What are Data Structures?

Written by Rachel J. Morris, last updated 1/6/17

Topics:

- What is this class?
- What are data structures?

Introduction

What are data structures, and why is there a whole class about them?

Introduction

What are data structures, and why is there a whole class about them?

Well… they're structures… that store data.

Introduction

What are data structures, and why is there a whole class about them?

Well… they're structures… that store data.

Is that it?

Introduction

What are data structures, and why is there a whole class about them?

Well… they're structures… that store data.

Is that it?

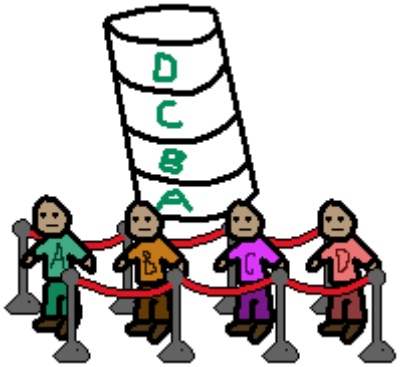
That's basically it, but building objects to store data actually gets pretty in-depth.

Introduction

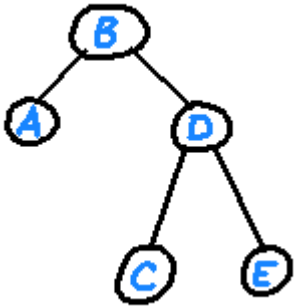
Learning about data structures includes learning to build structures that store their data in different manners, such as…

Storing data linearly, without a meaningful order.

Storing data in a queue or a stack type structure, where you can only access the front of the queue or the top of the stack.



Data stored in a tree structure, which can make searching faster than a list.



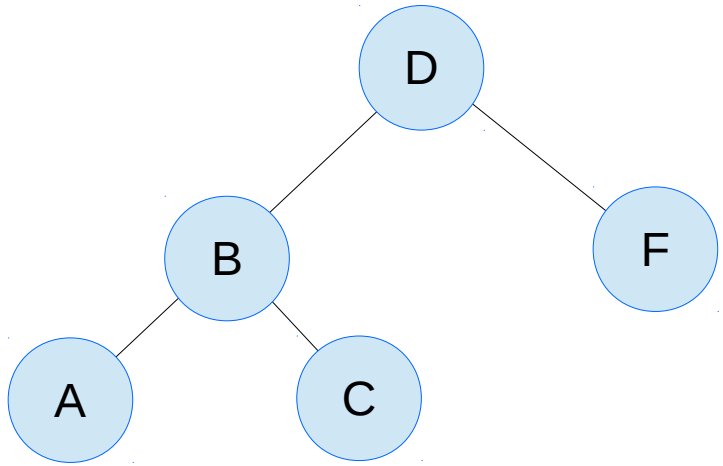
List vs. Tree

As an example, let's store sorted data in a list and in a tree.

List

0	1	2	3	4
A	B	C	D	F

Tree



List vs. Tree

As an example, let's store sorted data in a list and in a tree.

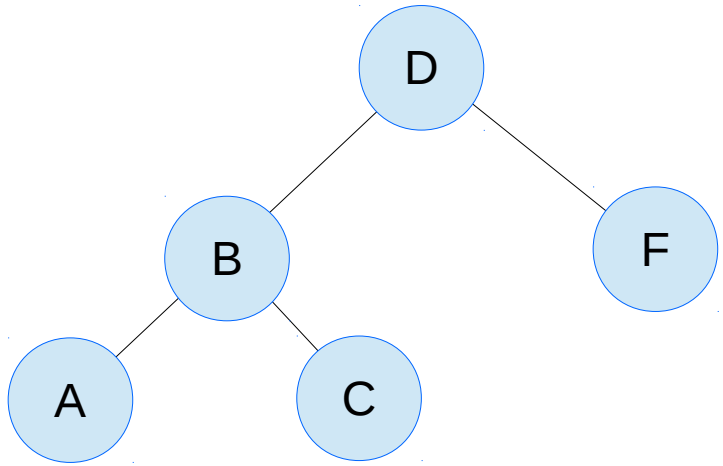
List

0	1	2	3	4
A	B	C	D	F

How would you write an algorithm to find "F" in the list?

If you did it in linear order, you would have to search 4 items before finding "F"!

Tree



List vs. Tree

As an example, let's store sorted data in a list and in a tree.

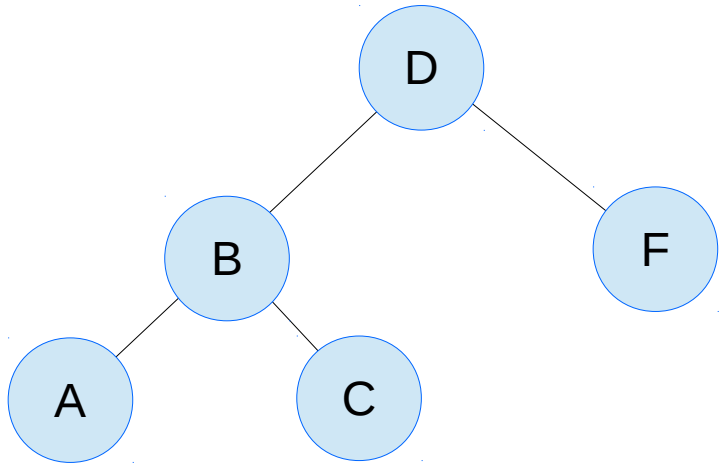
List

0	1	2	3	4
A	B	C	D	F

How would you write an algorithm to find "F" in the list?

If you did it in linear order, you would have to search 4 items before finding "F"!

Tree



For a binary search tree, everything less is to the LEFT, and everything greater is to the RIGHT.

List vs. Tree

As an example, let's store sorted data in a list and in a tree.

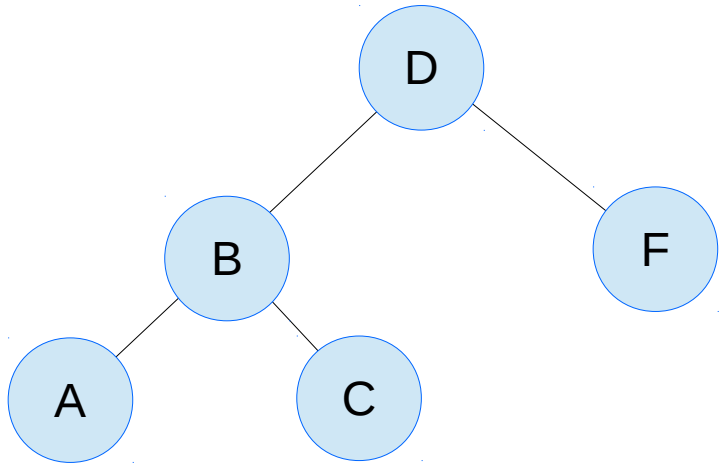
List

0	1	2	3	4
A	B	C	D	F

How would you write an algorithm to find "F" in the list?

If you did it in linear order, you would have to search 4 items before finding "F"!

Tree



For a binary search tree, everything less is to the LEFT, and everything greater is to the RIGHT.

If we start at "D", we just go RIGHT to find "F"!

List vs. Tree

As an example, let's store sorted data in a list and in a tree.

List

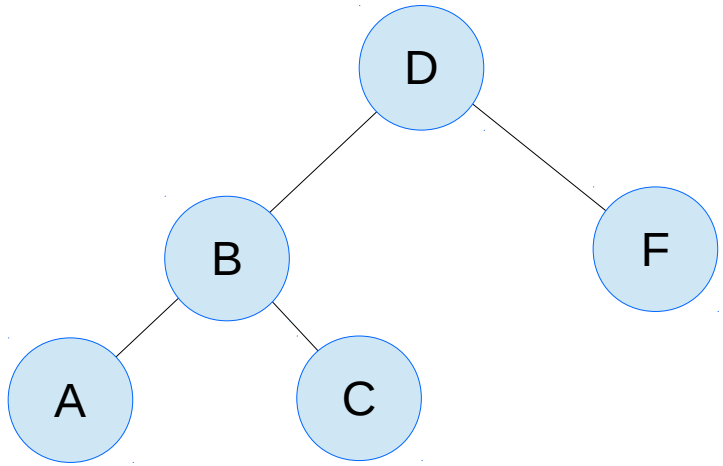
0	1	2	3	4
A	B	C	D	F

How would you write an algorithm to find "F" in the list?

If you did it in linear order, you would have to search 4 items before finding "F"!

Of course, we could write a better searching algorithm… but that's part of what data structures is about, too!

Tree



For a binary search tree, everything less is to the LEFT, and everything greater is to the RIGHT.

If we start at "D", we just go RIGHT to find "F"!

Data Structures topics

In order to study this topic, we will be covering many things throughout the semester…

Programming data structures

- Building a List object from an array
- Building a List object with pointers
- Building a Queue from a List
- Building a Stack from a List
- Building a Tree with pointers
- Building a Dictionary
- & maybe Heaps and Balanced Search Trees (if there's time…)

Data Structures topics

In order to study this topic, we will be covering many things throughout the semester…

Programming data structures

- Building a List object from an array
- Building a List object with pointers
- Building a Queue from a List
- Building a Stack from a List
- Building a Tree with pointers
- Building a Dictionary
- & maybe Heaps and Balanced Search Trees (if there's time…)

Learning about algorithms

- Measuring algorithm efficiency with O(n)
- Searching algorithms
- Sorting algorithms

Data Structures topics

In order to study this topic, we will be covering many things throughout the semester…

Programming data structures

- Building a List object from an array
- Building a List object with pointers
- Building a Queue from a List
- Building a Stack from a List
- Building a Tree with pointers
- Building a Dictionary
- & maybe Heaps and Balanced Search Trees (if there's time…)

Learning about algorithms

- Measuring algorithm efficiency with O(n)
- Searching algorithms
- Sorting algorithms

Writing safe, reusable structures

- Built-in exceptions in C++
- Building our own exception objects
- Using try/catch/throw to deal with errors.

So... how do we program a data structure?

We will be using Object Oriented Programming concepts to build abstract, reusable data structures that can be used on any kind of data.

We *will* review core C++ topics throughout the semester as they come up – such as pointers, operator overloading, and templates, but these are all important to understand as we work on data structures!

So... how do we program a data structure?

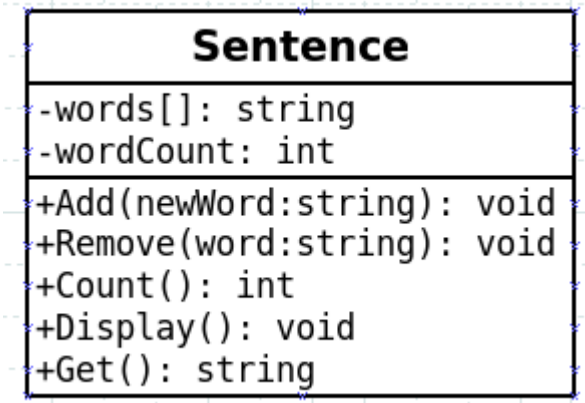
In case you forgot over the break…

- **We can write new objects by creating classes.**
- **Classes contain variables and functions.**
- **We can use pointers to point to memory addresses, which store data.**

So... how do we program a data structure?

In case you forgot over the break...

- We can write new objects by creating classes.
- Classes contain variables and functions.
- We can use pointers to point to memory addresses, which store data.




```
int* ptrCurrent;
ptrCurrent = &studentA;
```

This course focuses on writing classes that deal with data, and provide a nice interface for doing so. For example, "Insert", "Erase", "Find", "Count" functionality and so on.

So… how do we program a data structure?

We will provide the user (generally, a programmer using your data structure) **with an interface for storing data.**

The programmer doesn't need to know *how* things are implemented behind-the-scenes to use our structure; they just need to know that they can add and remove data, and find data.

The way that we implement "Insert" will differ based on what *kind* of structure it is… but the other programmer doesn't need to know that!

```
class ArrayList
{
    public:
    ArrayList();
    void Insert( string newData );
    int GetItemCount();

    private:
    string dataList[100];
    int itemCount;
};
```

```
void ArrayList::Insert( string newData )
{
    if ( itemCount == 100 )
    {
        cout << "Not enough space!" << endl;
        return;
    }

    dataList[ itemCount ] = newData;
    itemCount++;
}
```

So… how do we program a data structure?

Other programmers only know they can use these public functions.

They don't know how things are implemented, and they don't care!

```
class ArrayList
{
    public:
    ArrayList();
    void Insert( string newData );
    int GetItemCount();

    private:
    string dataList[100];
    int itemCount;
};
```

```
void ArrayList::Insert( string newData )
{
    if ( itemCount == 100 )
    {
        cout << "Not enough space!" << endl;
        return;
    }

    dataList[ itemCount ] = newData;
    itemCount++;
}
```

So... how do we program a data structure?

Beyond just storing data, our data structure will also need to handle error checking.

For example: is there some reason we can't add new data to the list?

How do we handle errors?

Do we handle them inside our structure, or do we pass the error to the other programmer to deal with?

(Either could be an appropriate approach!)

```
class ArrayList
{
    public:
    ArrayList();
    void Insert( string newData );
    int GetItemCount();

    private:
    string dataList[100];
    int itemCount;
};
```

```
void ArrayList::Insert( string newData )
{
    if ( itemCount == 100 )
    {
        cout << "Not enough space!" << endl;
        return;
    }

    dataList[ itemCount ] = newData;
    itemCount++;
}
```

So… how do we program a data structure?

As we go through the semester, we will be concerned with how we are <u>implementing</u> these functions (Insert, Remove, etc.)

And with <u>handling errors</u> so that programs don't crash from someone misusing our objects

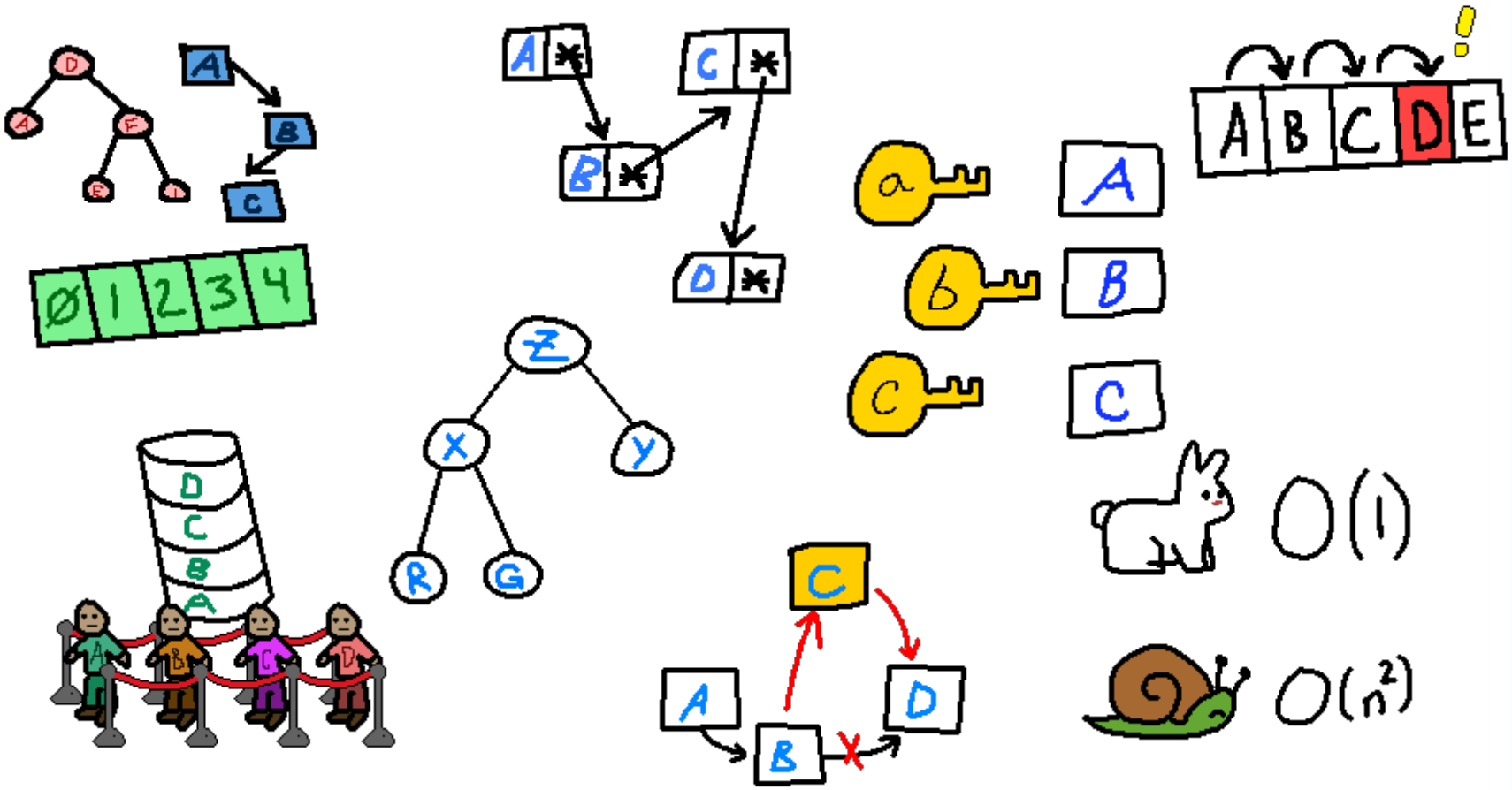
And with providing <u>efficient</u> algorithms for *insertions*, *searching*, etc.

```
class ArrayList
{
    public:
    ArrayList();
    void Insert( string newData );
    int GetItemCount();

    private:
    string dataList[100];
    int itemCount;
};
```

```
void ArrayList::Insert( string newData )
{
    if ( itemCount == 100 )
    {
        cout << "Not enough space!" << endl;
        return;
    }

    dataList[ itemCount ] = newData;
    itemCount++;
}
```

So... how do we program a data structure?

And, that's the class in a nutshell.



Tips for success...

Sometimes it can be overwhelming working with new topics, especially if the algorithms get complex. A few tips for doing well are…

- **If you can't figure out how to implement some functionality, try to sketch out the problem on paper.**
  For example, step through inserting one item, then two items, then three items. How does it work?
- **Getting acquainted with this stuff requires practice, so make sure you have plenty of time to code.**
- **Make sure you have a solid grasp of core C++ topics. If you're feeling shaky in a certain topic… go back and review!**
- **Ask questions if you're feeling stuck.**
  (No judgment… Nobody is born understanding data structures!)

Tips for success...

Sometimes it can be overwhelming working with new topics, especially if the algorithms get complex. A few tips for doing well are…

- **Data structures aren't a secret… there are a lot of resources online that you can reference if the lectures / book aren't doing it for you!**
  There are FREE, OPEN books! Free lessons on YouTube!
  Source code on GitHub and other locations!

- **Make sure you understand *how* each data structure works. You don't need to have the code memorized, but if you know how it works, you should be able to implement it whenever you need.**