Problema C

Juan Felipe Rubio Perdomo – 201718384
Jf.rubio@uniandes.edu.co
Juan Andrés Avelino Ospina – 201812676
ja.avelino@uniandes.edu.co

I Algoritmo de solución:

Para este problema, se desarrolló su solución dentro del main de la clase java. El problema se resolvió con un algoritmo que se basa en la geometría enfocada a algoritmos. A través de internet encontramos diferentes soluciones que abordaban el tema de diferentes modos. El que nos pareció más acorde a lo esperado fue aquel que trazaba líneas para determinar si un punto intersecaba un número de veces impar a los bordes de la figura.

Para visualizar mejor la problemática del problema es produnte analizar la siguiente imagen:

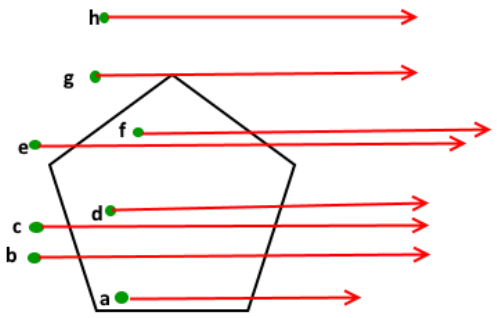

Imagen tomada de: GeeksForGeeks
En esta imagen se puede analizar como el número de veces que se intersecan define si un punto está dentro o no de una figura.
Sin embargo, este algoritmo tiene ciertos problemas a la hora de reconocer los puntos que están en contacto con los vértices y por eso se requiere más condiciones.

Anotación del desarrollo del método de resolución del problema: (esto se hizo basado en todo el código realizado en el main)

Contexto: A lo largo de toda la ejecución del problema se tendrá un arreglo de `Puntos` que recibirá los datos ingresados desde consola y separará los valores haciendo uso del `split` por el criterio de "". Por otro lado, se tendrán las variables iniciales para hacer la construcción de la figura y lograr determinar si un punto está dentro o no de la figura. Por otro lado, se tendrá la variable `n` que contendrá el tamaño y el arreglo `Poligono` que tendrá los valores del poligono. Sin embargo, es en `estaDentro` dónde se realiza todo el manejo de datos.

Precondición: Al inicio de la ejecución se tendrá un arreglo con valores ingresados por consola y de tamaño n definido también desde consola.Así mismo se tendrán todas las variables explicadas anteriormente para ser tratadas a lo largo del programa. Se verán así:

```
Punto[]poligono = new Punto[];
String[] data = br.readLine().split(" ");

int n = Integer.parseInt(data[0]);
int m = Integer.parseInt(data[1]);
int a = Integer.parseInt(data[2]);
int b = Integer.parseInt(data[3]);
```

Cota: El programa no tiene ninguna cota específica sin embargo, se puede considerar como cota el número de lados que tiene el poligono debido a que analiza cada uno de ellos para determinar si está o no dentro de la figura.

Avance y Ejecución: El avance en el recorrido será i—pues se avanza en las posiciones de los lados de la figura.

Postcondición: Se espera que al final de la ejecución se sepa si un punto está o no contenido en una figura. Esta respuesta sera interpretada según 3 valores difrentes en {1,0,-1} siendo 1 la especificación que está estrictamente adentro de la figura, 0 significa que el punto está en el borde y -1 significando que está fuera de la figura.

II Análisis de complejidades espacial y temporal:

Una vez implementado el progama, se procedió a calcular la complejidad temporal y espacial de la solución teniendo en cuenta los algoritmos utilizados y la manera en que se resolvió el problema, obteniendo lo siguiente:

$T(n) = O(n)$

$S(n) = O(1)$

Esta complejidad se calcula analizando cuantas veces se opera sobre el valor n. Para este caso en específico solo se recorre una vez sobre todos los lados y con esto ya es suficiente para determinar si un punto hace parte o no a una figura. Es por esto, que la complejidad de este algoritmo es lineal y se ejecuta con una velocidad alta.
Para tener en cuenta, la complejidad espacial es sencilla de calcular ya que en ningún momento se crea un arreglo aparte del poligono que nos entra por parámetros. Es por esto, que la complejidad es $O(1)$

III Comentarios finales:

No funcionan todos los casos de prueba. Sin embargo, no dimos con las soluciones previstas para estos casos especiales.