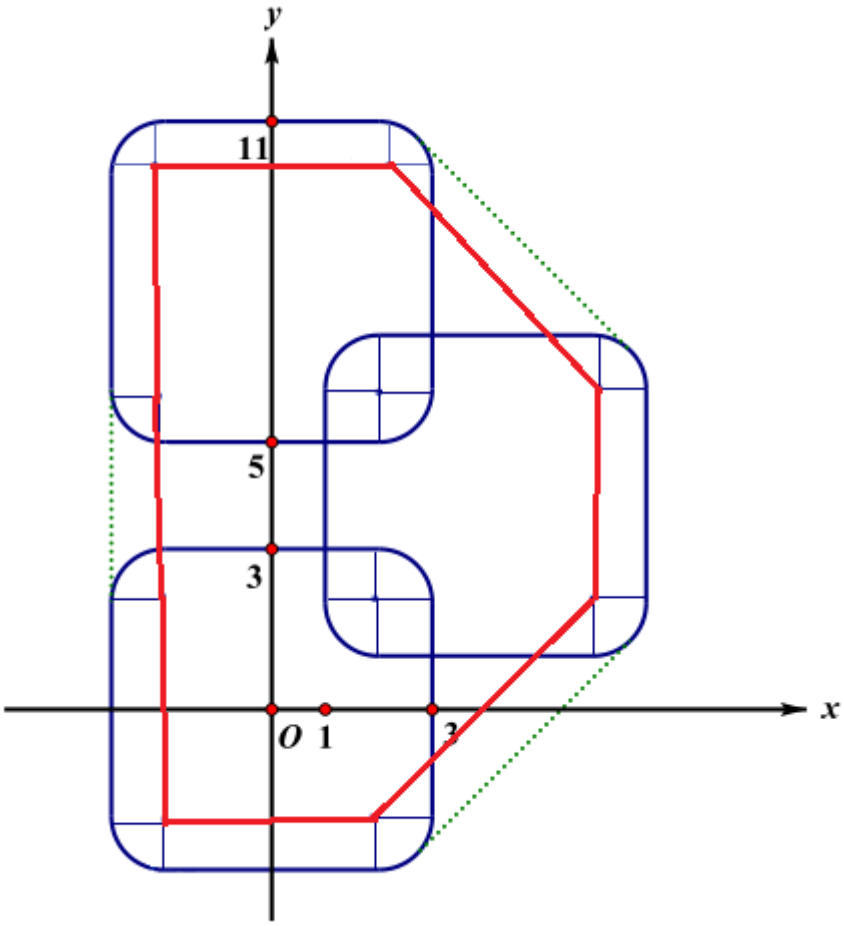
信用卡凸包-凸包精度问题解析

题目传送门

关于这道题，其实你画了图就会发现它所构成的凸包周长其实就是所有圆心构成的凸包周长$+2\pi r$



所以这道题其实就是求一个凸包就好了，$nlog_2n$ $Graham$扫描法。

但是关于这道题我还要引出一个重要的问题，就是关于凸包算法的精度控制。~~虽然我精度没错~~

我们设eps表示我们想将精度控制在什么范围内（一般在$1e$-7~$1e$-9之间）。

- $a == b$ `fabs(a-b)<eps`
- $a < b$ `a<b-eps` （$a \leq b$ `a≤b-eps`）
- $a > b$ `a>b+eps` （$a \geq b$ `a≥b+eps`）

同时，由于这道题需要旋转矩形，sin函数、cos函数都是你容易损失精度的地方，尽量少用。

欧几里得距离也是一样的，用欧几里得距离比大小时，可以不用开根的尽量不要开根。

根号会使你的精度大大损失，对于你读进来的实数，你可以$+eps$，这样可以避免一些关于"0"的精度问题。

附：$\pi = acos(-1)$

Code：

```
```

```cpp
#include <cmath>
#include <cstdio>
#include <algorithm>
using namespace std;
const int Maxn=10005;
const double eps=1e-9,pi=acos(-1.0);
struct Node{double x,y;}A[Maxn<<2],St[Maxn<<2];
int N,Cnt,Tot;
double L,H,R,x,y,theta,alpha,D;
double Multi(Node x,Node y,Node z){
    double x1,x2,y1,y2;
    x1=x.x-z.x,y1=x.y-z.y;
    x2=y.x-z.x,y2=y.y-z.y;
    return x1*y2-x2*y1;
}
double Euild(Node x,Node y){
    return sqrt((x.x-y.x)*(x.x-y.x)+(x.y-y.y)*(x.y-y.y));
}
double _Euild(Node x,Node y){
    return (x.x-y.x)*(x.x-y.x)+(x.y-y.y)*(x.y-y.y);
}
int Cmp(Node x,Node y){
    double vl=Multi(x,y,A[1]);
    if(vl<-eps)return 0;
    if(fabs(vl)<eps&&_Euild(x,A[1])>_Euild(y,A[1]))return 0;
    return 1;
}
int main()
{
    scanf("%d",&N);
    scanf("%lf%lf%lf",&H,&L,&R);
    L-=2.0*R,H-=2.0*R,D=sqrt(L*L+H*H)/2.0;
    alpha=atan(H/L),L/=2.0,H/=2.0;
    for(int i=1;i<=N;i++){
        scanf("%lf%lf%lf",&x,&y,&theta);
//      x+=eps,y+=eps;theta+=eps;//可以调节精度
        A[++Tot]=(Node){D*cos(alpha+theta),D*sin(alpha+theta)};
        A[++Tot]=(Node){-A[Tot-1].x+x,-A[Tot-1].y+y};
        A[Tot-1].x+=x,A[Tot-1].y+=y;
        A[++Tot]=(Node){D*cos(-alpha+theta),D*sin(-alpha+theta)};
        A[++Tot]=(Node){-A[Tot-1].x+x,-A[Tot-1].y+y};
        A[Tot-1].x+=x,A[Tot-1].y+=y;
    }
    for(int i=1;i<=Tot;i++){
        if(A[i].y-A[1].y<-eps || (fabs(A[i].y-A[1].y)<eps&&A[i].x-A[1].x<-eps)){
            swap(A[i],A[1]);
        }
    }
    sort(A+2,A+Tot+1,Cmp);
    St[++Cnt]=A[Cnt],St[++Cnt]=A[Cnt];
    for(int i=3;i<=Tot;i++){
        while(Cnt>1&&Multi(St[Cnt],A[i],St[Cnt-1])<eps)Cnt--;
        St[++Cnt]=A[i];
```

```
    }
    double Ans=Euild(St[1],St[Cnt]);
    for(int i=1;i<Cnt;i++)
        Ans+=Euild(St[i],St[i+1]);
    Ans+=2.0*R*pi;
    printf("%.2f\n",Ans);
    return 0;
}
```