OMIPS 2022 - Solución para *Esquinas cóncavas*

Entiende el problema

¿Qué te piden hacer? Karel empieza con su lado derecho pegado a la pared de una figura cerrada. Debe recorrer todo el contorno de la figura y contar las *esquinas cóncavas* que existen en la misma.

¿Qué tienes que entregar como resultado? Cómo resultado debes dejar un montón de zumbadores con el número de *esquinas cóncavas* en la figura.

¿Dónde tienes que entregar el resultado? Debes dejar el montón de zumbadores en la posición dónde empieza Karel.

Analiza el problema

¿Qué tienes que poder hacer para resolver el problema? 1. **Recorrer una figura usando su contorno:** Karel tiene que recorrer una figura, así que para este problema debes poder *seguir* el contorno de una figura. 1. **Darte cuenta que regresaste a la posición de inicio:** Es necesario que tengas una forma de darte cuenta que regresaste a la posición inicial, es decir, que ya acabaste de recorrer la figura. 1. **Detectar una *esquina cóncava*:** Debes poder diferenciar una *esquina cóncava* del resto del contorno de la figura. 1. **Contar las *esquinas cóncavas*:** Conforme vayas recorriendo la figura, debes ir contando **y recordando** las *esquinas cóncavas* por las que pasaste de modo que cuando llegues de nuevo a la posición inicial, puedas dejar el montón resultado.

Resuelve, uno por uno, los requerimientos del problema

Recorrer el contorno de la figura:

Karel empieza con la derecha pegada al contorno de modo que resulta sencillo recorrer la figura manteniendo siempre el contorno a la derecha de Karel. En cualquier momento pueden pasar alguna de las siguientes tres situaciones:

- Karel **NO** tiene pared a la derecha: Karel dee mantener la figura siempre a su derecha, si no tiene pared a la derecha debe *buscarla* de nuevo. Por lo que en ese caso debe girar a la derecha y avanzar.
- Karel tiene el frente-libre (y la derecha-bloqueada): En este caso basta con que Karel avance para seguir manteniendo el contorno a su derecha.
- Karel tiene el frente-bloqueado (y la derecha-bloqueada): En este caso Karel debe girar a la izquierda. No es necesario moverse, al girar-izquierda, automáticamente queda a su derecha el *siguiente* tramo de contorno.

▼ Haz clic para ver el código de Karel que recorre la figura manteniendo el contorno a su derecha.

```
define-nueva-instruccion recorre como inicio
    si derecha-libre entonces inicio
        repetir 3 veces gira-izquierda;
        avanza;
    fin
    sino inicio
        si frente-libre entonces avanza
        sino gira-izquierda;
    fin;
fin;
```

Darte cuenta que regresaste a la posición de inicio:

Aunque Karel no puede saber su ubicación exacta, sabes que si recorres la figura usando el código del punto anterior, eventualmente vas a pasar por la posición inicial. Una manera sencilla de darte cuenta de **cuál** es la posición inicial es *marcarla* con un zumbador. Es decir, dejar un zumbador ahí al momento de empezar.

Detectar una *esquina cóncava*:

Observando la definición de *esquina cóncava* del problema, puedes darte cuenta que al recorrer el contorno, las

esquinas cóncavas son aquellas en las que Karel tiene el frente y la derecha bloqueadas. De modo que esa es la forma más sencilla de detectarlas.

Contar las *esquinas cóncavas*:

Karel debe ir contando y *recordando* las esquinas cóncavas por las que pasa. Karel puede *recordar* cosas usando la variable de una función o utilizando la pila recursiva. Suele ser un poco más sencillo hacerlo con la pila recursiva. Una forma de hacerlo es la siguiente:

1. Karel *marca* la posición de inicio y avanza.
2. Karel llamará recursivamente a la función *recorre* para recorrer todo el contorno.
3. Antes de cada llamada a *recorre* valida si está en una esquina cóncava, en caso de estar, al final de la llamada (en el regreso de la recursión) dejará un zumbador de resultado.
4. Al llegar de nuevo a la posición de inicio y dejar el resultado, recoge el zumbador que dejaste inicialmente.

▼ Haz clic para ver el código.

```
iniciar-programa
    define-nueva-instruccion recorre como inicio
        si derecha-libre entonces inicio
            repetir 3 veces gira-izquierda;
            avanza;
        fin
        sino inicio
            si frente-libre entonces avanza
            sino gira-izquierda;
        fin;
    fin;

    define-nueva-instruccion resuelve como inicio
        si no junto-a-zumbador entonces inicio
            si frente-bloqueado y derecha-bloqueada entonces inicio
                recorre;
                resuelve;
                deja-zumbador;
            fin
            sino inicio
                recorre;
                resuelve;
            fin;
        fin;
    fin;

    inicia-ejecucion
        deja-zumbador;
        recorre;
        resuelve;
        coge-zumbador;
        apagate;
    termina-ejecucion
finalizar-programa
```

Prueba tu código (casos especiales)

Si envías el código de la sección anterior a OmegaUp obtendrás únicamente 46 puntos. ¿Qué está faltando?

Un caso en el que el codigo anterior falla es si Karel empieza junto a una *esquina cóncava*. Recuerda que el código inicia dejando un zumbador y llamando *recorre* para avanzar al siguiente *tramo* del contorno. Pero cuando Karel está en una *esquina cóncava* no avanza, simplemente gira. Esto hace que Karel *crea* que ya regresó a la posición inicial. De modo que hay que resolver esa situación, Karel debe *marcar* la posición de inicio y avanzar hasta una posición en la que esté seguro que no hay zumbador.

Una forma de hacerlo es girar a la izquierda hasta no estar en una *esquinas cóncavas*, pero recuerda que esas esquinas cóncavas también debes contarlas, de modo que conforme las vayas avanzando debes irlas *contando*, lo bueno es que estás en la posición del resultado, así que las puedes dejar directamente ahí. Si agregas el código:

```
    mientras frente-bloqueado y derecha-bloqueada hacer inicio
        gira-izquierda;
        deja-zumbador;
    fin;
```

al inicio de la ejecución del programa y envías el código verás que ahora obtienes 73 puntos.

Sigue habiendo un caso especial que está faltando ver.

No hay una técnica específica para darse cuenta de los casos especiales, se requiere práctica y creatividad para imaginar situaciones que no te especifican en el problema.

En este problema, el caso especial sucede cuando la figura tiene un *pasillo* de ancho 1 entre dos lados de su contorno y Karel empieza en ese pasillo. Piensa en la letra "E" de la figura de ejemplo del problema en su *hueco* superior. Si Karel inicia en se hueco y sigue todo el contorno verás que pasa de nuevo por la posición inicial antes de haber recorrido toda la figura completa y por lo tanto, su cuenta puede estar incompleta.

¿Cómo se puede resolver ese caso? Si analizas con cuidado esa situación verás que las veces que Karel pasa por la posición del zumbador sin haber recorrido toda la figura lo hace con una orientación distinta. De modo que para determinar que dio toda la vuelta, Karel debe de tomar en cuenta dos cosas:

- Que llegó a la posición del zumbador.
- Que llegó con la misma dirección con la que salió.

¿Cómo puede Karel saber con que dirección salió? Una forma de hacerlo es utiilzar la variable para indicar la dirección. Por ejemplo, un **0** puede significar norte, un **1** este, etc.

Agregando esa validación a `resuelve` el problema da 100 puntos.

OJO: Observa que si Karel empieza en una *esquina cóncava* con el código anterior y esta solución se contarán doble, así que no es necesario dejar los zumbadores iniciales.

▼ Haz clic para ver el código.

```
iniciar-programa
    define-nueva-instruccion recorre como inicio
        si derecha-libre entonces inicio
            repetir 3 veces gira-izquierda;
            avanza;
        fin
        sino inicio
            si frente-libre entonces avanza
            sino gira-izquierda;
        fin;
    fin;

    define-nueva-instruccion resuelve(n) como inicio
        si no junto-a-zumbador o
           (junto-a-zumbador y si-es-cero(n) y no orientado-al-norte) o
           (junto-a-zumbador y si-es-cero(precede(n)) y no orientado-al-este) o
           (junto-a-zumbador y si-es-cero(precede(precede(n))) y no orientado-al-sur) o
           (junto-a-zumbador y si-es-cero(precede(precede(precede(n)))) y no orientado-al-oe
ste)
           entonces inicio
            si frente-bloqueado y derecha-bloqueada entonces inicio
                recorre;
                resuelve(n);
                deja-zumbador;
            fin
            sino inicio
                recorre;
                resuelve(n);
            fin;
        fin;
    fin;

    inicia-ejecucion
        mientras frente-bloqueado y derecha-bloqueada hacer gira-izquierda;
        deja-zumbador;
        si orientado-al-norte entonces inicio
            recorre;
            resuelve(0);
        fin
        sino inicio
            si orientado-al-este entonces inicio
                recorre;
                resuelve(1);
            fin
            sino inicio
                si orientado-al-sur entonces inicio
                    recorre;
                    resuelve(2);
                fin
                sino inicio
                    recorre;
                    resuelve(3);
                fin;
            fin;
        fin;
        coge-zumbador;
        apagate;
    termina-ejecucion
finalizar-programa
```