122com Data structures and types

David Croft

Coventry University

david.croft@coventry.ac.uk

2017

Coventry University

122com Data structs & types

David Croft

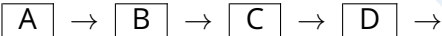Coventry University

Arrays C

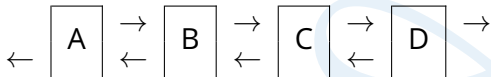A series of objects all of the same size and type.

```
array<char,5> arr = {'A', 'B', 'C', 'D', 'E'};
```

- Stored in contiguous blocks of memory.
- Python lists are functionally closest.
  - But are not arrays.
- Can't be resized.

Coventry
University

The challenger for array's crown.

- Series of nodes, each of which points to the next element.
    - And to the previous element if it's a doubly linked list.

Singly linked   A → B → C → D →

Doubly linked   ← A ⇄ B ⇄ C ⇄ D →

Coventry University

122com Data structs & types

David Croft

Arrays
Linked lists
  Array example
  LL example
Data structures
Abstract data types
Queues
Stacks
Sets
Other
Trees
Quiz
Recap

Linked lists II

A

Not in contiguous memory.

- Each node is separate.
- Scattered.
- C++ Dynamic memory (pointers!).
    - Discussed in depth later in module.
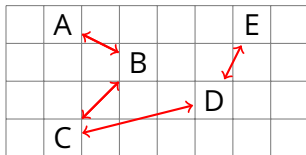
- Why would we use linked lists instead of arrays?
    - Can change size.
    - Can quickly insert and delete elements.

```
class Node:
    __prev = None
    __next = None
    value = None
```

```
class Node
{
private:
    Node *prev;
    Node *next;

public:
    int value;
};
```

122com Data structs & types

David Croft

Arrays

Linked lists
  Array example
  LL example

Data structures

Abstract data types

Queues

Stacks

Sets

Other

Trees

Quiz

Recap

Removing array elements

C

```
temp = ['A', 'B', 'C', 'D', 'E']
array<char,5> temp {'A', 'B', 'C', 'D', 'E'};
```

		A	B	C	D	E	

- Array in memory, multiple elements in a contiguous block.
- How do we remove elements from the middle?
  1. Remove element from the array.
  2. Move next element to occupy the empty space.
  3. Repeat.
- Is very slow with large arrays.

Removing linked list elements

I

- Linked list, separate elements scattered in memory.
- Each pointing to the next/prev element.
- How do we remove elements?
  1. Change pointers.
  2. Delete old element.

Coventry University

122com Data structs & types

David Croft

Arrays

Linked lists
Array example
LL example

Data structures

Abstract data types
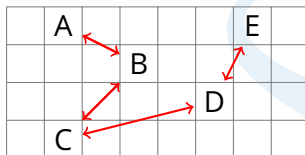
Queues

Stacks

Sets

Other

Trees

Quiz

Recap

Linked lists III

Advantages

- ■ Inserting and deleting elements is very fast.
  - ● $O(1)$.
- ■ No size limits, can keep adding new elements.
- ■ Doesn't waste memory.

Disadvantages

- ■ Not indexed.
  - ■ Can't ask for the 20th element etc.
  - ■ Have to step through the list (slow).
- ■ Needs more memory than an array to store the same number of elements.
  - ■ Have to keep track of where the next/prev nodes are.

Coventry University

Data structures

C

Arrays and linked lists are data structures.

- A specific way of storing data.
- Can see how the various elements of the structure are laid out in memory.
- Direct access to the underlying memory.

Coventry University

Abstract data types

I

As we move to storing more complex information in our software we well start to encounter Abstract Data Types (ADTs).
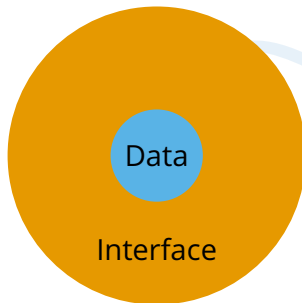
- Software engineering principal.
- Keep what a data type can do...
  ...and how it does it separate.

- Unlike data structure ADTs only concerned with the interface.
- Internals of ADTs can vary widely between implementations.

Coventry
University

122com Data structs & types

David Croft

Arrays
Linked lists
    Array example
    LL example
Data structures
Abstract data types
Queues
Stacks
Sets
Other
Trees
Quiz
Recap

Coventry University

ADT

Imagine an ADT like a car.

- It has a set of supported operations, go faster, go slower, turn left, turn right.
- Don't care how it achieves these.
- Don't care if, internally, it's using a combustion engine or an electric motor.
- Only care about the result.
- Keep people away from the internal workings/data.

Programmer

Data

Interface

122com Data structs & types

David Croft

Arrays

Linked lists
 Array example
 LL example

Data structures

Abstract data types

Queues

Stacks

Sets

Other

Trees

Quiz

Recap

Queues C

A First In First Out (FIFO) ADT.

- Ends of the queue called the front and back.
- New elements added to back of queue only.
  - Pushing - push(value)
- Old elements removed from front of queue only.
  - Popping - pop()
- No cutting in.
- Buffer to hold items for processing in the order in which they arrive.

- Which would be better for a queue? An array or a linked list?
  - Linked list.

Array as a queue.

front ⇒ | | | | | |
⇑

front ⇒ | A | | | | | push(A)
⇑

front ⇒ | A | B | | | | push(B)
⇑

front ⇒ | A | B | C | | | push(C)
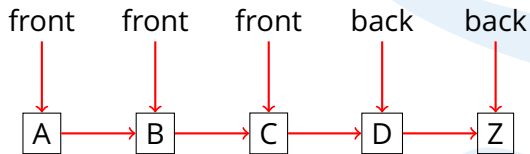⇑
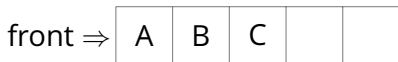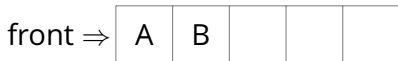
front ⇒ | B | C | | | | pop()
⇑

- Very similar to stacks.
  - Keep track of next free space.

  - Limited size.
- What happens when we pop()?
  - Have to shuffle every element forward one space.

  - Inefficient.

Coventry University

Linked list as a queue.

A

pop() , pop() , push(Z)

front   front   front   back   back

↓   ↓   ↓   ↓   ↓

[A] → [B] → [C] → [D] → [Z]

Coventry
University

Coventry
University

A First In Last Out (FILO) ADT.

- Ends of the stack are called the top and bottom.
- New elements add to top of stack only.
  - Pushing - push(value)
- Old elements removed from top of stack only.
  - Popping - pop()
- No cutting in.

- Which would be better for a stack? An array or a linked list?
  - Doesn't matter performance wise.
  - Linked list if n is unknown.

Array as a stack. I
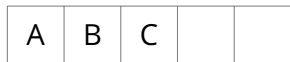
push(A)

push(B)

push(C)

pop()

- Keep track of position of the next free space in the array.
- Arrays have a fixed size.
  - Can't hold more values than we have space for.

Coventry University

Sets   C

- An unordered ADT.
    - Items ordered by the set.
    - You have no control over it.
- Sets contain unique elements.
    - Can't contain duplicates.

- Can add items to a set.
- Can remove items from a set.
- Can see if an item is in a set.
- Can't get the nth element.
    - It's unordered remember.

A				

add(A)

A	B			

add(B)

A	B			

add(A)

	B			

remove(A)

Coventry
University

122com Data structs & types

David Croft

Arrays

Linked lists
  Array example
  LL example

Data structures

Abstract data types

Queues

Stacks

Sets

Other

Trees

Quiz

Recap

...and the others

- Lots of other ADTs.
- Different names in different languages.

- Lists.
- Circular lists.
- Associative arrays.
  - Dictionaries/Maps.
- Double-ended queues.
- Trees.
- Graphs.

Coventry University

Variation on linked lists.

- Made of nodes and relationships.
- Root node at top.
- Each node can have $\geq$ 0 children.

- Binary search tree.
  - Very common type.
  - Ordered.
  - Max two children.
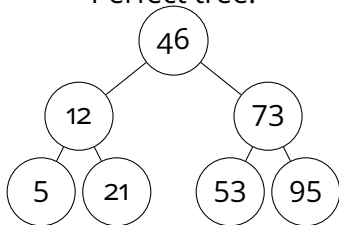 - Binary searching.
  - Very good for sets.
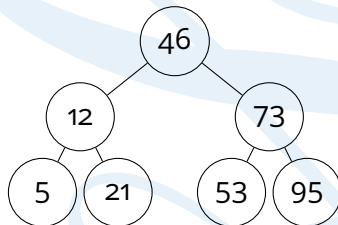
- Trees can be balanced or unbalanced.
- Not required for all trees.
- Going to be talking about BSTs from here on.

- Unbalanced because more than a one node difference between the two halves.
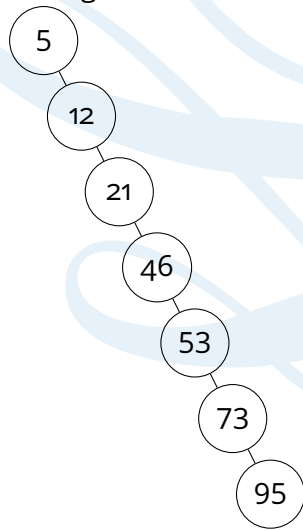 - For the whole tree...
  - ...and one of the subtrees.

Coventry
University
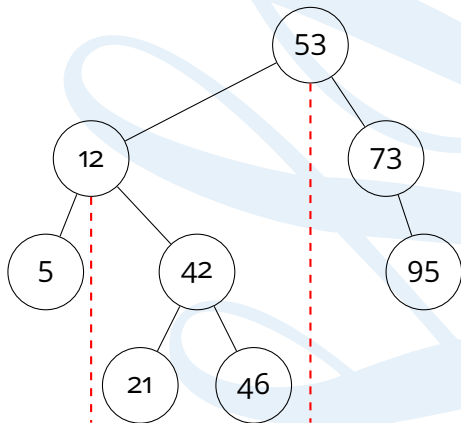
Important that you keep your BSTs balanced.

Degenerate tree.

Perfect tree.

Coventry University

Quiz

Stacks and queues are examples of ____

- Data structures.
- Linked lists.
- Arrays.
- Abstract Data Types.

Coventry
University

One advantage of linked lists over arrays is that _____

- ■ They use less memory.
- ■ They don't waste memory.
- ■ They can be used for queues.
- ■ They are faster to search though.

Coventry University

Stacks are a _____ type.

- FIFO.
- FOFI.
- FILO.
- FIDO.

Coventry University

The elements of a set can't _____

- Contain duplicates.
- Be sequences, ie. lists, strings.
- Be out of order.
- Be removed.

Coventry
University

ADTs separate the _____ of the type.

- ■ Input and output.
- ■ Attributes and methods.
- ■ Implementation and interface.
- ■ Code and software.

Coventry
University

122com Data structs & types

David Croft

Arrays

Linked lists
  Array example
  LL example

Data structures

Abstract data types

Queues

Stacks

Sets

Other

Trees

Quiz

Recap

Why do I care?

Everyone

- Need to understand the structures before we can pick the right one.
- Different data structures have very different characteristics.
- Huge effect on efficiency of your code.
- If you pick the right ADT it can save you a lot of code.
  - E.g. why write code to check for duplicates? Use a set and they can't exist.
  - E.g. why write code to find the most recent addition to a list, use a stack.

Coventry
University

Recap

- 🟢 Arrays.
  - 🟢 Advantages/disadvantages.
- 🟢 Linked lists .
  - 🟢 Advantages/disadvantages.
  - 🟡 How to insert/delete.
- 🟢 Difference between data structure and ADTs.
- 🟢 Stack.
  - 🟢 FILO.
  - 🟡 Using an array as one.
  - 🔴 Using a LL as one.

- 🟢 Queue.
  - 🟢 FIFO.
  - 🟡 Using an array as one.
  - 🔴 Using a LL as one.
- 🟢 Sets.
  - 🟢 No duplicates.
  - 🟡 Unordered.
- 🟡 Trees.
  - 🔴 Balanced/unbalanced.

Coventry
University

The End