Diseño y Análisis de Algoritmos - Proyecto
PROBLEMA C

Sebastian Camilo Puerto
201318518

Diciembre 10 , 2019

1. Algoritmo de Solución

IDEA GENERAL: Se itera sobre las aristas del polígono para determinar para $P = (a, b)$ su *Crossing Number* (el n[umero de veces que una semirecta cualquiera con origen en P cruza por alguna arista del pol[igono); al tiempo se determina si P pertenece a la arista actual, lo cual, de ser afirmativa la respuesta, termina el programa para el test actual. Al haberse recurrido todas las aristas, si la semirrecta cruzó por un número impar de aristas se concluye que el punto está en el interior del polígono, o que está en el exterior de caso contrario.

El algoritmo que soluciona el problema es un ciclo for que itera sobre los vértices, y utiliza la porción derecha de la recta $y = b$ que empieza en el punto $P = (a, b)$.

Lenguaje de descricipción de las anotaciones:

- $SR(P)$: la porción derecha de la recta $y = b$ que empieza en $P = (a, b)$.
- $\{V_0, V_1, \ldots, V_{N-1}\}$ los vértices ordenados que llegan por entrada. V_N se define como V_0.
- $ea(j, P)$: el punto P pertenece al segmento $V_{j-1}V_j$.
- $cruza(j, P)$: la semirecta $SR(P)$ corta de forma no tangente a la arista $V_{j-1}V_j$.
- $enPerim(P)$: el punto P se encuentra en el borde del polígono. Notamos que $enPerim(P) = \bigvee_{j=1}^{N} ea(j, P)$
- $CN(P)$: el número de veces que la semirecta $SP(P)$ cruza de forma no tangente el borde del polígono. Notamos que $CN(P) = \sum_{j=1}^{N} cruza(j, P)$.

Como *contexto* del problema se tienen las hipótesis dadas en el enunciado, además de que hay 2 arreglos $x : int[N+1]$, $y : int[N+1]$ que contienen las coorenadas x y y de cada vértice, además de las variables $crossings : int, crossings = 0$, $enArista : bool, enArista = False$ y $respuesta : int$.

Se puede comentar que se investigó también la técnica del *Winding Number* del punto, la cual cuenta el número de veces que las aristas del polinomio rodean al punto, de donde se determina que el punto está siempre y cuando el número de vueltas sea entero e impar. Sin embargo, aunque hay variaciones de esta técnica que son eficientes, para el problema presenta era más claro, conciso y eficiente utilizar el crossing number del punto.

2. Análisis de Complejidades Espacial y Temporal

El tamaño de la entrada es N, el número de vértices.

2.1. Análisis Espacial

Hay algunas variables como *crossings*, *enArista*, *respuesta* que tienen un tamaño constante por lo cual contribuyen un total de $O(1)$ a la complejidad espacial. Los arreglos x y y tienen tamaño $N + 1$, cada una contribuye $O(N)$ a la complejidad espacial. En conclusión, la complejidad espacial es la suma de éstas, y es simplemente:

$$S_C(N) = O(N)$$

2.2. Análisis Temporal

Podemos tomar las operaciones aritméticas y de comparación como operaciones básicas para contabilizar la complejidad temporal.

Determinar si el punto pertenece a la arista actual se con un número constante de operaciones: $T_{\in} = O(1)$. Determinar si la semirecta cruza a la arista actual y actualizar el valor de *crossings* también se hace con un número constante de operaciones, de modo que $T_{cruz} = O(1)$. Estas son las 2 rutinas que se ejecutan como cuerpo del ciclo, así que la complejidad de una iteración es $T_{iterac} = O(1)$

Como el método es solucionado en un ciclo cuya cota disminuye en al menos 1 con cada iteración, y se empieza desde la cota N, la complejidad total del algoritmo de solución del problema C es

$$T_C(N) = N\,O(1) = O(N)$$

3. Comentarios Finales

Vale la pena recalcar que en la solución no se hace uso del dominio M. En los ejemplos que se realizaron el tiempo de ejecución del problema fue siempre significativamente menor al segundo.