Práctica 2

Estructura de datos

Grupo 201 GII

María Guzmán Valdezate

Guillermo López de Arechavaleta Zapatero

Contenido

[Descripción y Análisis de Métodos 3](#_Toc192930990)

[Pruebas y gráficas 4](#_Toc192930991)

[Conclusiones 6](#_Toc192930992)

## Descripción y Análisis de Métodos

**clear ()**

Descripción: Elimina todos los elementos de la cola, dejando la cola vacía.

Complejidad algorítmica: No hay bucles en este método, solo modifica las referencias a los nodos y establece el tamaño en 0. Tiene O (1).

**element ()**

Descripción: Devuelve el primer elemento de la cola sin eliminarlo.

Complejidad algorítmica: Este método solo accede al primer nodo de la cola, tiene O (1).

**contains ()**

Descripción: Comprueba si un elemento se encuentra en la cola.

Complejidad algorítmica: Debido al bucle do-while es de O (n).

**circularIterator ()**

Descripción: Devuelve un iterador circular que recorre la cola de forma continua.

Complejidad algorítmica: El iterador solo recorre los nodos de un ciclo.

**hasNext ()**

Descripción: Comprueba si el elemento tiene un siguiente elemento en la cola.

Complejidad algorítmica: Solo comprueba la primera vuelta, por lo que es O (1).

**next ()**

Descripción: Devuelve el siguiente elemento de la cola.

Complejidad algorítmica: Solo accede al nodo actual y avanza al siguiente, por lo que es O (1).

**offer (T e)**

Descripción: Inserta un elemento en la cola.

Complejidad algorítmica: Actualiza las referencias a los nodos e inserta un nuevo nodo al final de la cola. Es O (1).

**poll ()**

Descripción: Elimina el primer nodo de la cola y lo devuelve.

Complejidad algorítmica: Solo modifica referencias, es O (1).

**peek ()**

Descripción: Devuelve el primer elemento de la cola sin eliminarlo.

Complejidad algorítmica: Solo accede al primer elemento, por lo que es O (1).

**iterator ()**

Descripción: Devuelve un iterador sobre los elementos de la cola.

Complejidad algorítmica: Solo crea un iterador que se inicializa con el primer nodo, por lo cual es O (1).

**size ()**

Descripción: Devuelve el tamaño de la cola.

Complejidad algorítmica: Utiliza el método size que simplemente, va almacenando el número de elementos de la cola. Es de O (1).

**remove ()**

Descripción: Elimina el último elemento devuelto por el iterador.

Complejidad algorítmica: Solo actualiza las referencias al eliminar un elemento, por lo cual es O (1).

## Conclusiones

Podemos ver que la mayoría de las operaciones tienen O (1), lo que significa que tienen una muy buena eficiencia. Muchos de estos métodos tienen esta complejidad porque simplemente modifican referencias o acceden directamente a los nodos sin recorrer la cola.

El método contains () tiene una complejidad de O (n) porque tiene que recorrer la estructura para poder verificar si hay un elemento en la cola, lo que implica que en el peor de los casos tiene que recorrer todos los elementos de la cola.

Podemos observar que con esta implementación de la cola podemos hacer operaciones de manera rápida y eficiente.