登顶计划

湖南师范大学附属中学 彭天翼

1 题目背景

爬山是一项非常有益于身心健康的运动。可是在爬山时，我们应该采取怎样的策略，才能尽快登上山的顶端呢？基于这个有趣的问题，作者思考出了下面这样一道题目。

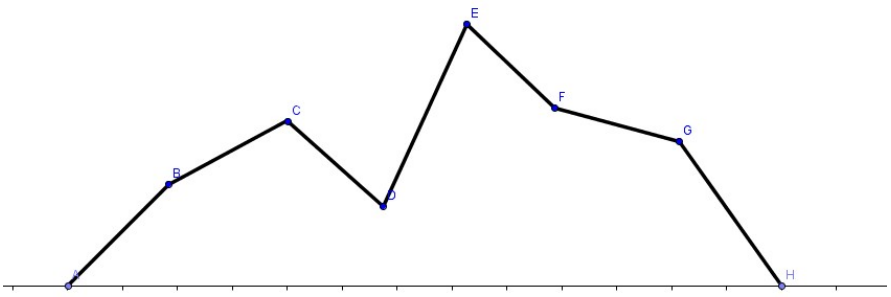
2 题目描述

【资源限制】

时间限制：2s 内存限制：512MB

【问题描述】

二维平面上的山脉由一系列顶点确定：$(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$。相邻的两个顶点之间由线段相连（保证$x_i$严格递增），这样就构成了一座连绵的山脉，每个点的y值代表了该点的高度。如下图所示：



我们定义山脉的内部为顶点之间的折线与x 轴的所夹部分（不包括顶点之间的折线）。如果顶点A与顶点B之间的连线段没有穿过山脉的内部，则我们称顶点A能看见顶点B（或B能看见A）。

现在pty从某个顶点出发，想要登到山脉的顶峰（最高点），他只能在顶点之间的折线上行走。经过思考，他将采取如下的一种登山方式：

1. 站在出发点，向左右看去，如果此时能够看到的最高山峰在左侧，则向左侧走去，否则往右侧走去。

2. 在行走的同时，pty仍然观察着此时左右的最高山峰，一旦发现一座比之前看到的都要高的山峰，他将向此时的最高峰走去。

3. 如果存在某个时刻，pty所站立的位置比左右能看到的最高峰都要高，则他到达了山脉的顶峰，此时他的爬山过程结束。

pty想知道，采取如上的策略，从每个顶点出发，到达最高点的路程分别是多少？（平面中两点的距离等于它们之间连线段的长度）

【输入格式】

第一行一个整数n，表示山脉顶点个数。

接下来n行，第i行两个整数x_i,y_i，表示第i个顶点的坐标。

保证x_i严格递增，y_i互不相同(y_1, y_n除外)，x_i,y_i都为非负整数。保证y_1,y_n的值为0。

【输出格式】

输出共n行：每行一个实数。

第i行的实数表示从第i个顶点出发，到达最高点的路程。

如果输出与标准输出的误差不超过$1e-2$，则该测试点得满分，否则得0分。

【样例输入】

6

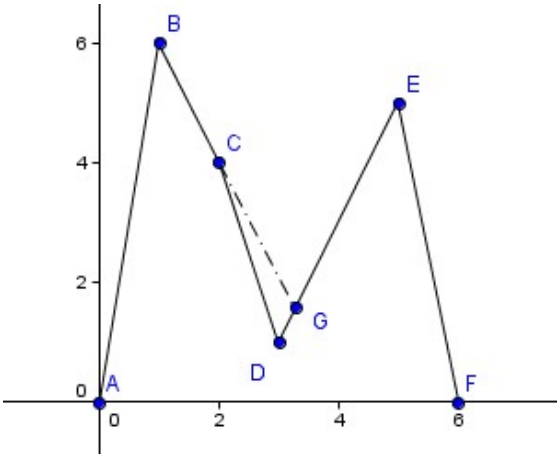
0 0

1 6
2 4
3 1
5 5
6 0

【样例输出】

6.08
0.00
2.24
6.52
9.87
14.97

【样例解释】



路线说明：
A点出发：$A->B$
B点出发：B
C点出发：$C->B$
D点出发：$D->G->D->C->B$
E点出发：$E->D->C->B$

F点出发：$F->E->D->C->B$

从D点出发时，看到的最高点是E，当步行至G点时，发现更高点B，转向后一直步行向B点。从其它点出发后都不需要转向。

【数据规模与约定】

所有的数据满足$xi, yi \leq 100000$。

测试点	限制
1-4	$n \leq 20$
5-8	$n \leq 70$
9-10	$n \leq 100000$且每个顶点都能直接看到最高点
11-14	$n \leq 30000$
15-20	$n \leq 100000$

3 题目分析

3.1 简化问题

题目中要求我们在爬山的任意时刻都要观察两侧当前的最高峰，让我们暂时将问题简单化：变为只有到达顶点时，才观察此时的最高峰。也就是说，现在的观测点，从无限个暂时变为有限的$O(n)$个顶点。

3.2 求一个顶点能看到的最高点

设i号点往左能看到的最高点为L[i]，往右能看到的最高点为R[i]。想象我们站在某个点上，初始时视线水平向左，接下来我们慢慢抬起头，直到我们的视线不再被某座山阻挡，此时我们便找到了L[i]。设当前的顶点为A，L[i]为B。则接下来的两个性质不言而喻：

- 前i个点都处于AB的下方（非严格）
- A，B为前i个点的上凸壳上两个相邻的点

由于A点一定是前i个点的上凸壳的最后一个点，所以B点就是这个上凸壳

的倒数第二个点[1]。于是我们只需要用一个栈维护好前i个点的上凸壳即可，每次加入第i+1个点，同时维护好凸壳的性质。由于总共只有O(n)个顶点，所以我们能在O(n)的时间内求出L[i]和R[i]。

3.3 构建图论模型

3.3.1 暴力做法

得出L[i]与R[i]后，我们有了一个颇为暴力的算法：枚举从每个顶点出发，模拟路线的变化，复杂度为$O(n^2)$或更高。这意味着我们需要继续挖掘其中被重复计算的冗余信息。

3.3.2 建树

令H[i]代表第i个顶点的高度，令T[i] = max(H[L[i]]，H[R[i]])，则T[i]意味着i点能够看到的最高峰的高度。

当我们从A点出发，到达第一个满足$T[B] >= T[A]$的B点时，接下来我们要走的路程和从B点出发走的路程将完全一样。所以A到最高峰的距离就等于B到最高峰的距离再加上A到B的距离。此时让B向A连一条有向边，边权为AB之间的距离。对所有的顶点做完这样的操作以后，我们将得到一棵树（因为最高峰没有入度，其余每个点的入度为1）。从树的根节点走向任意一个点的路径的边权和就等于该点到达最高峰的距离。dfs一遍即可得到每个点的答案。

3.3.3 需要解决的问题

对于一个A点，如何快速求出它往左（或者往右）第一个$T[B] >= T[A]$的点，这是一个十分经典的数据结构问题。我们可以用二分答案加区间RMQ算法解决。接下来介绍一种常数较小的做法。

对所有的顶点按从左往右的顺序建立双向链表。接下来按T值从小到大访问所有的点。每访问到一个点A，则往左第一个满足$T[B] >= T[A]$的B点就在此时双向链表中A的左侧。将A点和B点建边，然后将A点在双向链表中删除。

该算法的复杂度为排序复杂度。

[1]这样一个优美的结论着实令我惊讶。

3.4 回归原问题

至此，我们已经在O(nlogn)的排序时间内解决了一个被简化的问题，接下来让我们回归到原问题。

我们具备了在任意时刻观测最高峰的能力。这会给我们带来什么变化呢？

假设我们现在从A点出发，往右行走，走到某个点B上，发现了左侧的更高峰，转而向左行走。转折点B应该满足怎样的性质呢？设B所在的线段为P_1-P_2。B将满足：

- B点为A点左侧某两个点的连线（设为T_1，T_2）与P_1-P_2的交点
- 前A个点将位于T_1T_2连线的下方（非严格）
- T_1，T_2为前A个点的上凸壳上两个相邻的点

这意味着，有意义的观测点B将是上凸壳上某条边延伸以后与山的折线段的第一个交点。由于前i个点的上凸壳的总边数是O(n)的，所以有意义的观测点也是O(n)的[2]!

那么，如何快速求出这些观测点呢？

T_1与T_2的连线和P_1-P_2有交点，这意味着当P_2加入凸壳后，T_2将要被弹出，而P_1加入凸壳时，T_2没有被弹出。于是我们在维护凸壳的同时，通过求被弹出点和插入点的相关直线的交，就能得到所有观测点。复杂度仍然是O(n)的。

而有限的观测点的问题我们刚刚已经解决了。于是整个问题在O(nlogn)的时间内圆满解决。

4 总结

4.1 灵感来源

本题的灵感来源于对现实生活中爬山问题的思考。用算法知识对现实问题进行分析，不仅能提高我们的算法分析能力，还能反映学习就是为了“学以致用”的本质目的，并且能激发我们对信息学竞赛的兴趣，让我们乐在其中。希望这种思考方式能对大家有所启发。

[2]这又是一个令人兴奋的性质

4.2 解题思路

回顾我们的解题思路：我们先是将问题简单化，变无限为有限，接下来将问题化整为零，一步步地解决。这些都是在解题过程中的一般方法。

下面是作者对解题方法的一些体会，希望与大家分享：

在通常的解题过程中，我们需要在已知条件和所求问题中搭建起一座桥梁。可以利用已学的工具从已知条件顺推，看看我们还能得到什么。也可以从所求问题倒推，看看我们还需要知道什么。当顺推和倒推的大桥有了某个重合点，一条由条件通往问题的桥梁就建好了，问题也就解决了。

可解题的方法是千变万化的，不存在万能的解题秘籍。但值得庆幸是：“当你解决的问题越多，你解决下一个问题的可能性就越大。”所以只要我们勤于练习，乐于思考，解题能力就一定能不断迈上新的台阶。