ALGORITMOS PARA LA LOCALIZACIÓN DE ABEJAS ROBÓTICAS Y EVITAR COLISIÓN

Juliana Henao Arroyave
Gerónimo Zuluaga Londoño
Medellín, 6 de noviembre

Estructuras de Datos Diseñada

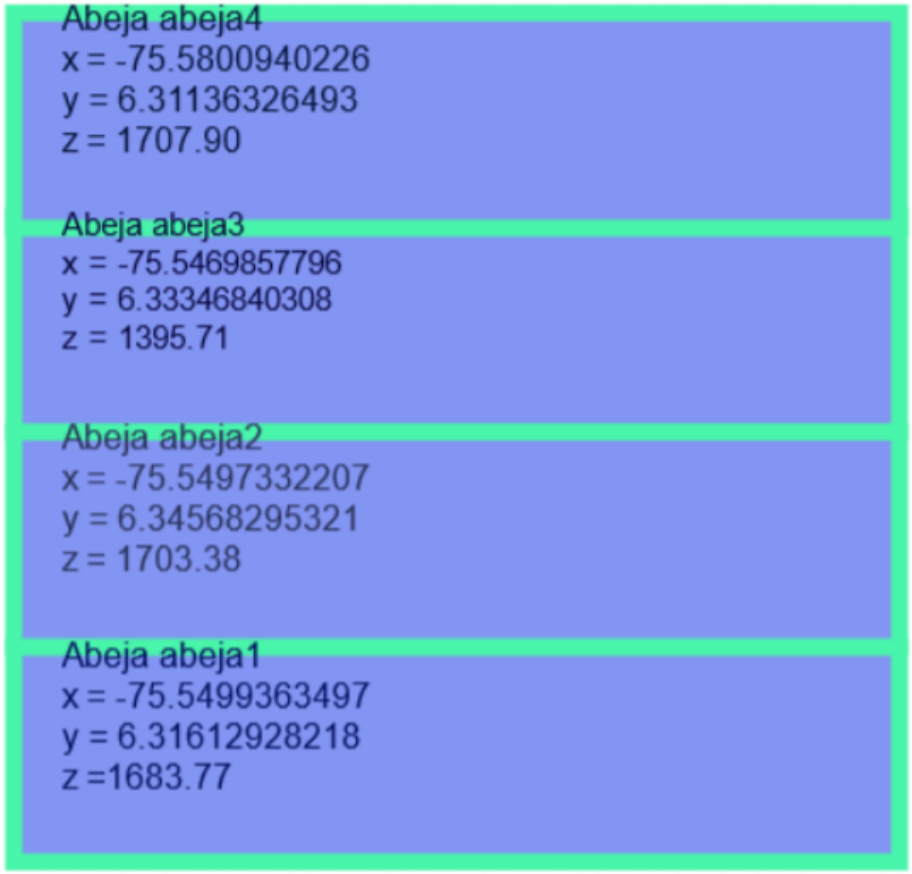

Gráfico 1: Una pila o Stack de clase *Abeja*. *Abeja* es una clase con atributos de doubles *x*, *y* y *z*

Operaciones de la Estructura de Datos

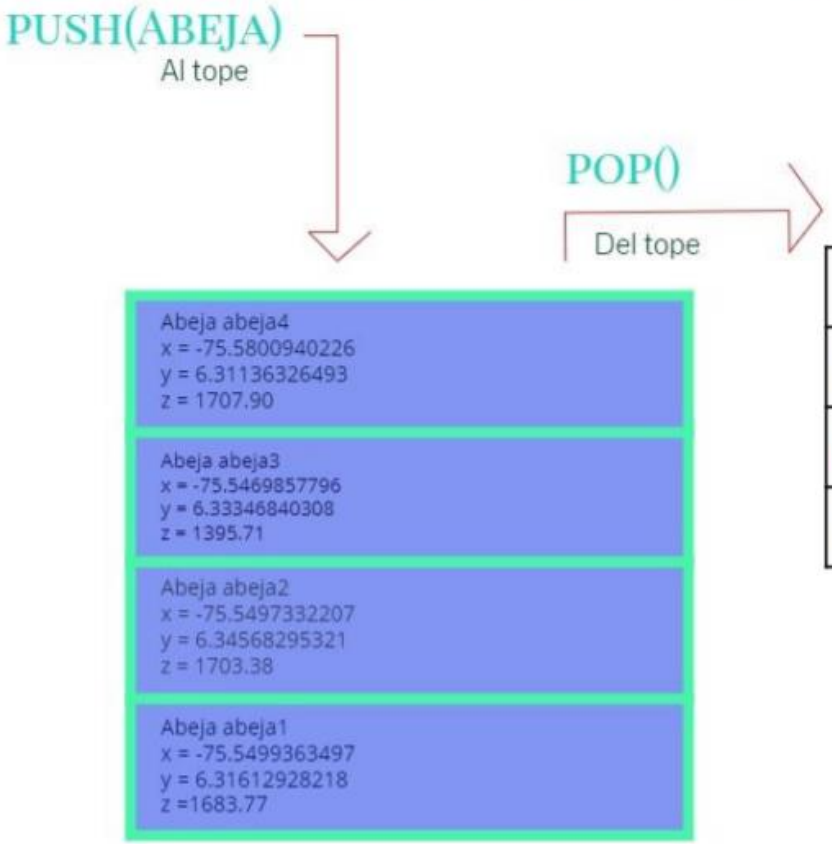

Gráfico 2: *Pop* es un método que pone una Abeja al tope de la pila y *push* añade una en el tope. El método *collisionDetector* utiliza *pop*

	Mejor caso	Peor caso
Pop	O(1)	O(1)
collisionDetector	$O(n^2)$	$O(n^2)$
Push	O(1)	O(1)

Tabla 1: Complejidad de las operaciones de la estructura de datos (pop y push) y del método para detector colisiones

Criterios de Diseño de la Estructura de Datos

- En la solución del problema se requiere usar el método pop para comparar la abeja que se retorna.
- La operación de pop en la estructura de datos Stack tiene complejidad O(1) en cualquier caso
- La operación quitar un elemento y retornarlo tiene complejidad en memoria de O(n)

- La complejidad de la operación para saber y borrar el elemento del tope de la pila es eficiente para el problema.

Consumo de Tiempo y Memoria

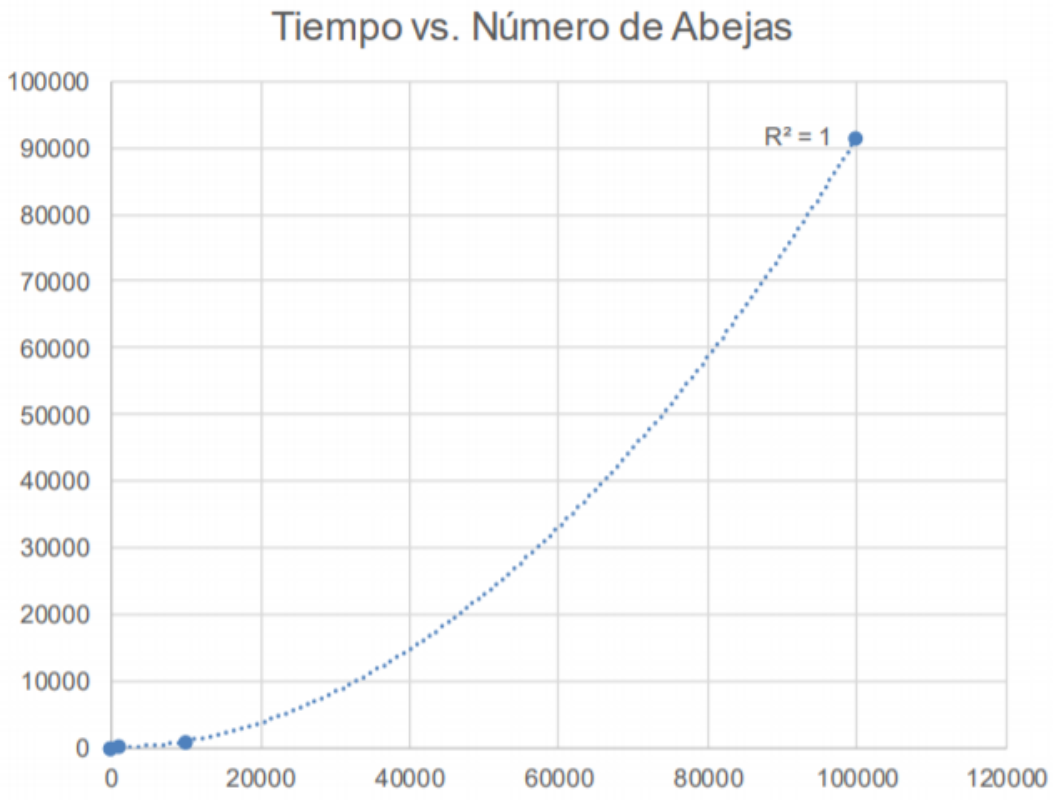

Gráfico 3: Gráfico del tiempo que demora en arrojar todas las respuestas. Es polinómica de grado 2 y se evidencia en la imagen.

Número de abejas	Tiempo Con Stack	Tiempo con LinkedList
10	0 ms	1 ms
100	1 ms	4 ms
1.000	2 ms	28 ms
10.000	2 ms	904 ms
100.000	17 ms	

Tabla 2: Tiempo en milisegundos para cada número de abejas a evaluar

Software Desarrollado

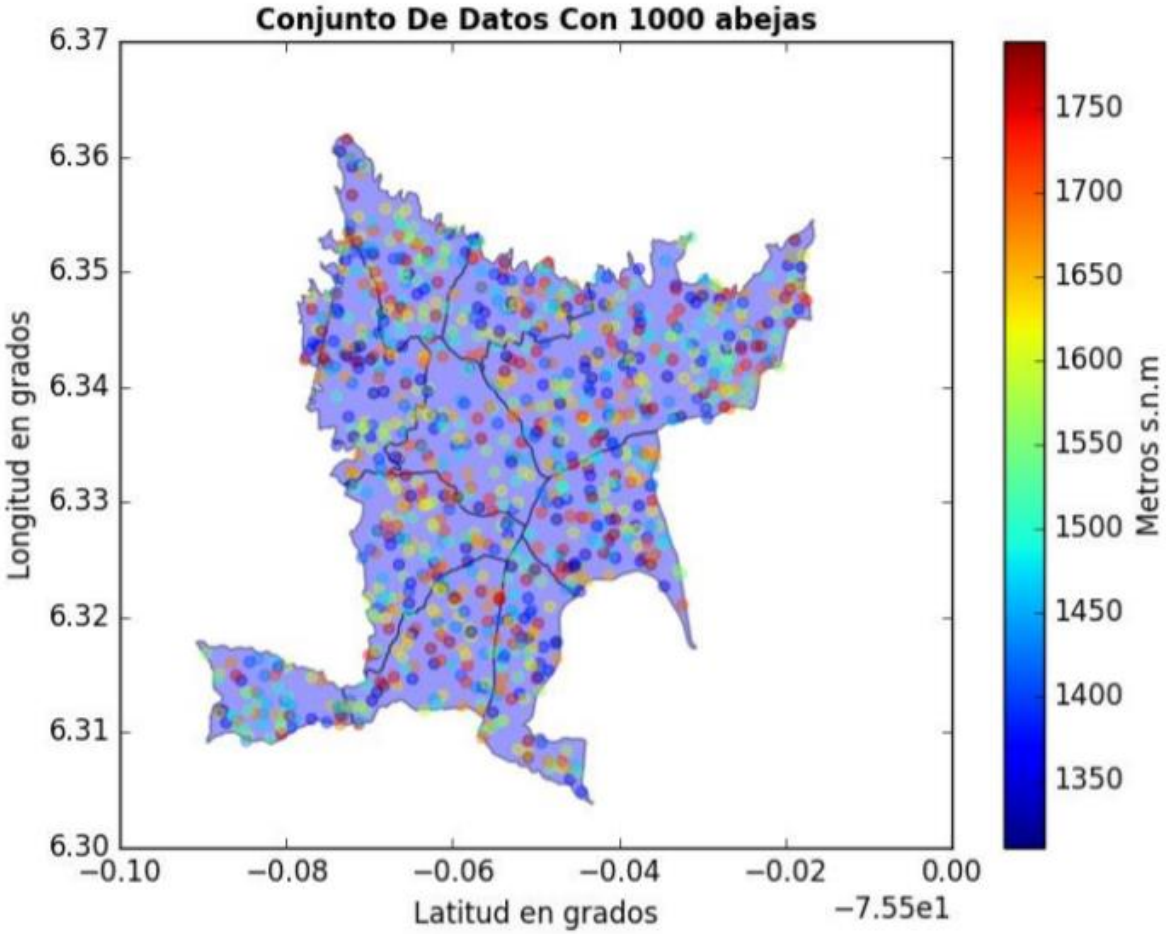

Gráfico 4: Mapa con 1000 abejas robóticas funcionando en Bello