Geometría

Caso de Estudio

(Programming Challenges - Cap. 13)

Ejemplo:

Plan de vuelo de Superman

Superman tiene al menos dos poderes que los mortales normales no poseen: visión rayos X y la capacidad de volar más rápido que una bala de extrema velocidad. Algunas de sus otras habilidades no son tan impresionantes: tú o yo probablemente podríamos cambiarnos de ropa en una cabina telefónica si nos proponemos hacerlo.

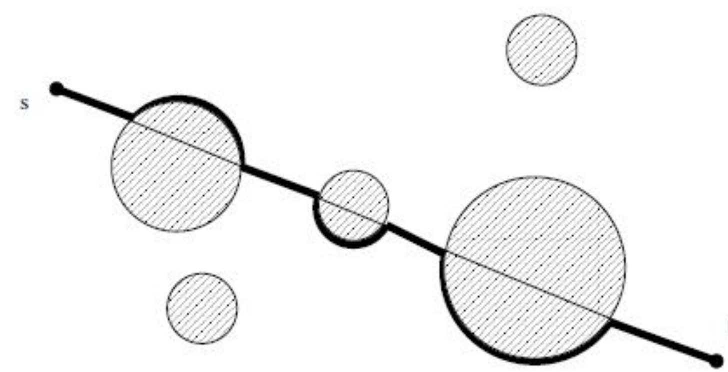
Superman busca demostrar sus poderes entre su posición actual $s = (x_s, y_s)$ y una posición objetivo $t = (x_t, y_t)$. El medio ambiente está lleno de obstáculos de forma circular (o cilíndrica). La visión de rayos X de Superman no tiene alcance ilimitado, estando limitada por la cantidad de material que tiene que ver a través de ellos. Está ansioso por calcular la longitud total de las intersecciones de los obstáculos entre los dos puntos para saber si intenta este truco.

Ejemplo:

Plan de vuelo de Superman

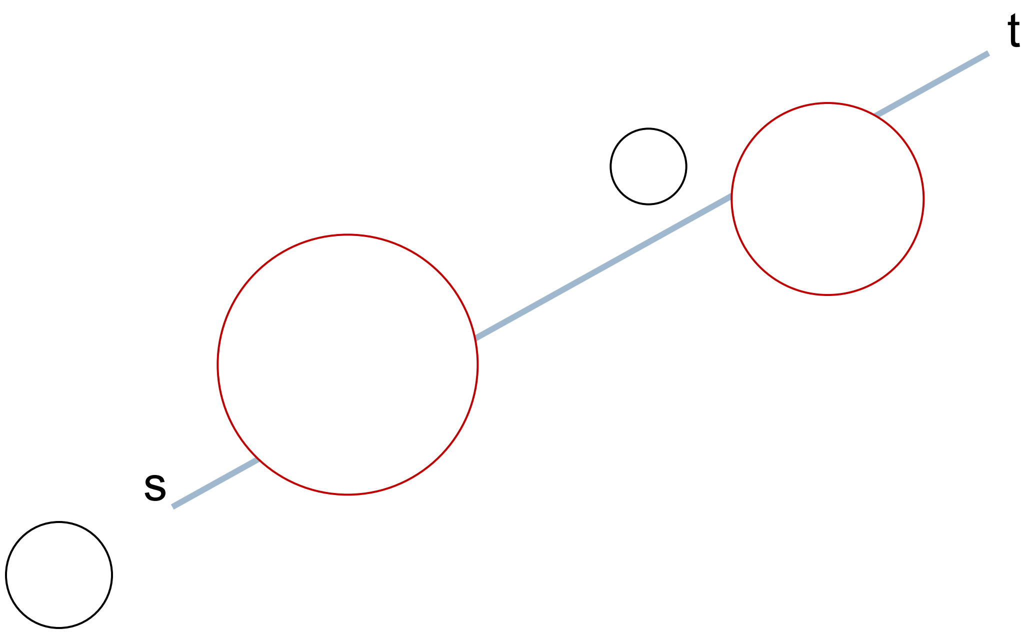
Teniendo en cuenta esto, el Hombre de Acero quisiera volar entre su posición actual y el objetivo. Él puede ver a través de objetos, pero no volar a través de ellos. Su camino deseado vuela directamente a la meta, hasta que choca con un objeto. En este punto, vuela a lo largo del límite del círculo hasta que retorna a la línea recta que une la posición inicial y final. Este no es el camino más corto sin obstáculos, pero Superman siempre toma el más corto de los dos arcos alrededor del círculo.

Usted puede asumir que ninguno de los obstáculos circulares se cruzan entre ellos, y que tanto las posiciones de inicio como las de destino quedan fuera de los obstáculos. Los círculos se especifican dando las coordenadas centrales y el radio.



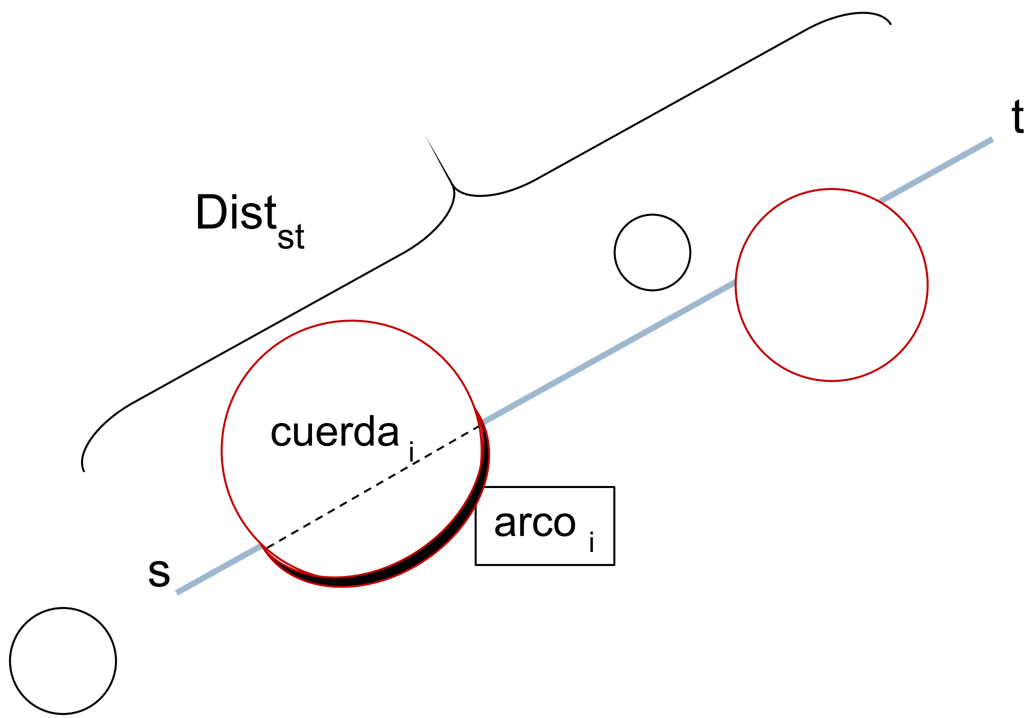
Plan de vuelo de Superman

Objetivo: Calcular la distancia mínima entre ***s*** y ***t*** (Dm)



Plan de vuelo de Superman

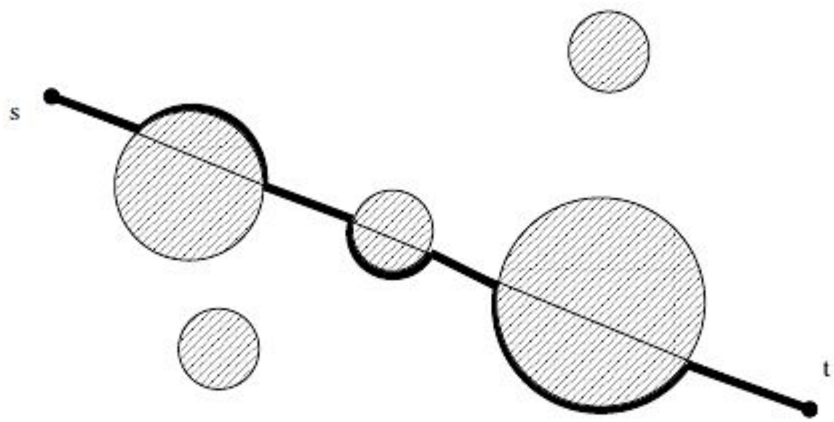
Objetivo: Calcular la distancia mínima entre ***s*** y ***t*** (Dm)



Dm = Dist$_{st}$ - ∑ { – cuerda $_i$ + arco $_i$ } ∀ círculo que interseca la recta ***st***

Plan de vuelo de Superman

Operaciones geométricas básicas requeridas para resolver el problema



Para resolver el problema se requieren tres operaciones básicas:

(1) Intersección entre un círculo y la línea ***l*** de la trayectoria entre ***s*** y ***t***

(2) Cálculo de la longitud de la cuerda

(3) Cálculo de la longitud del arco más pequeño del círculo que intersecta ***l***

Plan de vuelo de Superman

Objetivo: Calcular la distancia mínima entre ***s*** y ***t*** (Dm)

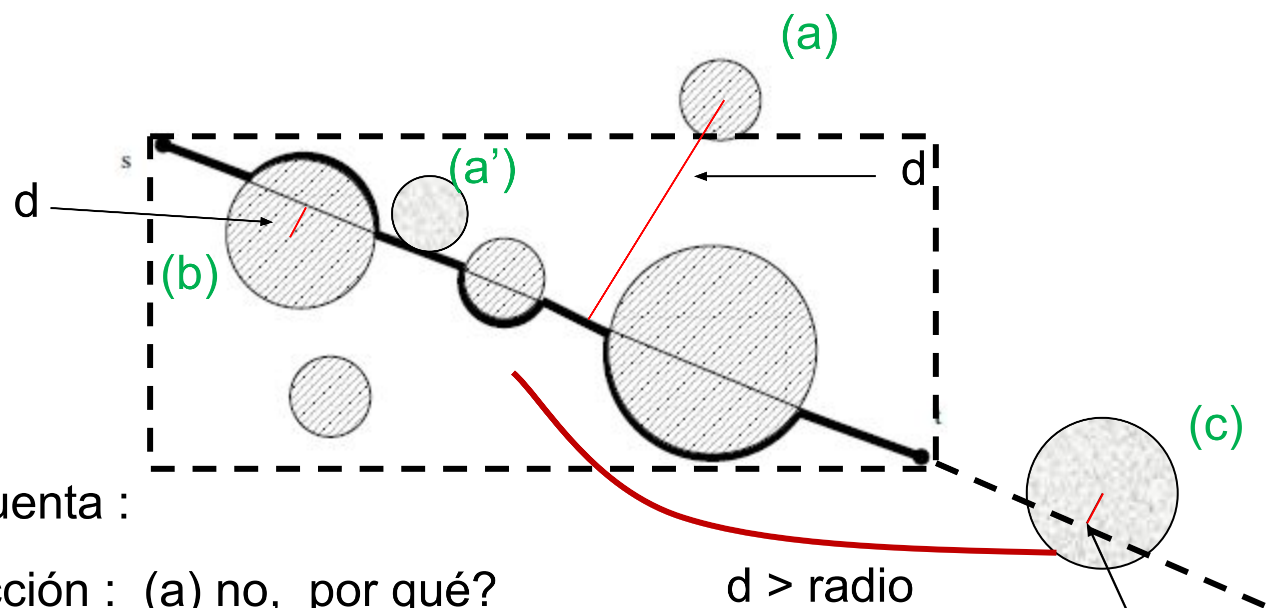
$Dm = Dist_{st} - \sum \{ -cuerda_i + arco_i \}\ \forall$ círculo que interseca la recta ***st***

Pasos a seguir:

```
Definir recta l que pasa por los puntos s y t ;
cuerdas= 0.0 ; arcos = 0.0;
Para cada círculo ci {
        si ci  se interseca con la recta l  entonces
            { calcular  arcoi  y cuerdai ;
              arcos  += arcoi ;
              cuerdas += cuerdai ;
             }
}
recorrido = distancia(s,t) - cuerdas + arcos;
printf(recorrido);
```

Plan de vuelo de Superman

(1) Intersección entre un círculo y la línea ***l*** de la trayectoria entre ***s*** y ***t***



Tener en cuenta :

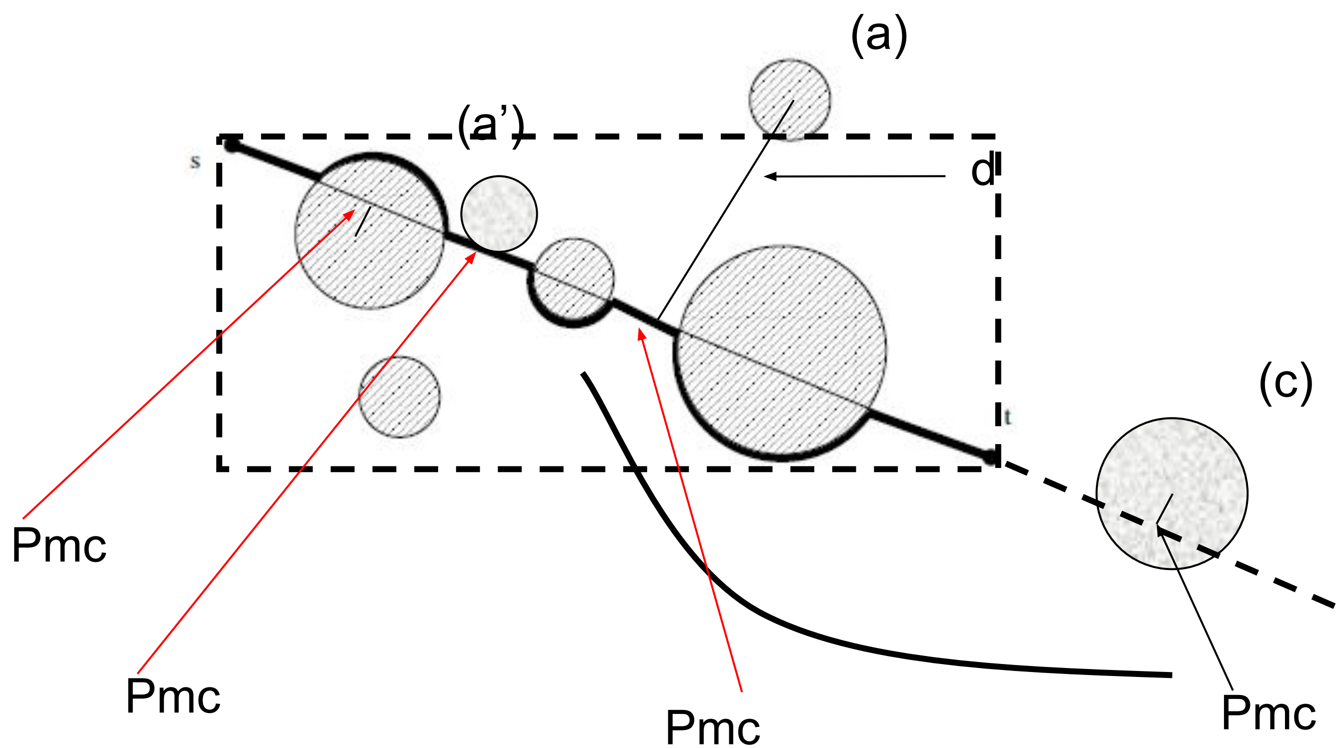
(1) Intersección : (a) no, por qué? d > radio

y (a’) ? sí, d = radio (pero, **no** lo tomamos)

(b) sí, d < radio

(c) sí, d < radio, pero no está en la trayectoria.
Cómo se chequea si está o no en la trayectoria?

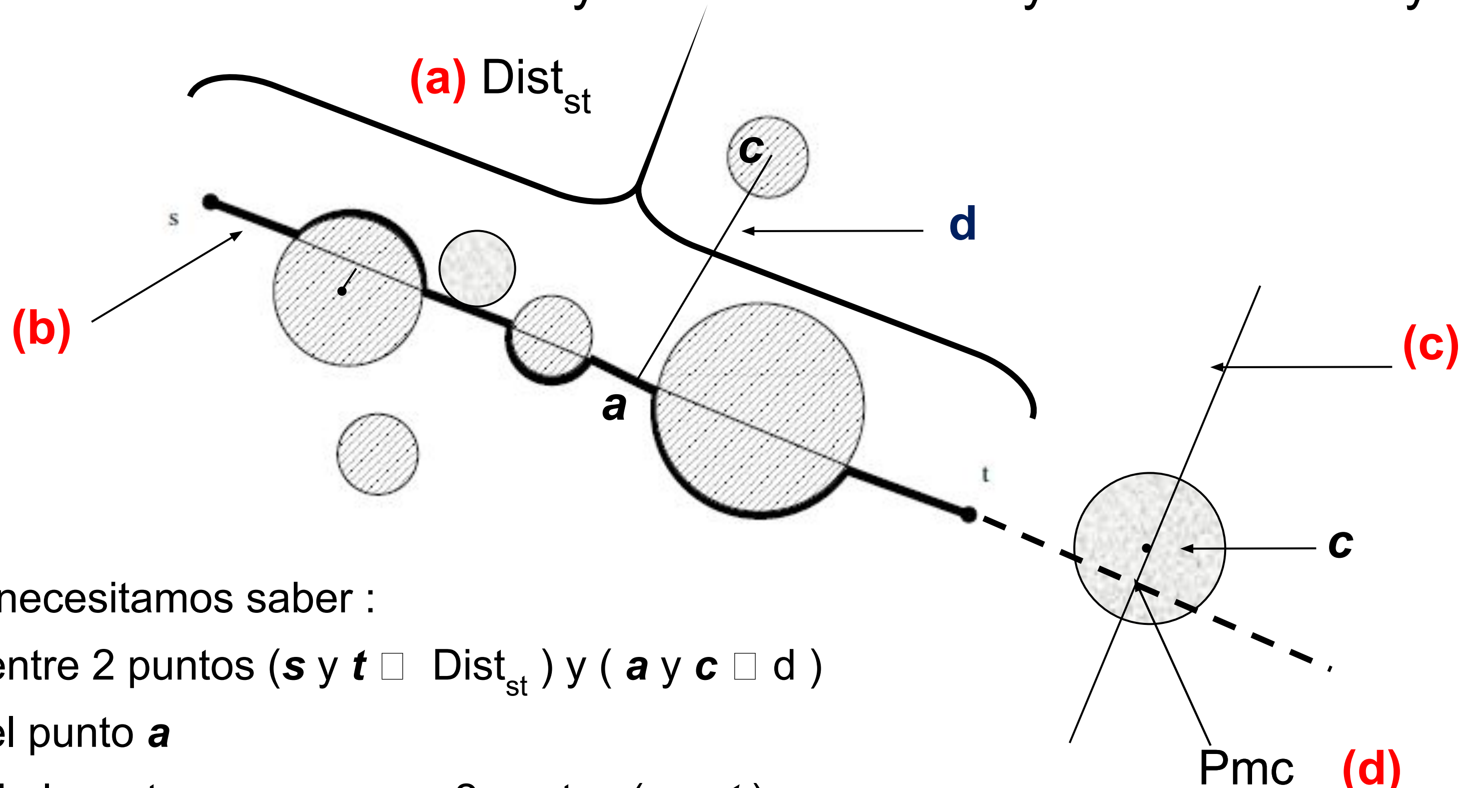
Plan de vuelo de Superman



Chequearemos que los puntos más cercanos al segmento de la trayectoria caigan en el Box formado por las coordenadas de ***s*** y ***t***

Plan de vuelo de Superman

(1) Intersección entre un círculo y la línea ***l*** de la trayectoria entre ***s*** y ***t***



Resumiendo, necesitamos saber :

(a) Distancia entre 2 puntos (***s*** y ***t*** □ $Dist_{st}$) y (***a*** y ***c*** □ d)

y cuál es el punto ***a***

(b) Ecuación de la recta que pasa por 2 puntos (***s*** y ***t***) .

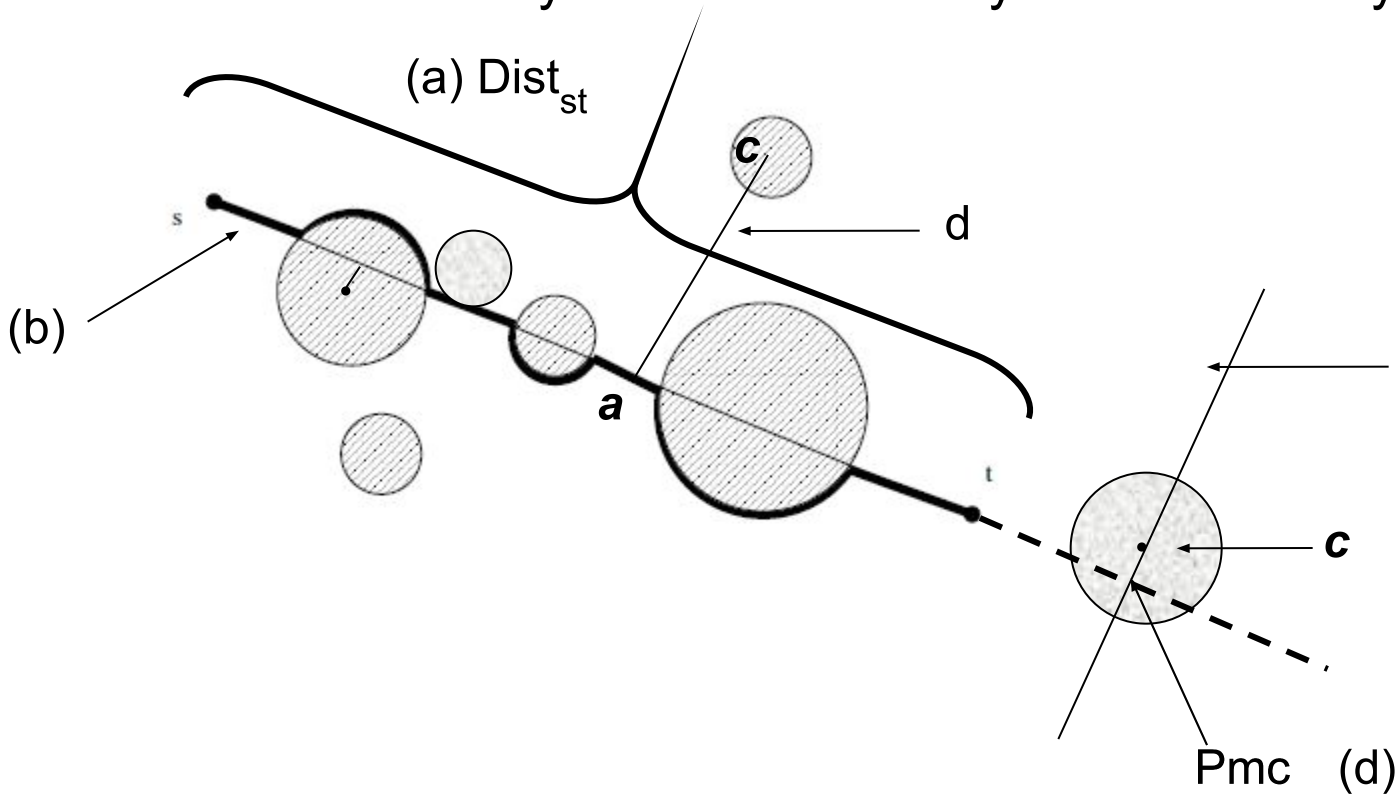
(c) Ecuación de la recta que pasa por 1 punto y es perpendicular a una dada (***centro círc. y (b)***).

(d) Punto de intersección entre 2 rectas. (Si las rectas son (b) y (c) □ Pmc y ***a***)

(e) Un punto (x,y) cae en el Box formado por las coordenadas (sx,sy) y (tx,ty)

Plan de vuelo de Superman

(1) Intersección entre un círculo y la línea ***l*** de la trayectoria entre ***s*** y ***t***

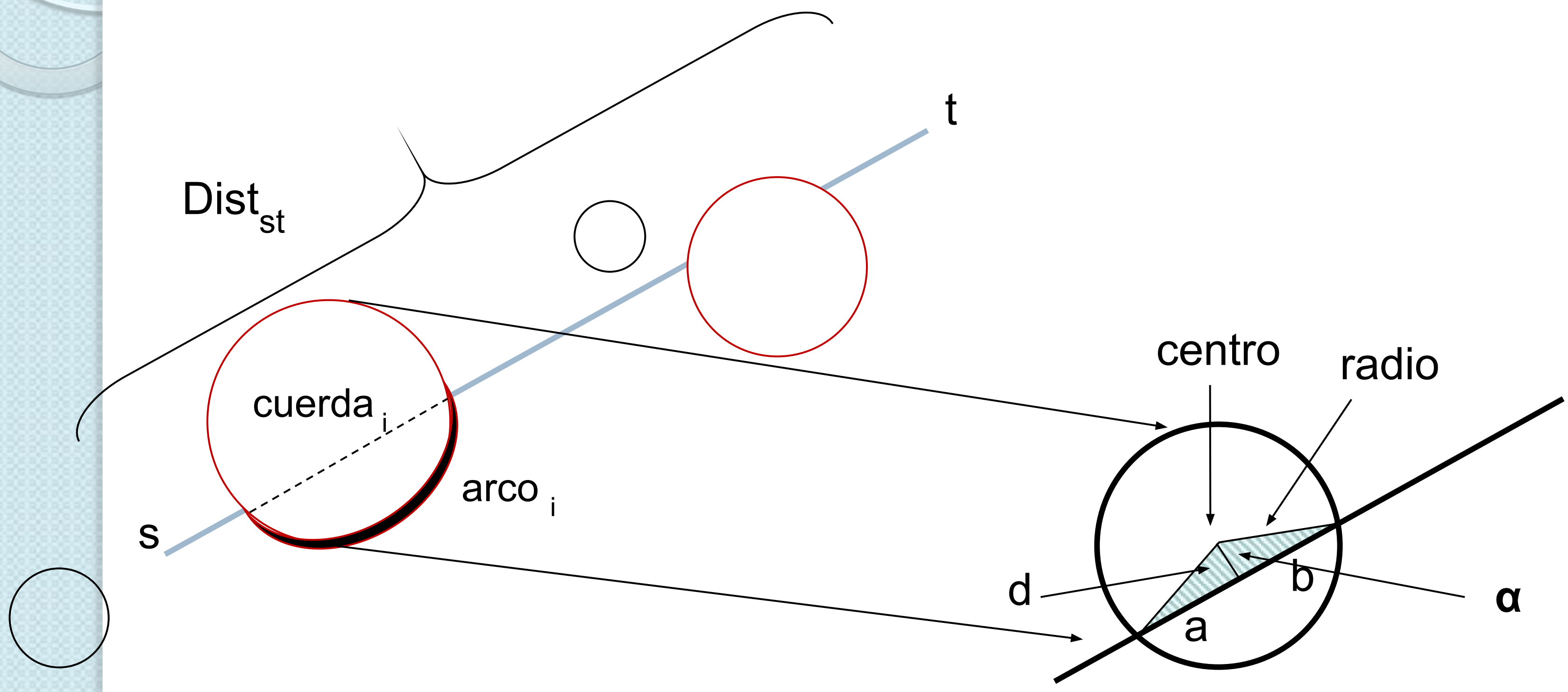


Conclusión:

Si d < radio de **Ci** y el Punto de intersección entre las 2 rectas cae en la trayectoria de los puntos **s** y **t** entonces tomo el círculo **Ci** . Existe intersección entre **Ci** y la trayectoria de Superman.

Plan de vuelo de Superman

(2) Cálculo de la longitud de la cuerda y (3) cálculo de la longitud del arco

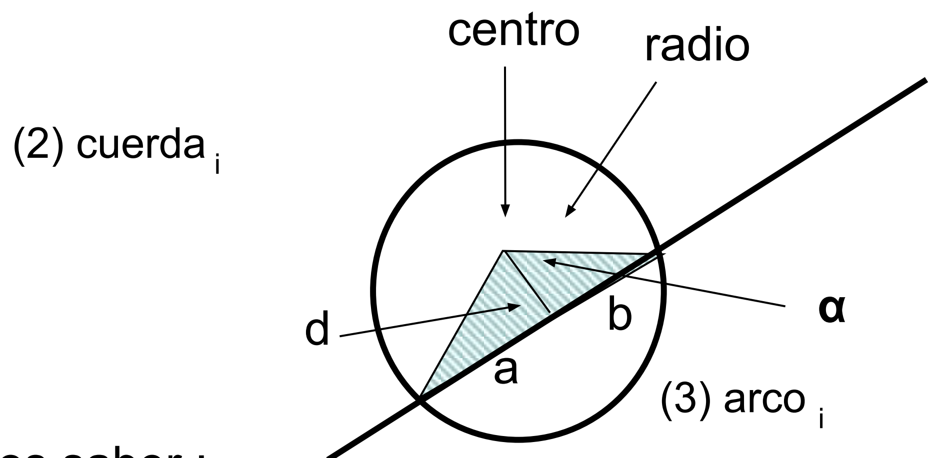


cuerda $_i$ = **a + b** ⇒ si conocemos **d,** aplicamos Pitágoras. **d = ?**
arco $_i$ = es la parte prop. del perímetro de la circ. corresp. al ángulo 2**α, α=??**

Dm = Dist$_{st}$+∑ { – cuerda $_i$ + arco $_i$ } ∀ círculo que interseca la recta st

Plan de vuelo de Superman

(2) Cálculo de la longitud de la cuerda y (3) cálculo de la longitud del arco



Resumiendo, necesitamos saber :

(2) Pitágoras : $r^2 = b^2 + d^2$; conocido "d" y "r" despejamos y calculamos **b** (tb **a**)

(3) Funciones trigonométricas para calcular **α**, es decir, cos **α** = d / r ; **α** = arc cos (d/r)

y para la parte prop. de la circunf corresp a 2**α** , tenemos que:

360 ° ---- > 2 π r (perímetro de la circunf)

2**α** ° ---- > x = 2**α**° 2 π r / 360° ---- > 2 **α** 2 π r / (2 π) ---> 2 **α** r

x = 2 * r * arc cos (d/r)

Plan de vuelo de Superman

```
point s;                  /* Superman's initial position */
point t;                  /* target position */
int ncircles;             /* number of circles */
circle c[MAXN];           /* circles data structure */

superman()
{
    line l;               /* line from start to target position */
    point close;          /* closest point */
    double d;             /* distance from circle-center */
    double xray = 0.0;    /* length of intersection with circles */
    double around = 0.0;  /* length around circular arcs */
    double angle;         /* angle subtended by arc */
    double travel;        /* total travel distance */
    int i;                /* counter */
    double asin(), sqrt();
    double distance();
```

continúa

Plan de vuelo de Superman

```
points_to_line(s,t,&l);

for (i=1; i<=ncircles; i++) {
        closest_point(c[i].c,l,close);
        d = distance(c[i].c,close);
        if ((d>=0) && (d < c[i].r) && point_in_box(close,s,t)) {
                xray += 2*sqrt(c[i].r*c[i].r - d*d);
                angle = acos(d/c[i].r);
                around += ((2*angle)/(2*PI)) * (2*PI*c[i].r);
                }
}

travel = distance(s,t) - xray + around;
printf("Superman sees thru %7.3lf units, and flies %7.3lf
                units\n", xray, travel);
}
```

Operaciones con Líneas rectas y puntos

- **distance(c[i].c,close)**
- **points_to_line(s,t,&l)**
  - Definir la recta que pasa por 2 Puntos
- **closest_point(c[i].c,l,close)**
  - Encontrar el Punto sobre la recta más cercano a un Punto dado
    - Definir la recta que pasa por un Punto y tiene Pendiente "m"
    - Encontrar el Punto de Intersección entre 2 rectas
- **point_in_box(close,s,t)**
  - Determinar si un punto (x,y) cae en el Box formado por las coordenadas (s_x,s_y) y (t_x,t_y)

Fin Caso de estudio