Lecture 12

Time and Space Complexity

```
   // Array is non - primitive
   // Whenever you create an array you get a variable also which contains the size of
the array
   // this variable is called length
   // You access length variable using <arrayName>.length
   // int[] intArray = new int[5]; then intArray.length == 5
   // String[] students = new String[7]; then students.length == 7
   // float[] floArray = new float[15]; then floArray.length == 15
   // String[] students = {"Ishan", "Vivek", "Sandeep"}; then students.length == 3
```

```java
// we are create an array of Strings. Name of the array is students
// Size of array = 9 - students.length
// First index = 0
// Last Index = 8
// String[] students = {"Ishan", "Piyush", "Varun", "Sandeep"};

// Array is non - primitive
String[] students =  new String[4]; // This creates an array named students of type String and size 9
students[0] = "Ishan";
students[1] = "Piyush";
students[2] = "Varun";
students[3] = "Sandeep";

/**
First Approach -
{"Ishan", "Piyush", "Varun", "Sandeep"}

------------------------
Second Approach -
new String[4];
students[0] = "Ishan";
students[1] = "Piyush";
students[2] = "Varun";
students[3] = "Sandeep";
**/
```

For loop syntax for collections

```
String[] students = {"Ishan", "Piyush", "Varun", "Sandeep"};
int[] intArray = {1,2,3,4,5};

// for(<dataType> <variableName> : <arrayVariableName>) {
//     // loop body
// }
for(String studentName : students) {
    System.out.println("Student is " + studentName);
}

for(int number : intArray) {
    System.out.println("Number is " + number);
}
```

```java
// new for loop for collections
for(String studentName : students) {
    System.out.println("Student is " + studentName);
}

// #42 - studentName == Ishan
// #43 - You print the name
// #42 - studentName == Piyush
// #43 -
// #42 - studentName == Varun
// #43
// #42 - studentName == Sandeep
// #43
// 42 - Oh! I have printed all the students
```

Time and Space Complexity

```
// How many seconds/minutes/hours/days/months will this program take to run?
// How do you specify the time taken to run a program? - Time Complexity
// How do you specifiy the memory usage of a program? - Space Complexity
```

Example

We have a bike with maximum speed 100km/hr

We need to go from point A (Delhi) to point B (Chandigarh)

Distance between Delhi and Chandigarh is 300km

What will be the time taken go from Delhi to Chandigarh?

1. Minimum? - 3 hrs - I run the bike at 100km/h the whole route
2. Maximum? - Infinite - I didn’t start the bike
3. Average - Time/Distance - I’m driving at 25km/hr for 25km, then I drive at 50km/h 50km, 100km/h for 25km

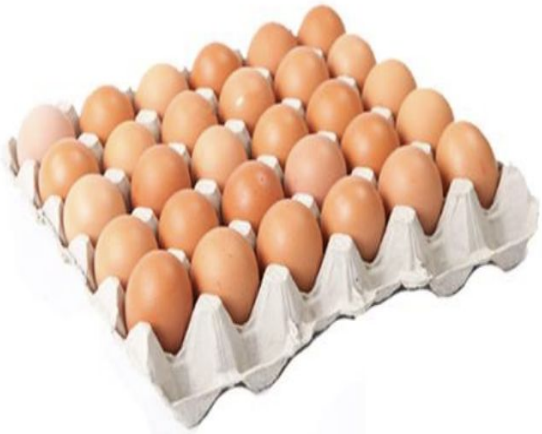
Example

Suppose I have a crate of eggs with 6 eggs.

For every egg that 1 eat I can eat up to 6 oranges.

If I eat all the eggs then how many oranges can I eat?

- Maximum (Worst Case) - 36
- Minimum (Best Case) - 0
- Average (Average Case) - average of oranges eaten



Boats Examples

I have a boat whose minimum speed is 1km/hr and maximum speed is 15km/hr

I need to go from point A to point B and the distance between those points is 30kms

How much time will I take?

Assume that I always start the boat.

- Best Case (Minimum) - 2hrs
- Worst Case (Maximum Time) - 30hrs
- Average Case - Time to reach/30

How do you judge if one program is better than other program?

When we are running computer programs.

We measure their

- Worst Case
- Best Case
- Average Case

Which one is better according to runtime (Let's assume we are measuring time in minutes and hours for now)?

Program 1

- Worst Case - 1hr
- Best Case - 30 min

Program 2

- Worst Case - 5 hrs
- Best Case - 5 min

Program 1 is better because its worst case runtime is better than Program 2

Which one is better according to runtime (Let's assume we are measuring time in minutes and hours for now)?

Program 1

- Worst Case - 1hr
- Best Case - 30 min

Program 2

- Worst Case - 5 hrs
- Best Case - 5 min

Program 1 is better because its worst case runtime is better than Program 2

Which one is better according to memory (Let's assume we are measuring memory in bits/bytes/mb/gb for now)?

Program 1

- Worst Case - 50 GB
- Best Case - 1 GB

Program 2

- Worst Case - 10 GB
- Best Case - 5 GB

Program 2 is better because its worst case memory usage is better than Program 1

```
int[] intArray = {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15........ infinite}; // size of the array is 10

for(int number : intArray) {
    System.out.println("Number is " + number); // How many times will this statement be printed? - 10
}

/**
    If 1 print statement takes 1 unit time to execute
    then,
    If array size == 5 then runtime == 5 units
    If array size == 10 then runtime == 10 units
    If array size == 15 then runtime == 15 units
    ...
    If array size == Infinity then runtime ==  infinite units

    So the worst case runtime (Time Complexity) depends on the size of the array

    If I denote size of the array by some variable N
    then Time Complexity  = O(N)
**/
```

Time Complexity Notations

Worst Case - Denoted by Big O Notation

Best Case - Denoted by Omega Notation

Average Case - Denoted by Theta Notation

best case (Ω) / average case(θ) / worst case (O)time complexity

How do we measure the runtime?

We measure runtime on the size of the input.

```java
int[] intArray = {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15..... infinite}; // size of the array is 10

for(int number : intArray) {
    System.out.println("Number is " + number); // How many times will this statement be printed? - 10
}

/**
    If 1 print statement takes 1 unit time to execute
    then,
    5 units
    10 units
    15 units
    infinite units

    The number of units taken to run this program == the size of the array
    // worst case time complexity  == size of the array units
    // Worst Case - O(size of the input)
    // Worst Case - O(n) .. where n is the variable which denotes the size of the input
**/
```