

Forum:[Hack Forums](#)

Thread: [DataStructures and Algorithms - Part 2 - LinkedLists 1.1](#)

c-_-m - March 15, 2017, 9:17 am

Linked Lists

So in the last thread we discussed [stacks](#), what they are were and a simple implementation of a stack using a fixed size array. In this tutorial we will discuss linked lists which is a Dynamic Data Structure. Using a linked list will cover some of the issues we faced in the last thread where we used a fixed length array i.e the array got filled.

Referencing(An aside on java)

Almost everything in java is an Object. This excludes our basic data types such as char, double, int, boolean etc...

So lets look at an object:

Object ob;

- here we declare a variable that is of type Object.
- Only enough memory is allocated to hold a memory address
- ob holds a reference, in this case a null reference.

ob = new Object();

- This allocates memory space to an instance of the type Object.
- Returns the address of that memory location and stores it in ob.
- ob is NOT the object but it lets us know where the Object lives.

So basically when we are passing ob around we are passing a reference of the object in this case:

Object p = ob;

- Here p is assigned the same Object reference of ob.
- So they point to the same Object
- This doesn't copy the object

Take a look at this and I'll try make this very clear.

So at the end of this first section:

The value of a is 12 and isUsed is false

The value of b is 10 and isUsed is false

Now the bottom section:

a = b; leaves our value of both a and b as 10 and isUsed still false.

b.value = 34; this line gives both a and b the value of 34 and isUsed still false.

a.value = 20; this also gives both a and b value of 20 and isUsed still false.

Have a mess around with this code if you're unsure and try fully understand what is happening

here.

The reason I have went into this so much is that referencing is the most important thing you must know to understand Dynamic Data Structures.

Nodes

All Dynamic Data Structures are formed from Nodes.

This means they are formed from: A value

(at least) one reference to the same type of Object

The references allow us to move into other Nodes, this means we aren't able to randomly access values like we are able to in Arrays. So basically Nodes have no idea where they are within the data structure.

The way we should view this node is like so:

As we can see the value is stored and also the information for the next node, keep in mind there is no information for the previous node, this is why nodes do not know where they lie within the data structure.

This is how our linked list would look with multiple nodes:

The starting Node is known as the root node.

Once again Nodes are only aware of nodes for which they hold a reference, so for example we could not jump straight from our root node to B without going through A first.

As we can see B does not hold a reference to another node so it's reference is set to null.

Congratulations

You now know everything about Dynamic data structures, the three main structures are Linked-Lists, graphs and trees(each has many variations).

But the only difference between all these is how many references each node can hold and how the nodes can be linked to one another, often called Topology.

In the next couple of threads we will focus on Linked-Lists and then Trees.

Linked-Lists are nodes ordered in a linear sequence, we step along the sequence to access the values in each node. We will go into more detail next time.