You are given a sequence of n integers *a*1, *a*2, ..., *an*.

Determine a real number *x* such that the weakness of the sequence *a*1 - *x*, *a*2 - *x*, ..., *an* - *x* is as small as possible.

The weakness of a sequence is defined as the maximum value of the poorness over all segments (contiguous subsequences) of a sequence.

The poorness of a segment is defined as the absolute value of sum of the elements of segment.

Input

The first line contains one integer *n* (1 ≤ *n* ≤ 200 000), the length of a sequence.

The second line contains *n* integers *a*1, *a*2, ..., *an* (|*ai*| ≤ 10 000).

Output

Output a real number denoting the minimum possible weakness of *a*1 - *x*, *a*2 - *x*, ..., *an* - *x*. Your answer will be considered correct if its relative or absolute error doesn't exceed 10- 6.

Example

Input

3  
1 2 3

Output

1.000000000000000

Input

4  
1 2 3 4

Output

2.000000000000000

Input

10  
1 10 2 9 3 8 4 7 5 6

Output

4.500000000000000

Note

For the first case, the optimal value of *x* is 2 so the sequence becomes  - 1, 0, 1 and the max poorness occurs at the segment "-1" or segment "1". The poorness value (answer) equals to 1 in this case.

For the second sample the optimal value of *x* is 2.5 so the sequence becomes  - 1.5,  - 0.5, 0.5, 1.5 and the max poorness occurs on segment "-1.5 -0.5" or "0.5 1.5". The poorness value (answer) equals to 2 in this case.

给定一个序列A

• 一个区间的poorness定义为这个区间内和的绝对值

• weakness等于所有区间最大的poorness

• 求一个x使得，序列A全部减x后weakness最小

• 1 ≤ n ≤ 2 \* 1e5

对这个数列中的每一个数减去一个相同的数字， 其最大连续和会呈现出单峰函数的现象， x过大或者过小都不行， 那么处理方法显然是三分。

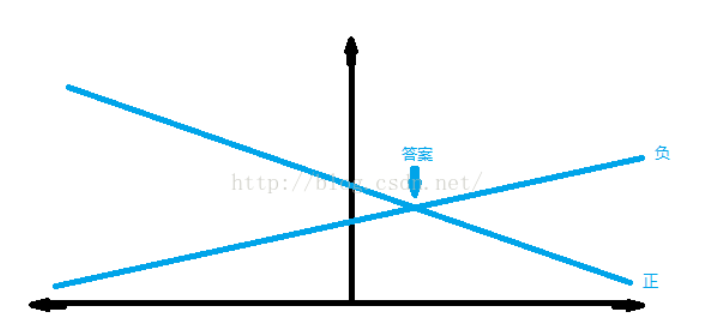
由于该题不是直接三分的答案， 因此三分出的x虽然精度在答案范围内， 但是求出的最大连续和却不一定满足精度。

二分或三分浮点数时， 最稳妥的方法是根据数据范围自己设置二分或三分的次数， 这样使得精度可以最大化的精确。

思路：

x越大，子序列连续和为正时，其绝对值的最大值越小；

x越小，子序列连续和为负时，其绝对值的最大值越小。



所以，随着x由小变大，求得的答案由大变小再变小，使用三分。

这题可以用三分，但是在三分判断条件的时候要注意，用循环限定次数比较好，因为double精度不够高。

我们都知道 [二分查找](http://blog.csdn.net/dgq8211/article/details/7770426) 适用于单调函数中逼近求解某点的值。

如果遇到凸性或凹形函数时，可以用三分查找求那个凸点或凹点。

下面的方法应该是三分查找的一个变形。

如图所示，**已知左右端点L、R，要求找到白点的位置。**

思路：通过不断缩小 [L,R] 的范围，无限逼近白点。

做法：先取 [L,R] 的中点 mid，再取 [mid,R] 的中点 mmid，通过比较 f(mid) 与 f(mmid) 的大小来缩小范围。

           当最后 L=R-1 时，再比较下这两个点的值，我们就找到了答案。

1、当 f(mid) > f(mmid) 的时候，我们可以断定 mmid 一定在白点的右边。

反证法：假设 mmid 在白点的左边，则 mid 也一定在白点的左边，又由 f(mid) > f(mmid) 可推出 mmid < mid，与已知矛盾，故假设不成立。

所以，此时可以将 R = mmid 来缩小范围。

2、当 f(mid) < f(mmid) 的时候，我们可以断定 mid 一定在白点的左边。

反证法：假设 mid 在白点的右边，则 mmid 也一定在白点的右边，又由 f(mid) < f(mmid) 可推出 mid > mmid，与已知矛盾，故假设不成立。

同理，此时可以将 L = mid 来缩小范围。

#include<iostream>

#include<stdio.h>

#include<stdlib.h>

#include<algorithm>

using namespace std;

#define maxn 200060

int n;

double a[maxn],sum1[maxn],sum2[maxn];

double cal(double x)

{

int i,j,f;

double maxnum=0,b;

sum1[0]=sum2[0]=0;

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

{

b=a[i]-x;

sum1[i]=max(b,sum1[i-1]+b);//sum[i]是从1到i的最大区间和

maxnum=max(maxnum,sum1[i]);

b=x-a[i];

sum2[i]=max(b,sum2[i-1]+b);

maxnum=max(maxnum,sum2[i]);

}

return maxnum;

}

int main()

{

int m,i,j;

double l,r,mid,mmid,minx,maxx;

while(cin>>n)

{

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

cin>>a[i];

l=-10005;r=10005;//因为|ai| ≤10000

for(i=0;i<100;i++)//这是在枚举x

{

double d=(l+r)/2.0;

mid=d;

mmid=(d+r)/2.0;

if(cal(mmid)-cal(mid)>=0)

r=mmid;

else l=mid;

}

printf("%.9f\n",cal(l));

}

return 0;

}