Exercise Sheet, Week 3

1 Determining the time complexity

The running time of an algorithm usually grows in proportion to the size of the input. To an algorithm, we assign a complexity class which determines the speed of such growth. Let's say that the size of the input is n (e.g. an array consisting of n elements). Then, an algorithm which takes $\leq 4n + 3123$ steps is in $\mathcal{O}(n)$.

When determining the complexity class, we strip away the (leading) constants. For example, algorithms taking $\leq 23492n$, $\leq 31n + 10$, or $\leq 2n + 10^{30}$ steps are all in $\mathcal{O}(n)$. In general, if the number of steps is less than or equal to $an + b$, for some constants a and b, then the algorithm is in $\mathcal{O}(n)$. Similarly, the algorithms which need $\leq an^2 + b$ steps belong to $\mathcal{O}(n^2)$, or with $\leq a \log n + b$ steps belong to $\mathcal{O}(\log n)$, etc.

An algorithm which multiplies all elements in the array:

```
int product(int[] arr) {
  int n = arr.length;
  int x = 1;
  int i = 0;

  while (i<n) {
    x = x * arr[i];
    i++;
  }

  return x;
}
```

An algorithm which modifies the last value in the array:

```
void modify(int[] arr) {
  if (arr.length == 0)
    throw Exception;

  int last = arr[arr.length - 1];

  if (last < 0) {
    last = -last;
  }

  arr[arr.length - 1] = last;
}
```

Finding the largest element of the array in three different ways:

```
int largest1(int[] arr) {
  int n = arr.length;
  int max = 0;

  for (int i=0; i<n; i++) {
    bool largest = true;

    for (int j=0; j<n; j++) {
      if (arr[i] < arr[j])
        largest = false;
    }

    if (largest)
      max = arr[i];
  }

  return max;
}
```

```
int largest2(int[] arr) {
  int n = arr.length;
  int max = 0;

  if (arr.length == 0) {
    return 0;

  } else {
    max = arr[0];

    for (int i=0; i<n; i++) {
      if (arr[i] > max)
        max = arr[i];
    }

    return max;
  }
}
```

```
int largest3(int[] arr) {
  sort(arr);

  if (arr.length == 0) {
    return 0;
  } else {
    int last = arr[arr.length - 1];
    return last;
  }
}
```

In `largest3` assume that `sort(arr)` is in $\mathcal{O}(n \log n)$.

To which complexity classes do those algorithms belong? Possible answers $\mathcal{O}(1)$, $\mathcal{O}(n)$, $\mathcal{O}(n \log n)$, $\mathcal{O}(n^2)$.

Procedure	Complexity Class
`product`	
`modify`	
`largest1`	
`largest2`	
`largest3`	

Hint: To start with, try what happens if you call `product(arr)` with `arr = [3, 4, 1, 12, 3]`. How many steps will it take? And what if `arr = [3, 4, 1, 12, 3, 1, 2, 33]`? Try the same strategy for the other algorithms as well.

2 Comparison of complexity classes

If the number of steps is smaller than $7n^2+2$ then it is also smaller than $7n^3+2$. This is because, for n getting larger and larger, the number n^3 grows faster than n^2. Moreover, the constants 7 and 2 can be replaced by any other constants and so any algorithm which is in $\mathcal{O}(n^2)$ is also in $\mathcal{O}(n^3)$. This justifies that we can see $\mathcal{O}$ to mean "proportional or less".

Example: If an algorithm takes less than $21n^3+6n+12$ steps then it is in $\mathcal{O}(n^3)$ because

$$\text{the number of steps} \leq 21n^3+6n+12 \leq 21n^3+6n^3+12 = 27n^3+12.$$

Indicate to which complexity class the algorithm belongs based on the number of steps it takes:

number of steps	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^3)$
1	✓	✓	✓	✓
$\leq n$	✗	✓	✓	✓
$\leq n^2$	✗	✗	✓	✓
≤ 341243				
$\leq 25n+72$				
$\leq n^2+n$				
$\leq n^3+n^2+n$				
$\leq 12+n\times(34+n)$				
$\leq n^3+4\times(1+31n+4n)$				
$\leq n\times(3n+n\times(83+n))$				
$\leq n^5$				
$\leq 2^n$	✗	✗	✗	✗

3 Exponentials and Logarithms

Recall that, for a number a and a positive integer n, we defined

$$a^n = a\times a\times\cdots\times a \text{ (repeated } n \text{ times)}, \qquad a^{-n}=\frac{1}{a^n} \quad\text{and}\quad a^{1/n}=\sqrt[n]{a}.$$

We also defined $\log_a b$ to be the value c such that $a^c=b$. Solve the following:

1. $2^2 = \ ?$, $3^2 = \ ?$, $4^2 = \ ?$, $5^2 = \ ?$
2. $2^3 = \ ?$, $3^3 = \ ?$, $4^3 = \ ?$, $5^3 = \ ?$
3. $4^3 = \ ?$, $4^{-2} = \ ?$, $4^{1/2} = \ ?$, $8^{1/3} = \ ?$
4. For which n is $\frac{16}{2^n}=1$? And when is $\frac{30}{2^n}\leq 1$? What is the ceiling $\lceil \log_2 30\rceil$?
5. $4^{3/2} = \ ?$, $5^{31}/5^{28} = \ ?$, $(\sqrt[2]{3})^6 = \ ?$
6. $\log_3 27 = \ ?$, $\log_5 \sqrt[3]{5} = \ ?$, $\log_2 \sqrt[3]{4} = \ ?$