- If you did this with a stack instead, do you think it would still be possible to have random recommendations without losing the data?

A stack is similar to a stack of pancakes so it does not allow random access to its elements. If you want to get the last pancake or element you must remove all elements or pancakes that are above it. This makes it impossible to have random movie recommendations without losing the data or movies that are above the “randomly chosen element.”

Stacks allow peeking which returns the element at the top of the stack but doesn’t remove it. This will not be a random recommendation as it always returns the first element.

- If you could have used any of the data structures we've seen in the course, would you still have stuck with an ArraySequence?

Yes, I would still choose ArraySequence due to the following reasons:

Linked Lists (single or double): it is possible to implement a Movie Ranking System with a linked list. Linked lists can be sorted using merge sort (O(n\*logn)) which is more efficient than using an ArraySequence that implements bubble sort (O(n^2)). The main difference between using linked lists and ArraySequence will be how to access random movie recommendations. The ArraySequence has a field called current that stores the index of the current movie. If you want to get the current movie/random movie recommendation what you need to do is just return data[current](O(1)). In linked lists, if you want to access the current element you have to traverse the list(O(n)) from the head to find the element you are looking for. Getting a random recommendation with ArraySequence is more efficient than getting random recommendations with linked lists.

Bags and Sets: Bags and Sets are both similar to Sequences but it has an essential difference both of these are unordered. When doing a movie ranking system it is crucial to have a way to maintain order. In this case, it is better to use a container class that has a similar format but has the ability to keep all elements in it in a specific order. In addition, both bags and sets use a variable called items to track the number of elements, add elements, and track the index of elements. Sequences use the current element as it has no direct access operations for locating an element with an index. Meaning that sets and bags can’t provide random recommendations using a current element.

Sequence: Allows for easy sorting, adding, and removing of movies and it provides random movie recommendations.

Stacks: Stacks do not allow random access to their elements. This means that when I want to get/peek at the last element I need to remove all the elements that are above it. This makes it impossible to have random movie recommendations without losing the movies that are above the “randomly chosen movie.”

Queues: a queue is similar to a waiting line, meaning that you enter from the back and go out from the front or you add to the back of the queue and remove from the front of the queue. This is FIFO. This means that a queue is not supposed to be sorted since the order is similar to a waiting line. So if I used the queue the movies would not be sorted according to their ratings.

-Whichever data structure you choose, which aspect of it do you think makes it a good fit?

I choose to use ArraySequences because of the following reasons:

A sequence is a container class that has an order for the elements in it. According to my own criteria, the movie ranking system ordered had to order movies from highest to lowest rating. This feature made sorting really easy to implement.

Sequences can’t locate an element with an index, positions of movies are specified relative to the current movie. This feature makes it easy to add and remove elements and it allows the user to get random recommendations when needed.