斜率优化DP

题意

1. 给定 n，L
2. 给定n个物品长度 编号为1~n
3. 将n个物品分成连续的若干段
   如果 从 I 到 J 组成一段，他的费用是 （J - I + sum[J] - sum[I-1] - L）²
4. 目标：总费用最小

思路

- F[I]： 前I个物品分成连续的若干份最小花费
  sum[I]： 前缀和
- 朴素方程
  F[I] = min（F[J] + （J - I + sum[J] - sum[I-1] - L）² ）
- 变形
  F[I] = min（F[J] + (sum[I] + I - sum[J] - J - L - 1) ²)
  令 S[I] = sum [I] + I , C = L +1
  F[I] = min (F[J] + (S[I] - S[J] - C) ²)
  = min (F[J] + S [I] ² - 2 * S[I] * S[J] - 2 * S[I] * C +2*S [J]* C + S[J] ² + C ²
  = min (F[J] + 2 * (C - S[I]) * S[J] + S[J] ²) + C ² - 2 * S[I] * C
  F [I] = F[J] + 2 * (C - S[I]) * S[J] + S[J] ²
  → F[J] + S [J] ² = F[I] + 2 (S [I] - C) * S [J]
  令 Y = F[J] + S [J] ²
  X = S [J]
  K = 2 (S [I] - C)
  → Y = F [I] + K * X
  接下来应该是套路，具体看代码

```
#include<bits/stdc++.h>
using namespace std;
typedef long long LL;
LL s[50005];
LL que[50005],head,tail;
LL f[50005];
double Y(LL j)
```

```
{
    return f[j]+s[j]*s[j];
}
double X(LL j)
{
    return s[j];
}
double js(LL j,LL k)
{
    return (Y(k)-Y(j))/(X(k)-X(j));
}
int main()
{
    ios::sync_with_stdio(false);
    LL n,L;
    cin>>n>>L;
    LL c;
    for(int i=1;i<=n;i++)
    {
        cin>>c;
        s[i] = s[i-1]+c;
    }
    for(int i=1;i<=n;i++)
    {
        s[i] = s[i] +i;
    }
    L = L+1;
    head = tail = 0;
    que[tail] = 0;
    for(int i=1;i<=n;i++)
    {
        while(head<tail && js(que[head],que[head+1]) <= 2*(s[i]-L))  //队头两个点的斜率 < K
        {
            head++;
        }
        int j=que[head];
        f[i] = f[j] + (s[i] - s[j] - L) * (s[i] - s[j] - L) ; // 由朴素方程转移
        while (head<tail && js(que[tail-1],que[tail])>js(que[tail],i)) //维护下凸包
        {
            tail--;
        }
        que[++tail] = i;
    }
```

```
51      cout<<f[n]<<endl;
52  }
```