题意

在一个2维平面上有两条传送带，每一条传送带可以看成是一条线段。两条传送带分别为线段AB和线段CD。lxhgww在AB上的移动速度为P，在CD上的移动速度为Q，在平面上的移动速度R。现在lxhgww想从A点走到D点，他想知道最少需要走多长时间

分析

代码

```cpp
#include<algorithm>
#include<bitset>
#include<cstdio>
#include<cstring>
#include<cstdlib>
#include<cmath>
#include<deque>
#include<iostream>
#include<map>
#include<queue>
#include<set>
#include<stack>
#include<string>
#include<vector>
#include<list>
#define For(i,a,b) for(int i=(a); i<=(b) ; i++)
#define _For(i,a,b) for(int i=(a); i>=(b) ; i--)
#define Memset(a,b); memset((a),(b),sizeof((a)));
#define Cout(a,b);  printf("%d",(a));printf(b);
#define Coutc(a,b);  printf("%c",(a));printf(b);
#define Couts(a,b);  printf("%s",(a));printf(b);
using namespace std;
const int INF = 0x3f3f3f3f;
typedef  long long LL;typedef  unsigned long long ULL;typedef  long double LDB;
inline LL CinLL(){LL x=0,f=1;char ch=getchar();while(ch<'0'||ch>'9'){if(ch=='-')f=-1;ch=getchar();}while(ch>='0'&&ch<='9'){x=x*10+ch-'0';ch=getchar();}return x*f;}
inline int Cin(){int x=0,f=1;char ch=getchar();while(!isdigit(ch)){if(ch=='-')f=-1;ch=getchar();}while(isdigit(ch))x=x*10+ch-'0',ch=getchar();return f*x;}
```

```cpp
const double eps = 1e-3;
struct Point{
    double x,y;
};
struct Line{
    Point a,b;
    double v;
}l[3];
inline double dis(Point a,Point b){
    return sqrt((a.x - b.x)*(a.x - b.x) + (a.y - b.y) *(a.y - b.y));
}
double js(Point p1,Point p2)
{
    double tot = 0.0;
    tot+=dis(l[0].a,p1)/l[0].v;
    tot+=dis(p1,p2)/l[2].v;
    tot+=dis(p2,l[1].b)/l[1].v;
    return tot;
}
Point tf1(Point p1)
{
    Point m1,m2;
    Point low,up;
    low = l[1].a;
    up = l[1].b;
    while(dis(low,up) >eps)
    {
        m1.x =low.x+(up.x  - low.x) /3.0;m1.y = low.y+(up.y - low.y)/3.0;
        m2.x =up.x -(up.x  - low.x) /3.0;m2.y = up.y -(up.y - low.y)/3.0;
        double kk = dis(low,up);
        if(js(p1,m1)<=js(p1,m2))
            up = m2;
        else
            low = m1;
    }
    Point res;
    res.x = (low.x+up.x) * 0.5;
    res.y = (low.y+up.y) * 0.5;
    return res;
}
double choose(Point p1)
{
    Point p2 = tf1(p1);
    return js(p1,p2);
}
Point tf0()
```

```cpp
{
    Point m1,m2;
    Point low,up;
    low = l[0].a;
    up = l[0].b;
    while(dis(low,up) >eps)
    {
        m1.x =low.x+(up.x  - low.x) /3.0;m1.y = low.y+(up.y - low.y)/3.0;
        m2.x =up.x -(up.x  - low.x) /3.0;m2.y = up.y -(up.y - low.y)/3.0;
        if(choose(m1)<=choose(m2))
            up = m2;
        else
            low = m1;
    }
    Point res;
    res.x = (low.x+up.x) * 0.5;
    res.y = (low.y+up.y) * 0.5;
    return low;
}
double solve()
{
    Point p1 = tf0();
    return choose(p1);
}
int main()
{
    For(i,0,1)
        cin>>l[i].a.x>>l[i].a.y>>l[i].b.x>>l[i].b.y;
    cin>>l[0].v>>l[1].v>>l[2].v;
    double ans = solve();
    printf("%.2lf\n",ans);
}
```