COMP2402 Study Session

Agenda

Implementation, Runtime, and Space Efficiency

Groups

ArrayStack
ArrayList

- What are they?
- How are they used?
- How space efficient is it?

ArrayList

- Standard Indexed Collection that can store any generic data type
- Pretty helpful if you need to store elements with a respective index but never the **Best**

Stacks (ADT)

- Can be implemented with **ONE** collection that can store indexed elements
- As long as it can offer getting / setting at any element and adding /removing elements L.I.F.O in O(1).

Runtime of Core Methods

What are the runtime of the following methods?

- add(i,x)/remove(i,x)
- get(i)/set(i,x)

Resize and Amortized Runtime

What is Amortized Runtime, and how is it related to Resize() in ArrayList along with some other Data Structures? Think of analogies with bill payments or spreading over time.

Answers

- add(i,x)/remove(i,x) = O(1 + n-i)
- get(i)/set(i,x) = O(1)

Amortized Run, put simply, is making up for an expensive operation, by doing it only once every couple of operations.

Queue/Deques

- What are they?
- How are they used?
- How space efficient is it?

Circular Arrays

Queues are pretty simple in terms of first in, first out; however, the indexing can get confusing. How do we index with Queues?

Runtime of Core Methods

What are the runtime of the following methods for ArrayQueue?

- add(x)/remove(i)
- get(i)/set(i,x)

Answers

- add(x)/remove() = O(1)
- get(i)/set(i,x) = O(1)

(j+n-1)%a.length

Deque and DuelArrayDeque

- What are they?
- How are they used?
- How space efficient is it?

Dual Array Deques

- Implementation of the Deque Interface with 2 arrays instead of one circular array.
- Front array is **Reversed.**

Runtime of Core Methods

What is the runtime of the following methods for ArrayQueue?

- add(i,x)/remove(i,x)
- get(i)/set(i,x)

Balancing in DuelArrayDeques

Starting with an empty DualArrayDeque what is the total runtime spent calling the balance() method in a sequence of m numbers of add() and remove() operation in a DualArrayDeque?

Answers

- add(x)/remove() = O(1 + min(i,n-i))
- get(i)/set(i,x) = O(1)
- If we make m sequence of add and remove operation for every call to the balance() during the operation, we call at least n/2 add() or remove() operation between two consecutive balance() operation. So it’s in amortized runtime which results in o(m) total time spent.

Rootish ArrayStack

- What are they?
- How are they used?
- How fast are the get, add, and remove methods?
- How space efficient is it?

Indexing Certain Elements

RootishArrayStacks have a method to get the index of an element from each block. What would be the index in block of an element that is in index 15 of a RootishArrayStack?

Backing Arrays

ArrayStack/ArrayList

Queue

Deque

DuelArrayDeque

Singly LinkedLists

DoublyLinkedLists

RootishArrayStack

SinglyLinkedList and DoublyLinkedList

- What are they?
- How are they used?
- How fast are the get, add, and remove methods?
- How space efficient is it?

Runtime of Core Methods

What are the runtime of the following methods?

- add(i)
- remove(i)
- get(i)

Queues as SinglyLinkedLists

If we were to implement a Queue with SinglyLinkedList, what would we use the head or tail for? Are there any useless bits?

Stacks as SinglyLinkedLists

If we were to implement a Stack with a SinglyLinkedList, what would we use the head or tail for? Are there any useless bits?

Answers

- add(x)/remove() = O(1 + min(i,n-i))
- get(i)/set(i,x) = O(1)

As a Queue, push to tail and pop from head

As a Stack, push to tail and pop from tail

SkipList

- What are they?
- How are they used?
- How space efficient is it?

SkipList

Sorted set that uses “Levels” of Linked Lists to Store All elements. Each level is EXPECTED to have half as much elements as the level below it. An ideal Skip list has every odd index of a level also stored inside the level above it. This is also why a coin works (P(1 heads) = ½, P(2 heads) ¼ …)

Runtime of Core Methods

What are the runtime of the following methods?

- add(i)
- remove(i)
- get(i)

Answers

- add(x)/remove()/get(x) = O(Log n)

Implementation Details

Roughly know how each function works. You might get a question that asks which method is correct and which is slightly off

All Other ADT Applications

(Unsorted) Sets:

- Stores unordered, distinct elements
- O(1) add, get, set, remove

(Sorted) Sets:

- Stores Distinct Elements in order

All Other ADT Applications

Map

- Stores a key, value pair
- Can get, add, remove, and set values through keys in O(1). **Not True for the inverse.**
- Can be helpful for representing Duplicate elements while still having access to the set methods and runtimes. Think A1 P7.