Exercises

Here are some exercises for you to work through. They should help you to get a clearer idea (or confirm your understanding of the 'Big O' notation. Remember:

- These are NOT fragments of Java code - They describe the algorithm and a lot of details have been removed.
- .... and don't try to understand them, they sort of make sense but you don't need to understand the algorithm to do the analysis
- The aim is to express how the algorithm changes it's behaviour (how much time it takes) as the amount of data increases

Given the following algorithms, what is their time complexity using the Big O notation?

1.

```
max = 0
For i = 0..(n-1)
   If A[i]>max
      max=A[i]
```

This is O(n)

2. Do the same with the following:

```
numberFound = 0
For i = 0..(n-1)
   For j = 0..(n-1)
      If A[i,j] == target
         numberFound = numberFound + 1
```

Here there are 2 nested loops both dependent on n – so, O(n^2)

3. In this example you should assume that the function BinarySearch is O(log n)

```
numberFound = 0
For i = 0..(n-1)
   For j = 0..(n-1)
      If BinarySearch(A[i,j])
         numberFound = numberFound + 1
```

Both loops vary with n. Binary search is O(log n)

So, this will be O(n^2 log n)

4. This example is a little more complex:

```
max = 0
For i = 0..(n-1)
   If A[i]>max
      max=A[i]

numberFound = 0
For i = 0..(n-1)
```

```
   For j = 0..(n-1)
      If A[i,j] == target
         numberFound = numberFound + 1
```

In the first loop we have time complexity O(n)

The second is O(n^2)

We take the maximum so the whole algorithm is O(N^2)

5. .... and what about this one?

```
numberFound = 0
For i = 0..(n-1)/2
   For j = 0..(n-1)/2
      If BinarySearch(A[i,j])
         numberFound = numberFound + 1
```

Both loops are O(N). Binary search is O(log n).

So, the whole loop is O(n^2 log n)

6. .... and one final one. Assume that randomInt(n) returns a random integer in the range [0..(n-1)] and that A is an array of length n

```
i = randomInt(n)
return (a[i])
```

This is not dependent on n – It's constant time.

This is written O(1)