Data Structures and Data Manipulation

What the Specification Says:
Explain how static data structures may be used to implement dynamic data structures;
Describe algorithms for the insertion, retrieval and deletion of data items stored in stack, queue and tree structures;
Explain the difference between binary searching and serial searching, highlighting the disadvantages and disadvantages of each;
Explain how to merge data files;
Explain the differences between the insertion and quick sort methods, highlighting the characteristics, advantages and disadvantages of each.

Static and Dynamic Data Structures

- Static data structures are those which do not change in size while the program is running. Most arrays are static, once you declare them, they cannot change in size.
- Dynamic data structures can increase and decrease in size while the program is running.

Advantages of Static Data Structures

- Compiler can allocate space during compilation
- Easy to program
- Easy to check for overflow
- An array allows random access

Disadvantages of Static Data Structures

- Programmer has to estimate maximum amount of space needed
- Can waste a lot of space

Advantages of Dynamic Data Structures

- Only uses the space that is needed at any time
- Makes efficient use of the memory
- Storage no longer required can be returned to the system for other uses

Disadvantages of Dynamic Data Structures

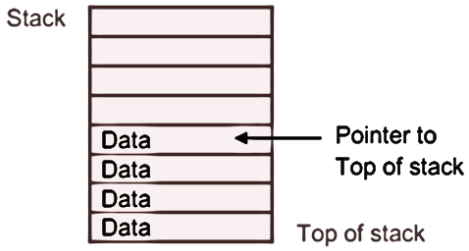
- Difficult to program
- Can be slow to implement searches
- A linked list only allows serial access

Static Structures holding Dynamic Structures

A static data structure (like an array) can hold a dynamic structure. The static structure must be big enough.

Stacks

A stack is a last in first out (LIFO or FILO) data structure. The head pointer will point to the most recent item of data which will be at the top. There are only two operations that can be applied, inserting and deleting/reading.



Inserting Data into a Stack

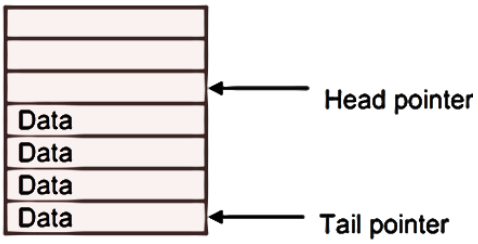
- First check that the stack is not full. If it is stop, and return an error.
- Next, increment the stack pointer, so it will now be pointing to the next empty data location.
- Finally insert the data into the location pointed to by the stack pointer.

Deleting and Reading from a Stack

- Check to see if the stack is empty. If it is stop and return an error.
- Copy the data item in the cell pointed to be the stack pointer.
- Decrement the stack pointer and stop.

Queues

A queue is a last in last out (LILO or FIFO) data structures. It has a head pointer, like that of a stack which points to the next empty data location, and a tail pointer which points to the last data item. Again, there are only two operations that can be done to a queue.



Inserting Data into a Queue

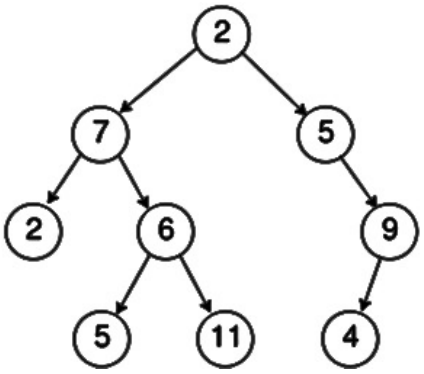
- Check that the queue is not full, if it is report an error and stop.
- Insert the new data item into the cell pointed to by the head pointer.
- Increment the head pointer and stop.

Deleting and Reading from a Queue

- Check that the queue is not empty, if it is report an error and stop.
- Copy the data item in the cell pointed to by the tail pointer.
- Increment the tail pointer and stop.

Binary Tree's

- A binary tree is a data structure, where each item of data points to another two items, and a rule is needed to determine the route taken from any data item.
- The data items are held in nodes.
- The possible routes are called paths.
- Each node has two possible paths.
- The nodes are arranged in layers.
- The first node is called the root, or root node.



Inserting Data into a Binary Tree

- Look at each node starting from the root
- If the new value is less than the value of the of the node, move left, other wise move right
- Repeat this for each node arrived until there is no node
- Then create a new node and insert the data.
- This can be written as:

```
1.  If tree is empty enter data item at root and stop.
2.  Current node = root.
3.  Repeat steps 4 and 5 until current node is null.
4.  If new data item is less than value at current node go left
else go right.
5.  Current node = node reached (null if no node).
6.  Create new node and enter data.
```

Deleting Data from a Tree

Deleting data from a tree is quite complicated, because if it has sub-nodes, these will also be deleted. There are two options.

- The structure could be left the same, but the value of that node set to deleted.
- The tree could be traversed, the value removed, then put back into a binary tree.

Serial Search

- Expects data to be in consecutive locations (such as, an array). Doesn't expect the data to be in any particular order.
- To find the position of a value, look at each value in turn, and compare it with the value that you are looking for.
- When the value is found, it's position must be noted. If it gets to the end and has not found the value, it is not in the array.
- Can be slow, especially for a large amount of data.
- The algorithm for this is:

```
1.  If n < 1 then report error, array is empty.
2.  For i = 1 to n do
a.  If DataArray[i] = X then return i and stop.
3.  Report error, X is not in the array and stop.
```

Binary Search

- Where the list is arranged in a particular order.
- The list is split in two, and compared to be either higher or lower than the value being searched for.
- The list is continually split further halving each time until the value is found.
- The algorithm for this is:

```
1.  While the list is not empty do
a.  Find the mid-point cell in the current list.
b.  If the value in this cell is the required value, return the cell
position and stop.
c.  If the value to be found is less than the value in the mid-point
cell then
  Make the search list the first half of the current search list
  Else make the search list the second half of the current
  search list.
2.  Report error, item not in the list.
```

Sorting

Sorting is placing values in order, such as numeric or alphabetic. There are two main types we need to know about. Insertion sort and quick sort.

Insertion Sort

Where the data of the files is copied into a new file, but copied into the correct location. The result is that the new file is in the correct order, although it's very time consuming.

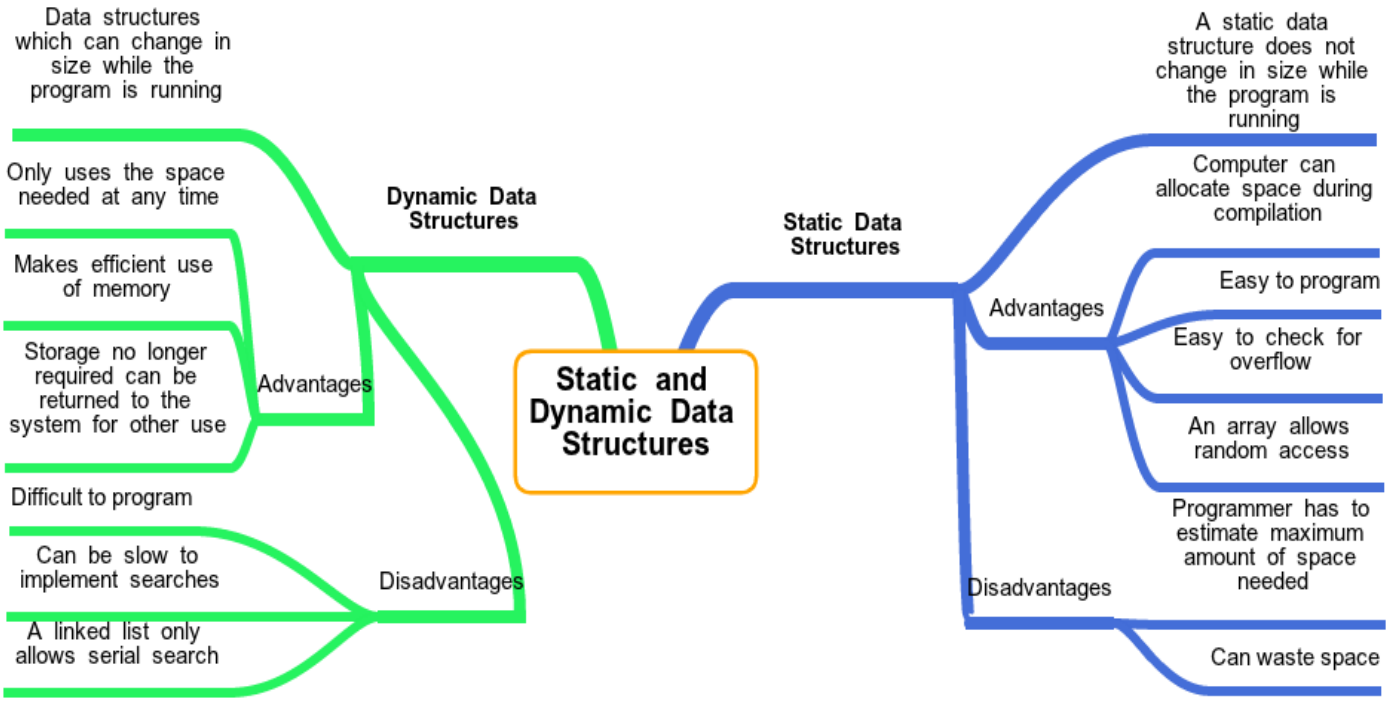
Quick Sort

- First the data is placed in a row with an arrow under the first and last values, pointing at each other, one is fixed where as one is movable.
- If the two values are in the correct order then move the movable arrow towards the fixed arrow, else swap the items and the arrows.
- Continue to repeat this until the arrows collide.
- Continue to repeat this process until the files are of a length of one.

Key Words

- **Data Structure** – Method of storing a group of related data.
- **List** – A simple one dimensional array
- **Pointers** – The numbers after the data, they point to the next data item.
- **Linked List** – A dynamic data structure similar to an array.
- **Queue** – A fist in first out data structure, containing a head and tail pointer.
- **Tree** – A data structure where each item of data points to two others.
- **Nodes** – These hold the data items.
- **Paths** – These are the routes between the nodes in a tree.
- **Layers** – Binary trees are arranged into layers, by the different levels.
- **Root** – The first node in a tree.
- **Insertion Sort** – A method of sorting data where all the items are copied to a new file, and put into the correct order.
- **Quick Sort** – A faster method of sorting data, involving two pointers which move towards each other, and swap values if data pointed at are in the wrong order.

Static and Dynamic Data Structures Summary:



Past Exam Questions and Answers

Showing the steps of serial search
e.g. Find York
Aberdeen, Belfast, Cardiff, Oxford, York

1. start at 'Aberdeen'…
2. look at each word in turn/then 'Belfast', 'Cardiff' etc…
3. …until 'York' is found

Showing the steps of binary search
e.g. Find York
Aberdeen,Belfast, Cardiff, Oxford, York

look at middle/'Cardiff'/'Glasgow'
'York' is in second half of list
repeated halving…
…until 'York' is found

What are the advantages of binary search over seial search?

(usually) faster because…
…alf of data is discarded at each step/fewer items are checked

How do you add a data item into a stack?

if stack is full…
…report error and stop
increment pointer
add data item at position 'pointer

How can quicksort be used to put a set of numbers in ascending order?
e.g. 30 9 46 14 22

	30 →	9	46	14	22 ←
swap 30 & 22	22 →	9	46	14	30 ←
	22	9 →	46	14	30 ←
	22	9	46 →	14	30 ←
swap 46 & 30	22	9	30 →	14	46 ←
	22	9	30 →	14 ←	46
swap 30 & 14	22	9	14 →	30 ←	46
	22	9	14	30 → ←	46

**

highlight first number in the list (the 'search number')
pointer at each end of list
repeat:
compare numbers being pointed to…
…f in wrong order, swap
move pointer of non-search number
until pointers coincide so search number in correct position
split list into 2 sublists
quick sort each sublist
repeat until all sublists have a single number
put sublists back together

Name another method or sorting

insertion sort or bubble sort

What is a static data structure?

size is fixed when structure is created/size cannot change during processing

What would be an advantage of static data structures over dynamic structures?

amount of storage is known/easier to program

How would you merge two files?
e.g. File A: Anna, Cleo,Helen, Pretti
Fie B: Billy, Ian, Omar, Rob, Tom

(Anna, Billy, Cleo, Helen, Ian, Omar, Pritti, Rob, Tom)
You must: get correct order, use all names used once

State the algorithm for merging two sorted files

open existing files
create new file
check existing files are not empty
use pointers/counters to identify records for comparison
repeat
compare records indicated by pointers
copy earlier value record to new file
move correct pointer
until end of one file
copy remaining records from other file
close files
assume common key
assume if 2 records are the same…
…only 1 is written to new file

What is a dynamic data structure?

size changes as data is added & removed/**size** is not fixed

What would be a disadvantage to the programer of using dynamic data structures over static ones?

more complex program to write

State a data structure which must be static

array/fixed length record

What steps need to be taken to add a new item to a binary tree

start at root
repeat
compare new data with current data
if new data < current data, follow left pointer
else follow right pointer
until pointer is null
write new data
create (null) pointers for new data

How can insertion sort be used to arrange numbers in order?
e.g. 17 2 3 26 5

Original set	***17***	2	3	26	5
Insert 17	17	***2***	3	26	5
insert 2	2	17	***3***	26	5
insert 3	2	3	17	***26***	5
insert 26/no change	2	3	17	26	***5***
insert 5	2	3	5	17	26

**

list of sorted numbers is built up…
…ith one number at a time being inserted into correct position
plus 1 mark per correct row *[max 4 rows]* *

What features of quick sort are not used in insertion sort?

set of numbers broken into multiple sets
uses pivots

What steps are needed to be taken to pop a data item from a stack?

if stack is empty…
…eport error and stop
output data(stack_pointer)
decrement stack_pointer

What is the meaning of a dynamic data structure

size changes as data is added & removed/size is not fixed

What's the main disadvantage of a dynamic data structures over static one?

more complex program to write

What data structure must be static

array/fixed length record

How would you add a new item to an existing binary tree?

start at root
repeat
  compare new data with current data
  if new data < current data, follow left pointer
  else follow right pointer
until pointer is null
write new data
create (null) pointers for new data