Syllabus

Topics

Abstract data structures such as vectors, queues and stacks

Implementation of abstract data structures with arrays and linked lists

Searching algorithms: Linear Search and Binary Search

Sorting algorithms: Bubble Sort, Insertion Sort, Quicksort and Mergesort

Recursion as a technique for algorithm design

Worst-case time complexity: the RAM model of computation, Big O notation and computational complexity

Upon successful completion of this module, you will be able to:

Utilise pseudocode and flowcharts to describe algorithmic processes

Compare and outline the functionality of both abstract and concrete data structures

Describe and implement various sorting and searching algorithms

Explain the concept of recursion and implement a recursive algorithm

Analyse the worst-case performance of algorithms in terms of a model of computation

Abstract Data Types

BOOLEAN

True or False, logical operations

INTEGER

Integer values, arithmetic

VECTOR

A fixed length sequential data structure with arbitrary access

Operation	Pseudocode
length	LENGTH(K)
select(k)	v[k)]

Operation	Pseudocode
store!(o, k)	v[k] <- o
construct new blank vector of length n	New VECTOR v(n)

QUEUE

First in, first out. Variable length, restricted access, data structure.
Access only to head and tail

Tail -> Element 3 -> Element 2 -> Element 1 <- Head

Operation	Pseudocode	
head	HEAD(q)	Returns the element at the head of the queue
dequeue!	dequeue(q)	Removes and returns the element at the head of the queue
enqueue!(o)	enqueue(o, q)	Adds element o to the tailor the queue
empty?	EMPTY(q)	Returns True or False; does not ‘empty’ the queue.
Construct new (empty) queue	New QUEUE q	

STACK

Last in, first out. Variable length. Only a single element is accessible - the top.

Top -> Element 3 <-> Element 2 <-> Element 1

Operation	Pseudocode	
push!(o)	PUSH(o, s)	Places o on the top of the stack
top	top(s)	Returns the element at the top the stack
pop!(o)	pop(s)	Removes and returns the element at the top of the stack
empty?	EMPTY(s)	Returns True or False; does not empty the stack
Construct new (empty) stack	New STACK s	

DYNAMIC ARRAY

A variable length sequential data structure with arbitrary access

Operation	Pseudocode
length	LENGTH(d)

Operation	Pseudocode
select(k)	d[k)]
store!(o, k)	d[k] <- o
removeAt!(K)	d[k] <- 0, k \le LENGTH(d), shifts all elements to the left
insertAt!(o,k)	d[k] <- o, shift to the right
construct new blank vector of length n	New DYNAMICARRAY d(n)

(Concrete) Data Structures

Arrays and linked lists are the main (in this module) data structure for implementing ADTs.

Search

Linear Search

Search a vector for an item:

```
function LinearSearch(v, item)
        for 1 <= i <= LENGTH(v) do
                if v[i] = item then
                        return i
                end if
        end for
        return -1
end function
```