INTEGRANTES DA EQUIPE

- Roberto Ferreira de Assis Filho (@robertoferreira7)
- Rodrigo de Oliveira Farias (@rodrigo-farias10)

COMENTÁRIOS DA EQUIPE

O código da classe SetWithQueue desenvolvido atende ao desafio proposto, usando a classe FilaArray como estrutura para armazenar elementos de forma ordenada e garantir que não haja duplicatas. A função add(element) assegura a ausência de duplicidade ao verificar se o elemento já está presente antes de enfileirá-lo, usando o método contains(element), que faz a busca linear pelo elemento na fila. A operação remove(element) retira o elemento especificado, criando uma fila temporária para manter a ordem dos elementos restantes, mantendo a integridade da estrutura. A operação contains(element) permite verificar a existência de um elemento no conjunto, enquanto size() retorna o número total de elementos e list() devolve todos os elementos armazenados em ordem, sem modificá-los. Essas operações garantem que o conjunto funcione conforme especificado, mantendo o comportamento e as características de um conjunto, mas com as características de uso de uma fila como estrutura de armazenamento.

O código apresenta uma implementação funcional, porém enfrenta alguns problemas de eficiência e limitações. A remoção de elementos, que preserva a ordem original, requer a criação de uma fila temporária, resultando em uma complexidade $O(n)$ para a operação de dequeue, o que pode ser ineficiente para grandes conjuntos. Além disso, a verificação de duplicidade através do método contains também é linear, $O(n)$, o que se contrapõe com a eficiência $O(1)$ esperada em estruturas de dados tradicionais de conjunto, como o set em Python. Embora o código inclua métodos essenciais como add, remove, contains, size e list, ele poderia ser melhorado, por exemplo, com um método para facilitar a iteração sobre os elementos. Esses pontos sugerem que, embora a implementação atenda aos requisitos básicos e passe nos testes fornecidos, ela pode não ser a mais eficiente ou adequada para cenários que exijam alta performance.

ATIVIDADE 01: IMPLEMENTAR SetWihQueue

A implementação de SetWithQueue está no repositório https://github.com/robertoferreira7/mini_projeto_filas.git no arquivo projeto_fila.py.

ATIVIDADE 02: DISCUSSÃO SOBRE DESEMPENHO

Para analisar a complexidade das funções na classe SetWithQueue, deve-se considerar a implementação de cada método e como ele interage com a estrutura FilaArray. Sendo FilaArray uma fila que utiliza uma lista para armazenar os elementos, a seguir encontra-se a complexidade de cada método desenvolvido na classe SetWithQueue.

Complexidade de cada método:

add(element): inicialmente, o método chama contains(element), que verifica se o elemento já está na fila. Essa operação tem complexidade O(n), pois percorre a lista para encontrar o elemento. Se o elemento não estiver presente, o método chama enqueue(element) para adicionar o elemento ao final da fila, que é uma operação O(1). Logo, a complexidade de add(element) é O(n) devido à chamada para contains.

remove(element): o método chama contains(element) para verificar se o elemento está na fila, o que é O(n). Se o elemento estiver presente, ele cria uma fila temporária (temp_queue) e usa um loop para realizar o dequeue de todos os elementos da fila original e reenfileirar aqueles que não correspondem ao elemento a ser removido. O dequeue() e enqueue() são ambos O(1) em uma lista. Como percorre todos os elementos da fila, a complexidade desse loop é O(n), resultando em uma complexidade total de O(n) para o método remove(element).

contains(element): este método percorre todos os elementos na fila subjacente para verificar a presença do elemento. A complexidade é O(n).

size(): este método simplesmente retorna o tamanho da lista subjacente da fila. Sendo size() no FilaArray uma operação constante, a complexidade é O(1).

list(): a complexidade de list() na classe SetWithQueue é O(1), já que ele retorna diretamente a lista sem realizar qualquer processamento adicional.

Os métodos que envolvem a verificação de elementos na fila, como add, remove e contains, têm complexidade O(n), pois precisam percorrer a lista para localizar ou remover elementos. Já size e list são operações rápidas, com complexidade O(1).

CONTEÚDO CONSULTADO PARA A REALIZAÇÃO DO PROJETO

- https://www.ufsm.br/pet/sistemas-de-informacao/2020/04/01/entendendo-listas-pilhas-e-filas
- http://sites.poli.usp.br/p/fabio.cozman/Didatico/Comp/Material/estruturas.pdf
- https://docente.ifsc.edu.br/vilson.junior/ed/04_Listas_Filas_Pilhas.pdf
- https://pessoal.dainf.ct.utfpr.edu.br/maurofonseca/lib/exe/fetch.php?media=cursos:if63c:if63ced_03_pilhasfilas.pdf