Algorithms & Data Structures I Week 5 Lecture Note

Notebook:	Algorithms & Data Structures I		
Created:	2020-10-21 4:14 PM	**Updated:**	2020-10-28 7:48 PM
Author:	SUKHJIT MANN		

Cornell Notes	**Topic:** Vectors, stacks and queues part 1	Course: BSc Computer Science Class: CM1035 Algorithms & Data Structures I [Lecture] Date: October 28, 2020

Essential Question:

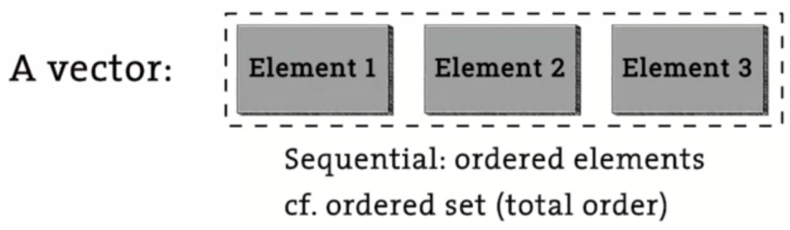
What are Vectors, stacks, and queues?

Questions/Cues:

- What is a vector?
- What is the length of a vector?
- What is the address of a vector?
- What operations can we perform on a vector?
- What is a Queue?
- What operations can we perform on a queue?

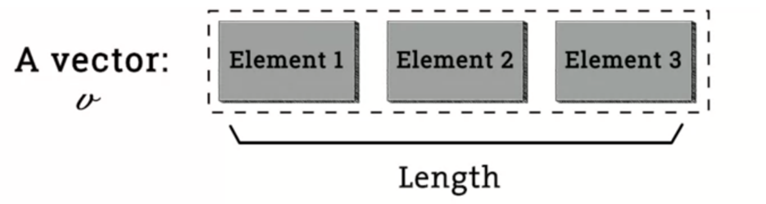
Notes

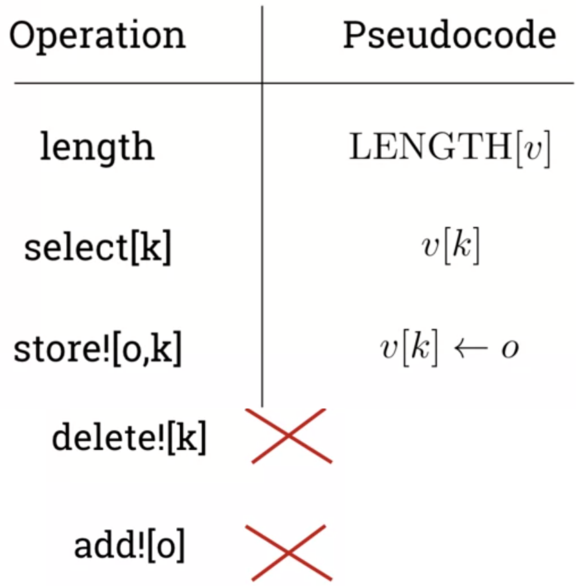
- Vector = Is a useful abstraction of memory and a simple building block for data storage. A vector is a finite fixed size sequential data collection. That is, it's a collection of elements of data where each piece of data comes after another piece of data and they can be structured in a line.



  - Because a vector is finite, it has a non-negative integer number of elements, it could be empty so the number of elements could be zero. A vector is fixed size, so the number of elements it has are fixed and cannot be altered; the number of elements is immutable.
  - Since the data has a fixed order in a line due to its sequential nature, the number of elements is called the length of the vector
    - The address of the element itself conveys useful information, it's often refer to as the index of the vector
- Operation (Vector) = is a function on the vector where if given the vector, an output can be produced, such as a number or the vector can be altered in some way
  - Length operation just returns the length of the vector, which is the number of elements

- select[k] operation which just returns the Kth element from the vector, which is the value stored at that element. So if we want a particular element of the vector, we apply the select operation to obtain it
- The store![o,k] operation which actually alters the vector, in particular it sets the Kth element of the vector to have the value O, this operation then actually alters the vector

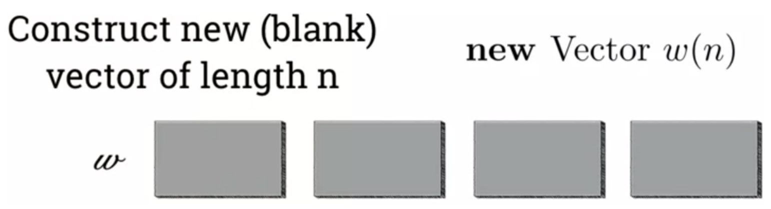


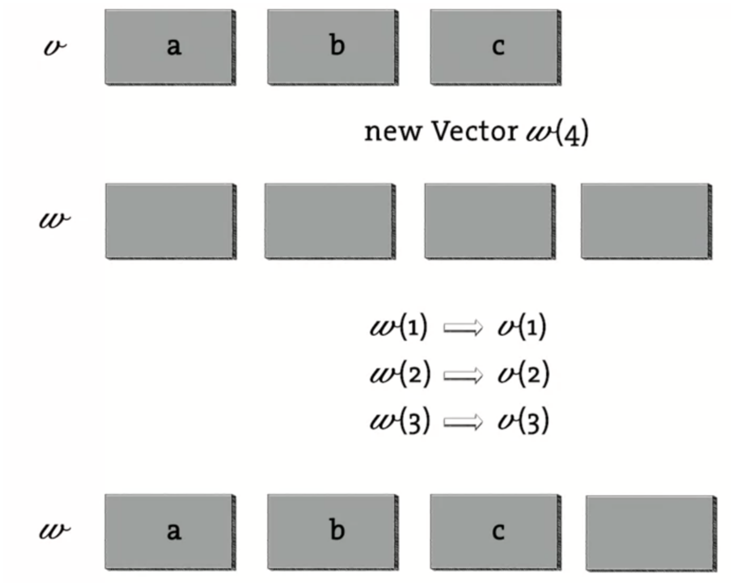


Operation	Pseudocode
length	$\text{LENGTH}[v]$
select[k]	$v[k]$
store![o,k]	$v[k] \leftarrow o$
delete![k]	✗
add![o]	✗

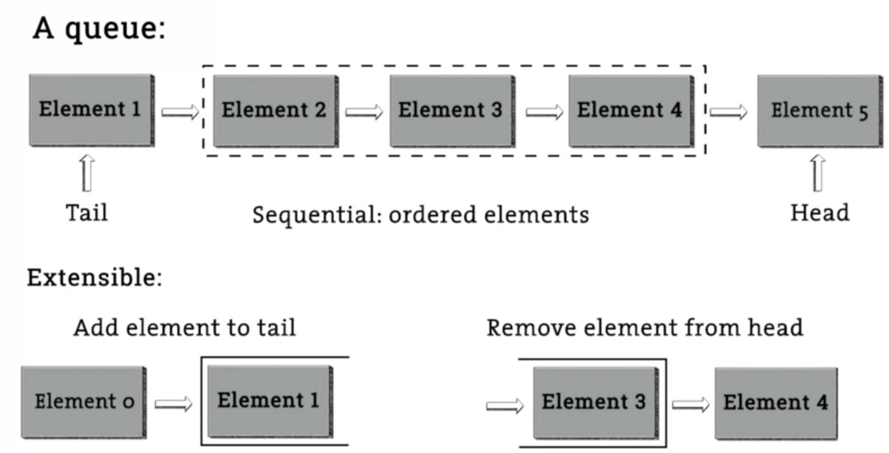
- Because the length of a vector is fixed, we can't delete any element from the vector as this would alter the length
- Also we can't just add a new element or just arbitrarily change the length of the vector
- One thing can be done is just to create a new vector of a fixed length. For example if we have a vector of length 3 and we actually want a vector of length 4 to store more data, we can just create a new vector of length 4 and we assign the first three elements of this new vector, the values of the original vector

***** Elements can store references to data!**



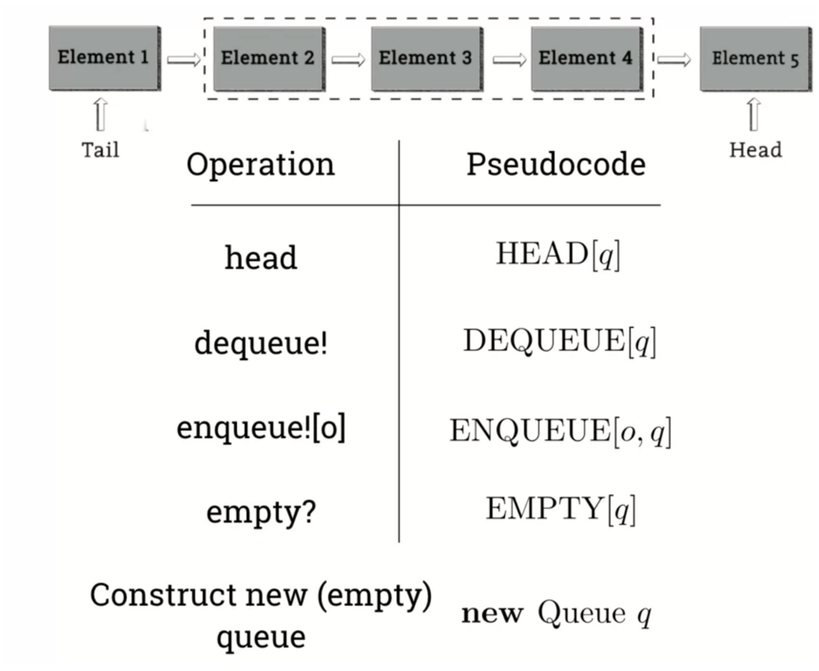


- Queue = The fundamental concept is that there is a resource, but it cannot be immediately accessed, so there needs to be a wait. Generally, the longer you have been waiting, the sooner the resource will be made available to you, often called first come, first served. In computing, a similar terminology is first in, first out or FIFO, but it's talking about the head of a queue. The first item in the queue will be the first to leave.
    - A queue is sequential so that it can be formed into a line, this makes sense as a queue is used interchangeably with the word line. But now we only have access to access to very specific elements, the head and the tail of a queue. Firstly, a queue is not fixed, it's extensible. We can add elements to a queue, and so the length of a queue can be altered.
    - The way we add elements to queues is very particular, we cannot add elements arbitrarily, only in a specific location, which is the tail of the queue
    - If we wish to remove an element from the queue, this is removed from the head of the queue
        - A queue resembles a vector, but now with an extensible length and with two privileged elements called the head and the tail, where elements can be removed and add respectively



    - Since we only have access to the tail and the head of a queue, we aren't able to read arbitrary elements of the queue.
        - Since access is restricted to only the head and the tail of the queue, all the other elements are just waiting to be addressed.
    - In order to access an arbitrary element of a queue, we first need to remove all of the elements in front of it from the queue so that this element is now at the head of the queue, and then we can address it
- Operation (Queue) = The operations on queues are as follows:
    - The head operation which returns the element at the head of the queue, so you get back the value which is stored at the head of the queue

- The dequeue! operation, which returns the element of the head of the queue, so you a value out, and then removes that element from the queue
- The enqueue![o] operation, which adds an element to the tail of the queue with the contents of that element having the value o
- The empty? operation, which just asks if the queue has no elements. It returns true if it's empty and false otherwise



Operation	Pseudocode
head	$\text{HEAD}[q]$
dequeue!	$\text{DEQUEUE}[q]$
enqueue![o]	$\text{ENQUEUE}[o, q]$
empty?	$\text{EMPTY}[q]$
Construct new (empty) queue	$\textbf{new} \text{ Queue } q$

Summary

In this week, we learned about what a vector is, what the length of a vector is, what operations we can perform on a vector, what is a queue and what operations we can perform on a queue.