Guillermo Tell U52362_es

Guillermo Tell entrena su puntería lanzando flechas contra una diana. El centro de la diana es el punto $(0, 0)$ del plano, y cada flecha acaba clavada en un punto (x, y).

Escribe un programa que mantenga un *marcador* con los k tiros más cercanos al centro de la diana. El programa recibe dos enteros, k y M. Después lee una secuencia de puntos 2D (las posiciones de cada flecha). Cada M tiros, el programa debe mostrar el marcador con los k tiros más cercanos al centro, ordenados de más cercano a más lejano. Si dos tiros tienen la misma distancia al centro, va primero el que tiene la coordenada x más pequeña, y si coinciden, el que tiene la y más pequeña.

Si en algún momento hay menos de k tiros, el marcador muestra solo los que haya.

Si al final de la entrada la cantidad de tiros no es múltiplo de M, hay que mostrar el marcador una última vez.

Entrada

La primera línea contiene dos enteros k y M. Las líneas siguientes contienen dos reales x e y cada una, las coordenadas de cada tiro.

Salida

Cada M tiros, el marcador con los k tiros más cercanos al centro (o menos si no hay suficientes), uno por línea con el formato `(x, y)`. Entre dos tablas consecutivas, una línea con `---`.

Observación

El centro de interés en este problema es la **eficiencia**.

No es necesario calcular raíces cuadradas para comparar distancias al centro: $\sqrt{x^2 + y^2} < \sqrt{x'^2 + y'^2}$ es equivalente a $x^2 + y^2 < x'^2 + y'^2$.

Si usáis un `Heap`, pensad qué signo debe tener la prioridad para que el tiro "peor" de los k mejores esté siempre accesible en la cima.

En los archivos públicos (icono del gatito) encontrarás los contenedores de PRO2 (`stack.hh`, `queue.hh`, `heap.hh`, y su dependencia `assert.hh`), un `program.cc` vacío para empezar, un `Makefile` y el directorio `.vscode` con la configuración para compilar y depurar con VS-Code.

Implementa el programa en el archivo `program.cc` y envíalo al Jutge. Puedes usar cualquier contenedor de la STL y los de PRO2 que necesites.

Ejemplo de entrada 1

```
3 4
1.0 2.0
-0.5 0.3
3.0 4.0
0.1 -0.1
```

Ejemplo de salida 1

```
(0.1, -0.1)
(-0.5, 0.3)
(1, 2)
```

Ejemplo de entrada 2

```
5 3
1.0 1.0
-2.0 0.5
0.1 0.2
0.0 0.5
-1.0 -1.05
3.0 3.0
-0.1 0.1
```

Ejemplo de salida 2

```
(0.1, 0.2)
(1, 1)
(-2, 0.5)
---
(0.1, 0.2)
(0, 0.5)
(1, 1)
(-1, -1.05)
(-2, 0.5)
---
(-0.1, 0.1)
(0.1, 0.2)
(0, 0.5)
(1, 1)
(-1, -1.05)
```

Información del problema

Autoría: Pau Fernández
Traducción: Pau Fernández

Generación: 2026-04-06T16:55:29.887Z