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

122com Data structures and types

Coventry University

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

Overview

1 Arrays
2 Linked lists
- Array example
- LL example

3 Data structures
4 Abstract data types
5 Queues
6 Stacks
7 Sets
8 Other
9 Trees
10 Quiz
11 Recap

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

Arrays

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.

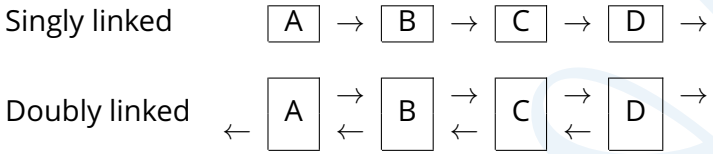
Arrays
Linked lists
Array example
LL example
Data structures
Abstract data types
Queues
Stacks
Sets
Other
Trees
Quiz
Recap

Linked lists

C

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.



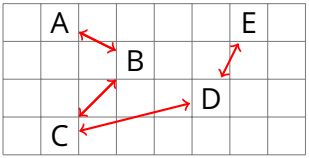
Arrays
Linked lists
Array example
LL example
Data structures
Abstract data types
Queues
Stacks
Sets
Other
Trees
Quiz
Recap

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;
};
```

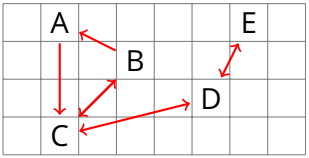
Arrays
Linked lists
Array example
LL example
Data structures
Abstract data types
Queues
Stacks
Sets
Other
Trees
Quiz
Recap

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;
};
```

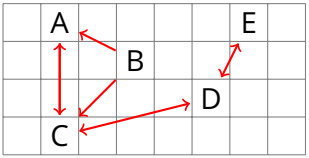
Arrays
Linked lists
Array example
LL example
Data structures
Abstract data types
Queues
Stacks
Sets
Other
Trees
Quiz
Recap

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
```

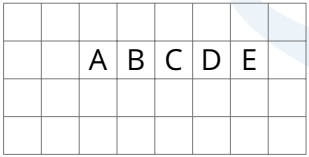
```
class Node
{
private:
    Node *prev;
    Node *next;

public:
    int value;
};
```

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'};
```



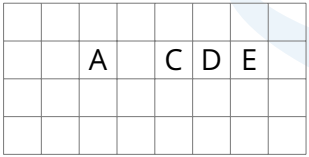
- Array in memory, multiple elements in a contiguous block.
- How do we remove elements from the middle?

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'};
```



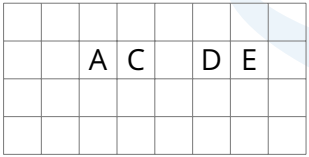
- Array in memory, multiple elements in a contiguous block.
- How do we remove elements from the middle?
  1. Remove element from the array.

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'};
```



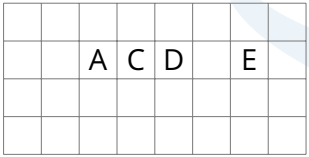
- Array in memory, multiple elements in a contiguous block.
- How do we remove elements from the middle?
  1. Remove element from the array.
  2. Move next element to occupy the empty space.

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'};
```



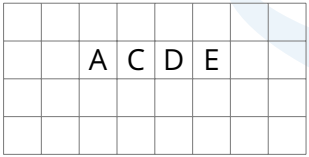
- Array in memory, multiple elements in a contiguous block.
- How do we remove elements from the middle?
  1. Remove element from the array.
  2. Move next element to occupy the empty space.
  3. Repeat.

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'};
```

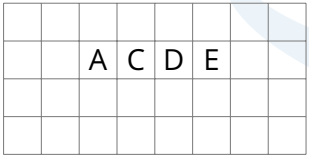


- Array in memory, multiple elements in a contiguous block.
- How do we remove elements from the middle?
  1. Remove element from the array.
  2. Move next element to occupy the empty space.
  3. Repeat.

Removing array elements

C

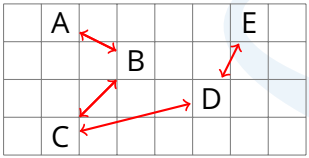
```
temp = ['A', 'B', 'C', 'D', 'E']
array<char,5> temp {'A', 'B', 'C', 'D', 'E'};
```



- 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.

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

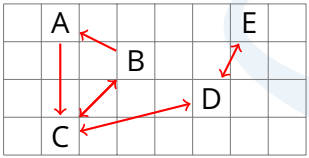
Removing linked list elements



- Linked list, separate elements scattered in memory.
- Each pointing to the next/prev element.
- How do we remove elements?

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

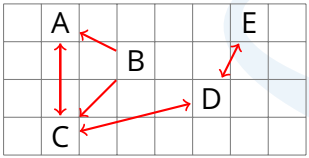
Removing linked list elements



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

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

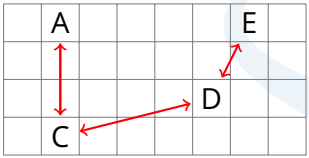
Removing linked list elements



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

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

Removing linked list elements



- 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.

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 20^{th} 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.

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

Data structures

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.

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

Abstract data types

As we move to storing more complex information in our software we well start to encounter Abstract Data Types (ADTs).

- Software engineering principal.

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

Abstract data types

As we move to storing more complex information in our software we well start to encounter ADTs.

- Software engineering principal.
- Keep what a data type can do...

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

Abstract data types

As we move to storing more complex information in our software we well start to encounter ADTs.

- Software engineering principal.
- Keep what a data type can do...

  ...and how it does it separate.

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

Abstract data types

As we move to storing more complex information in our software we well start to encounter 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.

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

Abstract data types

As we move to storing more complex information in our software we well start to encounter 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.

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

ADT

Imagine an ADT like a car.

- It has a set of supported operations, go faster, go slower, turn left, turn right.

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

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.

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

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.

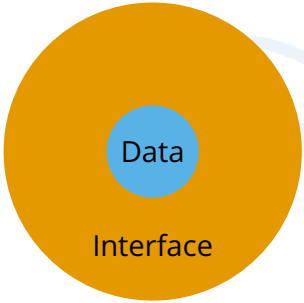
Arrays
Linked lists
Array example
LL example
Data structures
Abstract data types
Queues
Stacks
Sets
Other
Trees
Quiz
Recap

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



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

Queues

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.

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

Queues

C

A 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.

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

Queues

C

A 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?

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

Queues

C

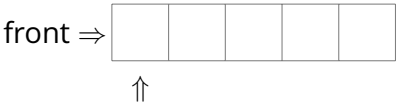
A FIFO ADT.

- 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.

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

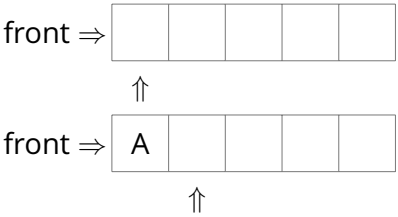
Array as a queue.



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

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

Array as a queue.

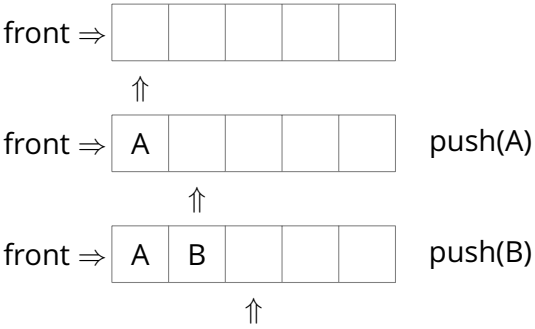


push(A)

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

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

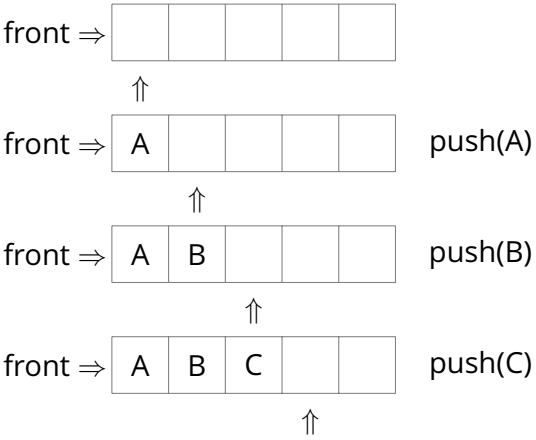
Array as a queue.



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

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

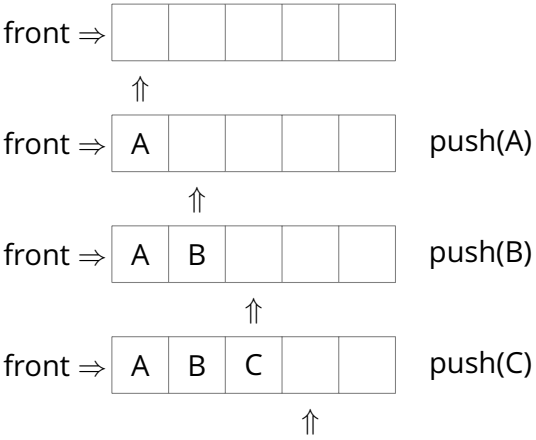
Array as a queue.



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

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

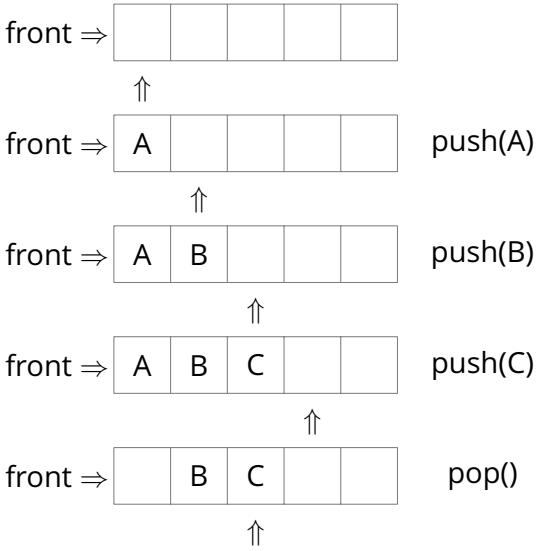
Array as a queue.



- 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.

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

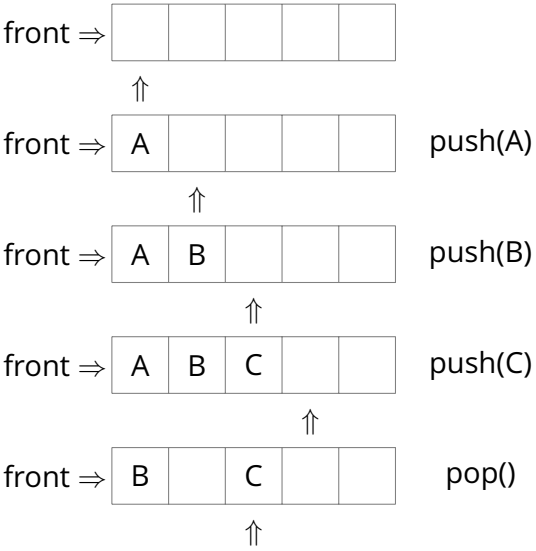
Array as a queue.



- 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.

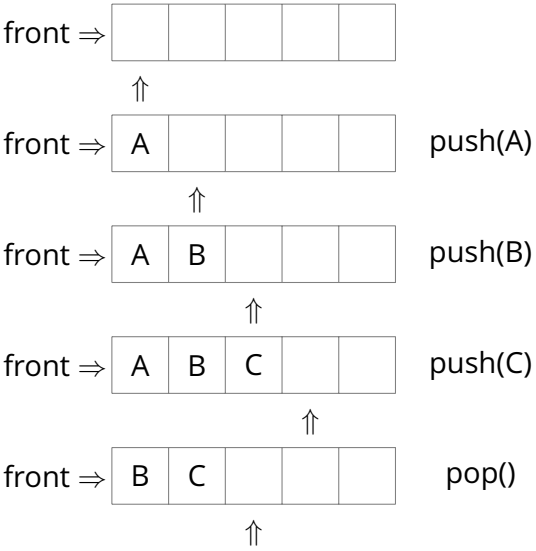
Arrays
Linked lists
Array example
LL example
Data structures
Abstract data types
Queues
Stacks
Sets
Other
Trees
Quiz
Recap

Array as a queue.



- 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.

Array as a queue.

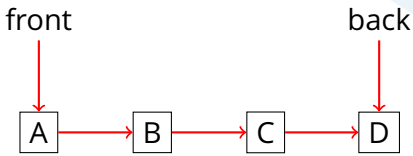


- Very similar to stacks.
  - Keep track of next free space.
  - Limited size.
- What happens when we pop()?
  - Have to shuffle every element forward one space.
  - Inefficient.

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

Linked list as a queue.

A

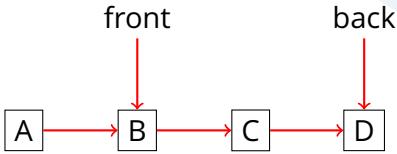


Arrays
Linked lists
Array example
LL example
Data structures
Abstract data types
Queues
Stacks
Sets
Other
Trees
Quiz
Recap

Linked list as a queue.

A

pop()

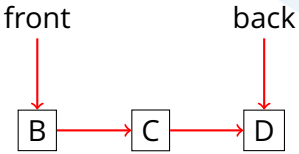


Arrays
Linked lists
Array example
LL example
Data structures
Abstract data types
Queues
Stacks
Sets
Other
Trees
Quiz
Recap

Linked list as a queue.

A

pop()

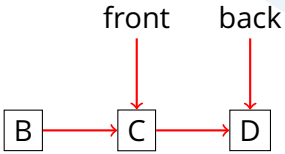


Arrays
Linked lists
Array example
LL example
Data structures
Abstract data types
Queues
Stacks
Sets
Other
Trees
Quiz
Recap

Linked list as a queue.

A

pop() , pop()

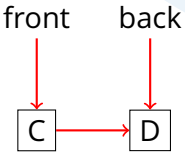


Arrays
Linked lists
Array example
LL example
Data structures
Abstract data types
Queues
Stacks
Sets
Other
Trees
Quiz
Recap

Linked list as a queue.

A

pop() , pop()

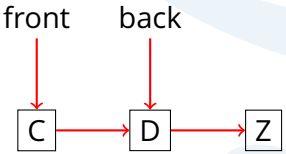


Arrays
Linked lists
Array example
LL example
Data structures
Abstract data types
Queues
Stacks
Sets
Other
Trees
Quiz
Recap

Linked list as a queue.

A

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

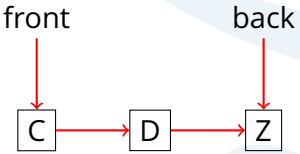


Arrays
Linked lists
Array example
LL example
Data structures
Abstract data types
Queues
Stacks
Sets
Other
Trees
Quiz
Recap

Linked list as a queue.

A

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



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

Stacks

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.

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

Stacks

C

A 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?

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

Stacks

C

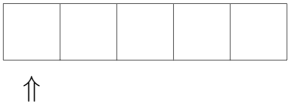
A FILO ADT.

- 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.

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

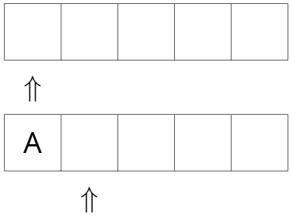
Array as a stack.



- 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.

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

Array as a stack.

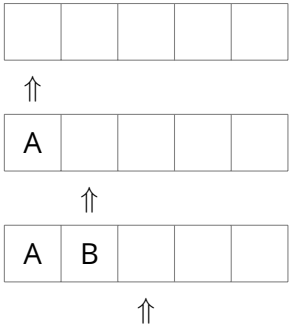


push(A)

- 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.

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

Array as a stack.



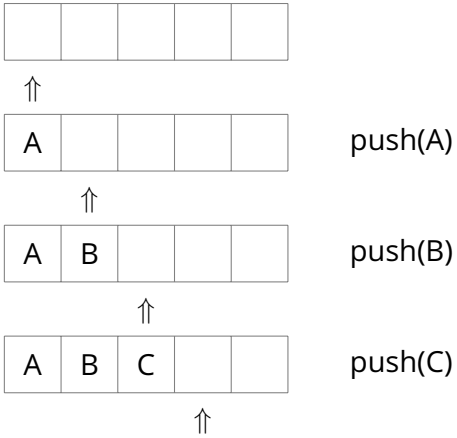
push(A)

push(B)

- 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.

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

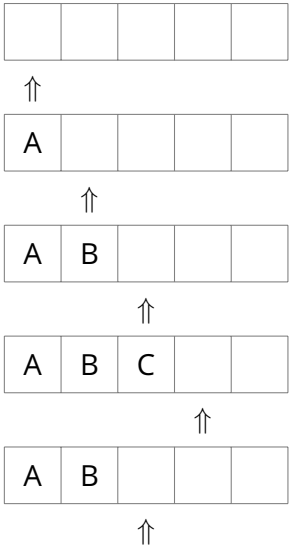
Array as a stack.



- 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.

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

Array as a stack.



- 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.

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

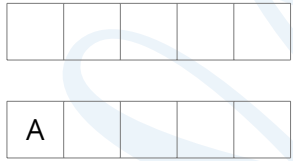
Sets

- 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 n^{th} element.
  - It's unordered remember.

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

Sets



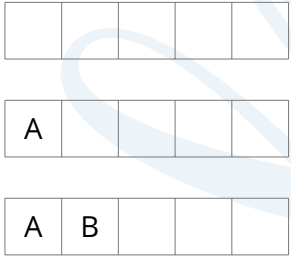
- An unordered ADT.
  - Items ordered by the set.
  - You have no control over it.
- Sets contain unique elements.
  - Can't contain duplicates.

- 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 n^{th} element.
  - It's unordered remember.

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

Sets

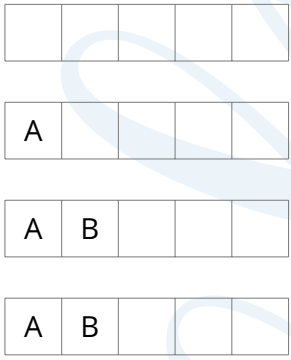
- An unordered ADT.
  - Items ordered by the set.
  - You have no control over it.
- Sets contain unique elements.
  - Can't contain duplicates.



- 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 n^{th} element.
  - It's unordered remember.

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

Sets



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

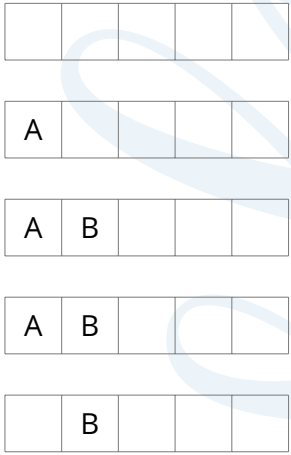
- Can add items to a set.
- Can remove items from a set.
- Can see if an item is in a set.
- Can't get the n^{th} element.
  - It's unordered remember.

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

Sets

- 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 n^{th} element.
  - It's unordered remember.



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.

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

Break

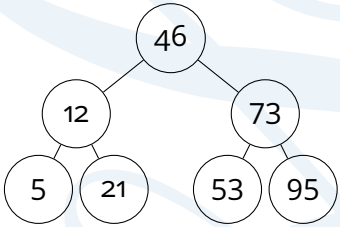
Trees

A

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.



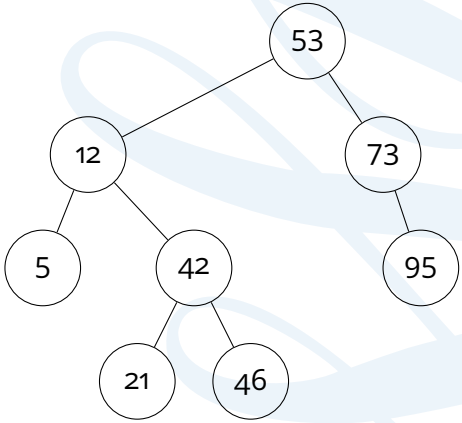
Arrays
Linked lists
Array example
LL example
Data structures
Abstract data types
Queues
Stacks
Sets
Other
Trees
Quiz
Recap

Balance

A

- 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.

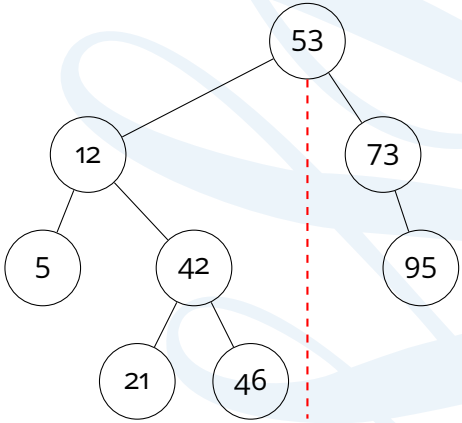


Balance

A

- 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...

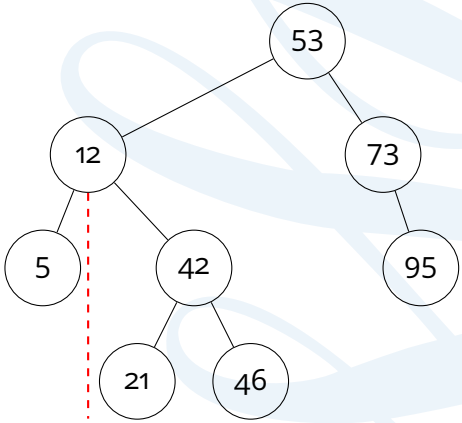


Balance

A

- 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.

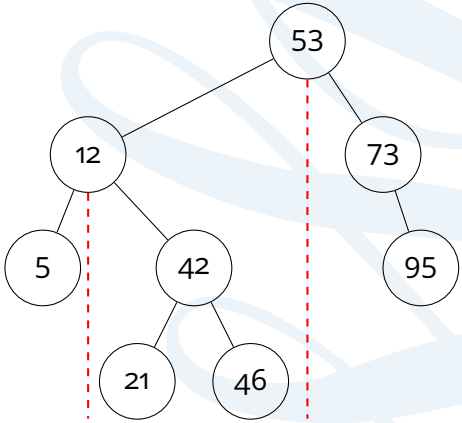


Balance

A

- 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.

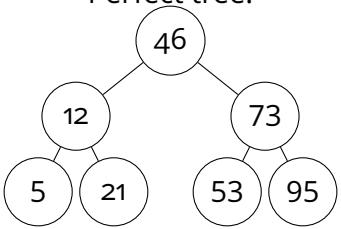


Balance

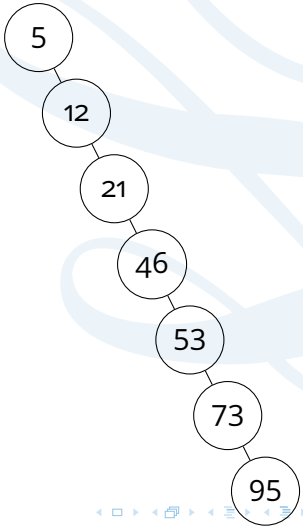
A

Important that you keep your BSTs balanced.

Perfect tree.



Degenerate tree.



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

Quiz

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

Q1

Stacks and queues are examples of ____

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

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

Q1

Stacks and queues are examples of ____

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

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

Q2

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.

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

Q2

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.

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

Q3

Stacks are a ____ type.

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

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

Q3

Stacks are a ____ type.

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

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

Q4

The elements of a set can't ____

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

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

Q4

The elements of a set can't ____

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

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

Q5

ADTs separate the ____ of the type.

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

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

Q5

ADTs separate the ____ of the type.

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

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.

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

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.

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

Expectations

C

- Complete the yellow Codio exercises for this week.
- Attempt the green Codio exercises for next week.

- If you have spare time attempt the red Codio exercises.

- If you are having issues come to the PSC.

  `https://gitlab.com/coventry-university/programming-support-lab/wikis/home`

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

The End