Problema C

0 Identificación

Daniel Santiago Tenjo 201815459
Santiago Estupiñan 201813414

1 Algoritmo de solución

El algoritmo de solución que elegimos se basa en crear una línea horizontal hasta el infinito desde el punto que se desea conocer si esta adentro o no del polígono. Se basa en una explicación hallada en GeeksForGeeks.
Si tal línea se interseca un numero impar de veces con los arcos que forma el polígono, es porque tal punto esta dentro de este. Si es intersecado un numero par, es porque no esta dentro del polígono. Aquí también se tuvieron en cuenta casos especiales como que en el cual la línea creada solo interseque algún vértice del polígono, pues aparecería como impar su numero de intersecciones, pero en verdad está afuera del polígono.

El algoritmo esta divido en 4 subrutinas. La primera *estaEnLinea* verifica que un punto este contenido dentro de un segmento, con una simple operación con máximos y minimos entre los puntos que entran por parámetro.
La segunda subrutina se llama *orientación* y se encarga de, a partir de tres puntos, verificar la orientación de estos. Esto resulta útil para verificar si dos segmentos se intersecan pues, si sí lo hacen, la orientación será diferente para cada uno los dos puntos de un segmento manteniendo los otros dos puntos del otro segmento fijos. Ello se ve mejor en la figura 1, donde los puntos del segmento rojo se mantienen fijos mientras que los dos puntos del segmento negro se intercambian. Ello da como resultado diferentes orientaciones para los recorridos amarillo y verde. Todo lo anterior se hace con un pequeño calculo tomado de GeeksForGeeks que es realizado con multiplicaciones y restas de las coordenadas de los puntos.
Ctx C: px, py, qx, qy, rx, ry; int //Coordenadas de los puntos a revisar
Pre Q: true
Pos R: rta:int ∧ rta∈{0, 1, 2} //0 si son colineales, 1 para sentido horario, dos para lo contrario

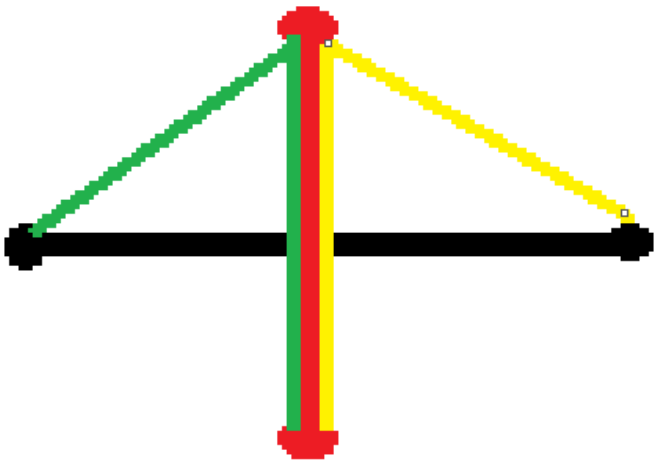

Figura 1

La tercera subrituna es llamada *seInterseca* y calcula, mediante una combinación de las anteriores dos subrutinas, si en definitiva dos segmentos se intersecan. Para ello revisa casos generales y especiales de los resultados de orientaciones y de si están en la línea.

Finalmente, la cuarta y principal subrutina es llamada *estaAdentro*. En ella, para cada arco del polígono, se verifica si es cortado por el segmento desde el punto a analizar hasta el infinito en el eje x. Si tal cuenta es impar, se determina que el punto esta adentro y, si no, esta afuera. En este método también se analiza si el punto esta en el borde (es colinear y esta adentro de algún segmento del poligono).
Ctx C: M, N:nat $\wedge$ $1 \leq M \leq 10^3 \wedge 3 \leq N \leq M^2 \wedge$ F[0..2*N] = [f(0),…,f(N)]
Pre Q: true
Pos R: rta:int $\wedge$ rta$\in${-1, 0, 1}

2 Análisis de complejidades espacial y temporal
T(n) = O(estaAdentro)*O(seInterseca)*O(orientación)*O(estaEnLinea)
O(n)*O(1)*O(1)*O(1) //estaAdentro se realiza para los n vértices del polígono
O(n)
S(n) = O(n) //Arreglo donde se guardan las coordenadas de los puntos del poligono

3 Comentarios finales
Después de realizar la solución planteada, se hicieron alrededor de 30 casos de prueba para verificar los posibles casos. Ello arrojó que habían ciertos casos(alrededor de 3 de los probados) en los que no se retornaba la respuesta correcta, especialmente en aquellos donde se debería retornar que se estaba en el borde o adentro del polígono. Se trató de identificar los errores u omisiones de casos de borde que pudieran provocar tal error, pero no fue posible hallarlos. En cuanto a desempeño general, omitiendo tales anomalías, el algoritmo se comporta muy bien en tiempo.