Introduction

Hello once again! I had intended to make 2 more of these, however, it is easiest for me to combine all the remaining material into one document that will be decently longer than the average but it will wrap up the necessary material in a way that will satisfy all those who decided to read.

Higher Dimensional Arrays

In the last section we discussed 1-Dimensional Arrays and their uses, now it is time to introduce the idea of multidimensional arrays. Hypothetically you should be able to have N-dimensional arrays but we will stick to 2 dimensions for now

When you think of multidimensional arrays, simply think an array of arrays like this:

```
int[][] a = [[1,2,3],[4,5,6]];
```

Java and other C-based languages are ***row major*** this means for indexing purposes the first index you specify is for the row i.e. to access the 2 we would use the following notation:

```
a[row][column] => a[0][1];
```

We can iterate through every element in this collection using a nested for-loop as follows, remember its rows first then columns:

```
for(int r = 0; r<a.length;r++){
    for(int c = 0; c < a[r].length; c++){
        System.out.println(a[r][c]);
    }
}
```

It is important for the bounds to be correct as it could lead to problems, so please use the notation as given above.

Autoboxed Types

Last section I mentioned that primitive data types like int, double and char have Object type versions of themselves. This is true and goes for all the primitive types, this is what allows us to store primitive data types inside more advanced data structures like Arraylists.

These autoboxed types are called as such because when you pass in a primitive to where only objects are accepted, Java automatically turns the primitive into an Object version of itself. These object types allow us to perform methods to ease ourselves when programming, Integer.parseInt() and Character.toString() are methods that you should most likely already be familiar with and many more methods exist in these classes for you to use.

Inheritance and Polymorphism

One of the key concepts in any Object Oriented programming language is Inheritance and Polymorphism. In Java inheritance is in the form of subclass-superclass relationship. This can be shown through more Java keywords. Consider the following class definition:

```
public class subClass extends superClass {}
```

We use the extends keyword whenever we want to show a relationship between two classes, you always extend the superClass. When you extend a class the subclass inherits all public methods from the superClass that can then be reused in the subClass. This however is not necessary, in the event you want a class to have distinct functionality from its parent in regards to an inherited method, you can simply override the method by redefining it in your own class. Polymorphism means "many forms", and it occurs when we have many classes that are related to each other by inheritance. Polymorphism allows us to use inherited methods in different ways. For example, think of a superclass called Animal that has a method called animalSound(). Subclasses of Animals could be Pigs, Cats etc

Files and Exceptions

Now we want to be able to consider data from outside files. To create a file object in Java:

```
File fileName = new File("filePath");
```

We can then use this File Object to construct a Scanner:

```
Scanner sc = new Scanner(fileName);
```

We can then use a loop to iterate through the contents of a file:

```
while(sc.hasNextLine()){//some code to run}
```

If you want to write to a file you must construct a PrintWriter Object:

```
PrintWriter out = new PrintWriter("outputPath");
```

From there just .println methods to write to the file. Be sure to close the Scanner and PrintWriter when you are done processing the file prior to the program quitting

When you detect an error condition, you should throw the appropriate exception object using the throw keyword:

```
throw new IllegalArgumentException("This shouldn't happen");
```

Sometimes you will want to handle an exception before it reaches the user for these consider a try-catch block

```
try{
    //statements that might throw an exception
}catch(//the exception you want to catch){
    //what you want to happen when exception
    is caught
}
```

You can also attach a finally block to the try-catch block in order to put some code that executes no matter what, a finally block is not required in the structure however!

In Java there are two key types of exceptions: Checked and unchecked exceptions, checked exceptions can occur beyond your control while unchecked exceptions are your fault in the event that you are potentially dealing with an exception in the current method that you cannot handle add the throws clause to the method signature to tell the compiler you are aware of the exception and you expect termination to occur when the exception occurs.

```
public void readData(String filename) throws FileNotFoundException {
      //some code to run
}
```

Finally it is important to understand you can create custom exceptions the same way you
make custom classes just extend RuntimeException in you Class definition when you do it!

Computer Networks

WAN- a collection of local-area networks (LANs) or other networks that communicate with one another. A WAN is essentially a network of networks, with the Internet the world's largest WAN.

MAN- a network with a size greater than LAN but smaller than a WAN. It normally comprises networked interconnections within a city that also offers a connection to the Internet.

LAN- a collection of devices connected together in one physical location, such as a building, office, or home. A LAN can be small or large, ranging from a home network with one user to an enterprise network with thousands of users and devices in an office or school.

The Open Systems Interconnection (OSI) model describes seven layers that computer systems use to communicate over a network. There are seven layers to the model and attached are protocols associated with each layer:

1. Physical Layer: RS232, 100BaseTX, ISDN, 11
2. Data Link Layer: RAPA, PPP, ATM, Fiber Cables
3. Network Layer: IPv5, IPv6, ICMP, IPSEC, ARP, MPLS
4. Transport Layer: TCP, UDP
5. Session Layer: NetBIOS, SAP
6. Presentation Layer: MPEG, ASCH, SSL, TLS
7. Application Layer: HTTP, FTP, POP, SMTP, DNS

The extent of knowledge you should have over these is simply the layer names and some protocols

Dijkstra's Algorithm

Dijkstra's Algorithm is an algorithm that is used for finding the shortest distance, or path, from starting node to target node in a weighted graph Here are the steps:

1. The very first step is to mark all nodes as unvisited,
2. Mark the picked starting node with a current distance of 0 and the rest nodes with infinity,
3. Now, fix the starting node as the current node,
4. For the current node, analyze all of its unvisited neighbors and measure their distances by adding the current distance of the current node to the weight of the edge that connects the neighbor node and current node
5. Compare the recently measured distance with the current distance assigned to the neighboring node and make it as the new current distance of the neighboring node
6. After that, consider all of the unvisited neighbors of the current node, mark the current node as visited
7. If the destination node has been marked visited then stop, an algorithm has ended, otherwise, choose the unvisited node that is marked with the least distance, fix it as the new current node, and repeat the process again from step 4.

This algorithm completes in $O(N^2)$ time

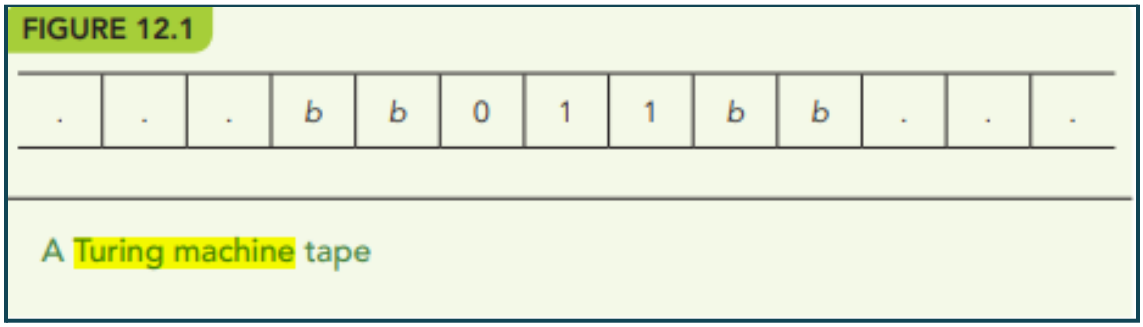
For purposes of this review guide, a fellow TA made a really good video explaining Dijkstra's Algorithm and how to go about solving problems that use it: [Dijkstra's Algorithm: An Overview]

Turing Machines

A Turing machine is a theoretical model of computation that includes a (conceptual) tape extending infinitely in both directions. The tape is divided into cells, each of which contains one symbol.

The Turing machine is designed to carry out only one type of primitive operation. Each time such an operation is done, three actions take place:

1. Write a symbol in the cell (replacing the symbol already there).
2. Go into a new state (it might be the same as the current state).
3. Move the "read head" one cell left or right.



The image above is an example of a turing machine tape. For purposes of Cannon's assignments you start on the leftmost blank spot. Turing machine instructions are formatted as such: (Current State, Current Symbol, Next Symbol, Next State, Direction). The Symbol is what is in the boxes, you will always start in state 1 (unless stated otherwise) and you will perform the given operations until you run out, or potentially run into a loop.

Let's do an example!

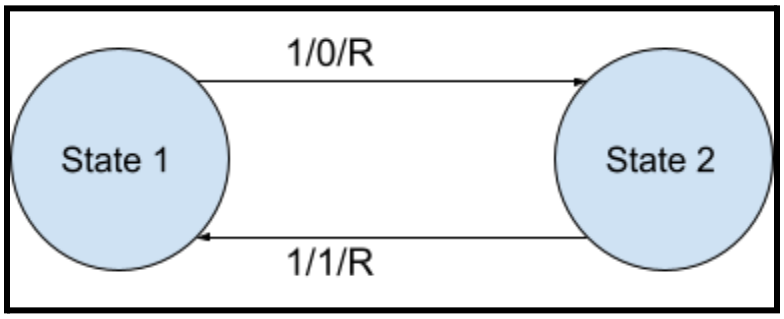
Suppose we have this tape: … b 1 1 1 b … and the following instruction set: (1,1,0,2,R) and (2,1,1,1,R) Determine its output:

Remember we start from the leftmost non blank and start in state 1
First set of instructions to execute is (1,1,0,2,R) so we will write a 0 in its place the state is now a 2 and we move to the right tape is now b 0 1 1 b. Now we execute (2,1,1,1,R) Write a 1 in its place go to state 1 and move right, tape is now b 0 1 1 b, now we execute (1,1,0,2,R) Write a 0 in its place go to state 2 and move right tape is now b 0 1 0 b and we have no more matching instructions! Final tape is: … b 0 1 0 b …

You'll be asked to draw State Diagrams that coordinate with the instruction sets from time to time.

1. Draw a bubble for each state, in the last example there are two state so 2 bubbles
2. Draw arrows to bubbles with labels that contain (current symbol, next symbol, direction)

So for the previous example here is the state diagram for these instructions (1,1,0,2,R) and (2,1,1,1,R):



Anytime the current state and next state match you will have an arrow looping back to the same state!

Conclusion

That concludes the material being covered by these notes sessions for COMS W1004. This has honestly been a really fun experience to make and share these materials with you all. It has been so great being your TA for the semester! These notes will be refined and updated as syllabus content changes over time due to internal changes with the CS Department at Columbia. For those of you who are planning to major in CS, I wish you luck in COMS W3134 Data Structures and Algorithms and the rest of your CS journey. For those of you who took this class to fulfill an elective requirement I wish you the best of luck with whatever path you have decided to take. You will be successful as long as you remain determined.

Best wishes and take care!
-Griffin N