University of Tehran
College of Engineering
Fouman Faculty of Engineering

Project 1

Data Structure Teacher Assistance Project
Computer Engineering

Alireza Mahdizadeh

DEC 2023

Question:

A dynamic array excels in implementing a Sequence interface with robust support for worst-case O(1)-time indexing. Additionally, it efficiently handles the insertion and removal of items at the array's back in constant amortized time. However, the efficiency falters when it comes to operations at the array's front, necessitating the shifting of every entry to maintain sequential order and incurring a linear time cost.

Conversely, a linked-list data structure offers the advantage of supporting insertion and removal operations at both ends in a worst-case O(1) time frame, as exemplified in PS1. This advantage, however, comes at the trade-off of linear-time indexing.

The challenge is to harmonize these trade-offs and devise a data structure that seamlessly combines the strengths of both worlds. The objective is to create a solution that ensures worst-case O(1)-time index lookup while also accommodating amortized O(1)-time insertion and removal at both ends. Impressively, this can be accomplished while maintaining space efficiency, utilizing O(n) space to store n items. Such a meticulously crafted data structure strikes a balance, offering optimal performance for storing and managing sequences of items.

Solution:

A dynamic array stands out as a versatile data structure, seamlessly facilitating a Sequence interface with the added benefit of worst-case O(1)-time indexing. Its prowess extends to the realm of array manipulation, allowing for the efficient insertion and removal of items at the back of the array in amortized constant time. This exceptional performance, however, encounters a bottleneck when confronted with the task of inserting or deleting items at the front. The sequential nature of the array demands the shifting of every entry, resulting in a linear time complexity for these operations.

On the flip side, a linked-list data structure presents an alternative with its capability to support insertion and removal operations at both ends in a worst-case O(1) time frame, as highlighted in PS1. Yet, this advantage comes at the expense of linear-time indexing, posing a trade-off between efficient index lookup and versatile insertion and removal capabilities.

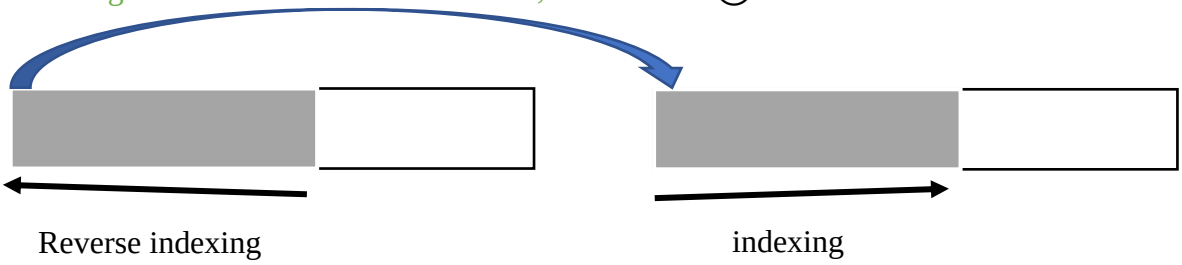
The challenge lies in reconciling these trade-offs to create a data structure that harmoniously combines the best attributes of both worlds. The goal is to craft a solution that not only ensures worst-case O(1)-time index lookup but also facilitates amortized O(1)-time insertion and removal at both ends. Strikingly, this can be achieved while maintaining space efficiency, utilizing O(n) space to store n items. The synthesis of these characteristics results in a robust and balanced data structure capable of meeting the diverse demands of sequence management.

(because We cant have space back)

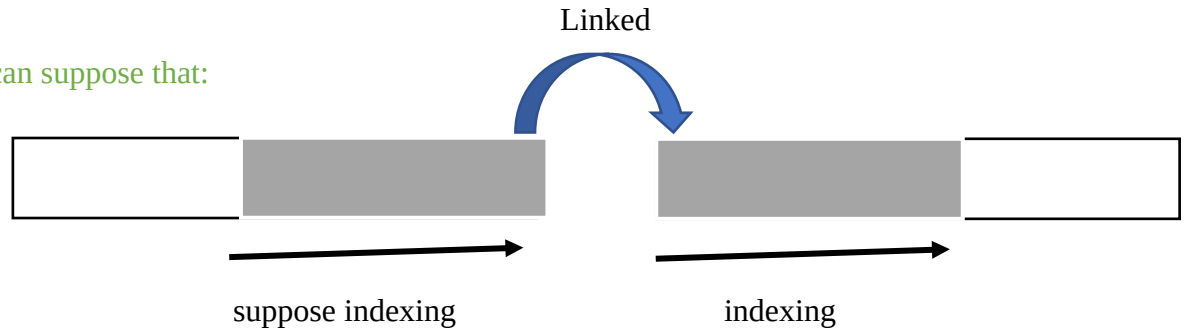
We can use Dynamic Array for one side and that’s Done!!!!

But for another Side we cant reserved space back and so we must use another way!!!!

We can make another Dynamic Array like before **BUT** must indexing it **REVERSE** and connected together with **Linked List** and now, We Make it 😊



We can suppose that:



DONE!