

for(int j=0;j<m;j++)

{

scanf("%d",&mat[i][j]);

}

}

for(int i=0;i<m;i++)

{

head=tail=0;

for(int j=0;j<r-1;j++)add\_max(mat[j][i],j);

for(int j=r-1;j<n;j++)

{

add\_max(mat[j][i],j);

vmax[j-r+1][i]=get\_num(j-r+1); //这时的vmax[j-r+1][i]表示第i列中的第j-r+1行到第j行的最大值

}

}

for(int i=0;i<=n-r;i++)

{

head=tail=0;

for(int j=0;j<c-1;j++)add\_max(vmax[i][j],j);//这里就是二维单调队列维护了

for(int j=c-1;j<m;j++)

{

add\_max(vmax[i][j],j);

vmax[i][j-c+1]=get\_num(j-c+1);

}

}

for(int i=0;i<m;i++)

{

head=tail=0;

for(int j=0;j<r-1;j++)add\_min(mat[j][i],j);

for(int j=r-1;j<n;j++)

{

add\_min(mat[j][i],j);

vmin[j-r+1][i]=get\_num(j-r+1);

}

}

for(int i=0;i<=n-r;i++)

{

head=tail=0;

for(int j=0;j<c-1;j++)add\_min(vmin[i][j],j);

for(int j=c-1;j<m;j++)

{

add\_min(vmin[i][j],j);

vmin[i][j-c+1]=get\_num(j-c+1);

}

}

int ans=0;

for(int i=0;i<=n-r;i++)

{

for(int j=0;j<=m-c;j++)

{

ans=max(ans,vmax[i][j]-vmin[i][j]);

}

}

printf("%d\n",ans);

}

return 0;

}