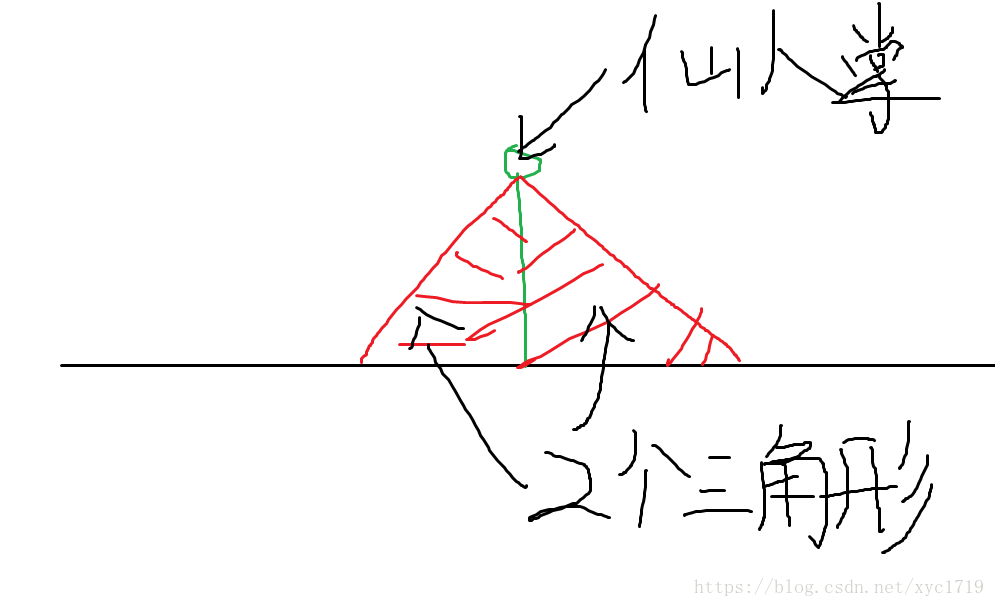


**【分析】**

这道题在考场看到我是完全的蒙蔽的，惊人的妄想着能否用数据离散化后的dp来骗分。。。  
我果然很菜。。。

好吧，对于我这种菜鸡来说，正解似乎有些难想，让我们考虑从最基本的情况开始考虑。先将所有仙人掌按p为第一关键字排序，从左向右扫。这里我们考虑一个贪心策略，**如果仙人掌两两之间距离足够远，则仙人掌之间跳跃的最小高度一定为最高仙人掌的高度**。 关于两个相邻的仙人掌如果可以下落再上跳，则到下一个仙人掌的高度就为它自身的高度。否则就不下落一路上升。

不难发现上述贪心策略如果碰到的下下个仙人掌的高度的高度差，大于它与下个仙人掌的距离差，就很容易出现一种无解情况。难道，这道题目贪心就无法完成  AC的使命吗？

再次观察题目(这就是不能好好利用题目信息的惨痛教训)，稍加思考，我们发现关于每棵仙人掌的那些不可能越过的点(不考虑上下的方向)，恰好关于仙人掌形成两个tan为1的等腰直角三角形，如图

这里我们可以以每个三角形的左端点为关键字排序。如果到第i棵仙人掌的左端点在原有区间的右端点的右边，区间最左边就为p[i]-h[i]，最右边就为  
p[i]+h[i]，同时用仙人掌的高度更新最大高度。反之，就不能下落，且需要构造一个更大的三角形，以跳过这段区间中的所有仙人掌。

1. #include<cstdio>
2. #include<cstring>
3. #include<iostream>
4. #include<algorithm>
5. **using** **namespace** std;
6. **int** n;
7. **struct** node{
8. **double** p,h;
9. }a[301000];
10. **bool** cmp(node x,node y){
11. **return** x.p-x.h<y.p-y.h;
12. }
13. **double** ans=0;
14. **int** main(){
15. cin>>n;
16. **for**(**int** i=1;i<=n;i++)
17. {
18. scanf("%lf%lf",&a[i].p,&a[i].h);
19. **if**(a[i].h>a[i].p){
20. cout<<-1<<endl;
21. **return** 0;
22. }
23. }
24. sort(a+1,a+n+1,cmp);
25. **double** minn=a[1].p-a[1].h;
26. **double** maxx=a[1].p+a[1].h;
27. **for**(**int** i=2;i<=n;i++){
28. **if**(a[i].p-a[i].h>=maxx){
29. minn=a[i].p-a[i].h;
30. maxx=a[i].p+a[i].h;
31. ans=max(ans,a[i].h);
32. }
33. **else**{
34. //          minn=min(minn,(a[i].p-a[i].h));
35. maxx=max(maxx,(a[i].p+a[i].h));
36. ans=max(ans,(maxx-minn)/2.0);
37. }
38. }
39. printf("%.1lf\n",ans);
40. **return** 0;
41. }