**斜率优化DP个人总结**

[acm菜鸟生活](http://blog.163.com/dwq_life/blog/#m=0&t=1&c=fks_087074093086080068081083081095085080083068093084080074084) 2010-08-28 17:19:05 阅读6 评论0   字号：大中小 订阅

      学斜率优化也是因为当时做单调队列而开始的，其实两种DP都是基于题目的DP方程在选取最优值时存在单调性。因此这样就可以把本来O(n)的时间压到O(1)了。听起来也觉得有点神奇吧！！！！！

我前面已经发过一篇斜率优化的是基础了，想明白了相信很容易就可以A掉！！

可以先从这论文里学基于单调性的DP：<http://wenku.baidu.com/view/ef259400bed5b9f3f90f1c3a.html>

做斜率优化DP题目最重要的就是先把题目的DP方程写出来，然后就是构造出 X、Y、K 的问题了，但要注意要证明 X、K的单调性，如果不单调的话就不可以用斜率优化了！！！！！！

拿个题目为例：<http://acm.hdu.edu.cn/showproblem.php?pid=2829>

先把题目的方程写出来，一般都比较容易吧.........

根据题意我们就可以例出DP方程：

a[i][j]  =  min{ a[i-1][k] + s[j-2]\*s[j-1] - s[k]\*s[j-1] - sum[j-2] + sum[k] }   i < k < j     表示 j 后断开，且一共断开了 i 次

s[i] = w[1] + w[2] + ... + w[i] ;                 sum[i] = s[1]\*w[1] + s[2]\*w[2] + ... +s[i]\*w[i] ;

好了，方程写出来了，就要开始构造X、Y、K 了

 对于每次我们是要找最优的 k 值！！！

因此我们就可以把  a[i-1][k]+sum[k] 看作是 Y， 把 s[k]看作是 X， s[j-1] 看作是斜率K，很明显X、Y、K都是单调递增的~~~~~

这样基本上就等于套模板一样就能把题目做出来了！！！！！！！！！！

但有一点要注意：如果X不是严格递增（或递减）的话就会出现斜率不存在的情况！！！！！！自己处理一下就好了~~~~~~

斜率优化的代码一般都不长就写一下入队出队（要注意边界问题！！！！！），几乎就没了，呵呵！！！！

代码：

#include<iostream>  
using namespace std;

\_\_int64 s[1005], sum[1005],a[1003][1003],mark[1003];  
double list[1003];

int main()  
{  
    \_\_int64 n,m,w,i,j,l,r, min;  
    while(scanf("%I64d%I64d", &n,&m) && n)  
    {  
        for(i=1; i<=n; i++)  
        {  
            scanf("%I64d", &w);  
            s[i] = s[i-1] + w;  
            sum[i] = sum[i-1] + s[i]\*w;  
        }  
        if(m)  
        {  
            for(i=2; i<=n; i++)  
            {  
                a[1][i] = s[i-1]\*s[i-2] - sum[i-2];  
            }    
            for(i=2; i<=m; i++)  
            {  
                l=r=0;  
                list[0] = 0;  
                mark[0] = i;  
                for(j=i+1; j<=n; j++)  
                {  
                  while(l<r && a[i-1][mark[l+1]]+sum[mark[l+1]-1] - s[j-1]\*s[mark[l+1]-1] <= a[i-1][mark[l]]+sum[mark[l]-1] - s[j-1]\*s[mark[l]-1])  
                    {  
                        l++;  
                    }  
                    a[i][j] = a[i-1][mark[l]] + s[j-2]\*s[j-1] - s[mark[l]-1]\*s[j-1] - sum[j-2] + sum[mark[l]-1];  
                    while( ((a[i-1][j]+sum[j-1])-(a[i-1][mark[r]]+sum[mark[r]-1])) / (s[j-1]-s[mark[r]-1]) <= list[r])  
                    {  
                        r--;  
                    }  
                    list[r+1] = ((a[i-1][j]+sum[j-1])-(a[i-1][mark[r]]+sum[mark[r]-1])) / (s[j-1]-s[mark[r]-1]);  
                    mark[r+1] = j;  
                    r++;  
                }  
            }  
            min = s[n]\*s[n-1] - sum[n-1];  
            for(i=m+1; i<=n; i++)  
            {  
                if(a[m][i]+sum[i-1]-s[n]\*s[i-1] < min)  
                    min = a[m][i]+sum[i-1]-s[n]\*s[i-1];  
            }  
            min += s[n]\*s[n-1] - sum[n-1];  
        }  
        else  
            min = s[n]\*s[n-1] - sum[n-1];  
        printf("%I64d\n", min);  
    }  
    return 0;  
}