Assignment 1

1. $n^3 + 625 \leq n^3 + 625n^3$ for $n \geq 1$

$n^3 + 625 \leq 626n^3$ for $n \geq 1$

take $c = 626$ and $n_0 = 1$

$n^3 + 625 \leq cn^3$ for all $n \geq n_0$ and

$\therefore O(n^3)$

2. prove by contradiction

$$\frac{\frac{n^3}{16}}{n} \leq \frac{cn}{n}$$

$$\frac{n^2}{16} \leq c$$

This would mean that there exists a constant c and a number n_0 where $n^3/16 \leq cn$ for all $n \geq n_0$. Therefore $n^2/16 \leq c$ for all $n \geq n_0$. Since n can be any number greater than n_0, there will eventually be a number n that makes $\frac{n^2}{16} > c$ as $n^3/16$ is a continous function. Therefore the statement $\frac{n^3}{16} \leq cn$ contradicts itself, meaning that $\frac{n^3}{16}$ is NOT $O(n)$

3. Algorithm(A, B, n)

Input: Arrays A and B of size n
A, B store positive integers

```
1.  i ← 0
2.  sum ← 0
3.  while i < n do
4.      if A[i] < n then
5.          for j ← 0 to A[i] do
6.              sum ← sum + B[j]
7.      i ← i + 1
8.  return sum
```

	Best (assuming n = 0)	Worst (assuming A = [0, 1, ..., n-1])
1.	1 primitive operation	1 primitive
2.	1 primitive	1 primitive
3.	1 does not enter as i = n	n will happen n times
4.		n will happen n times
5.	–	n(n-1) will happen n(n-1) times
6.	–	n(n-1) will happen n(n-1) times
7.	–	n will happen n times
8.	1 primitive	1 primitive

∴ Best $O(1)$

∴ Worst $O(n^2)$

4. Algorithm rearrange(A, n)
Input: Array A of size n

A stores positive integers

```
1.    for i ← 0 to n
2.        if A[i] % 2 != 0 and i != 0 then
3.            current ← A[i]
4.            for j ← i to 1
5.                A[j] = A[j-1]
6.                j = j - 1
7.            A[0] = current
8.        i = i + 1
9.    return A
```

Worst case complexity (array contains all odd numbers)

1.	n	will happen from 0 → n times
2.	$2n$	will happen 2n times as there are 2 comparisons
3.	$n-1$	variable will be created n times
4.	$(n-1)^2$	will happen n-1 times bc of outer and will occur n-1 times bc of inner
5.	$(n-1)^2$	
6.	$(n-1)^2$	
7.	$n-1$	
8.	n	happen 0 → n times
9.	1	primitive

adding all the time complexities up you get

$O(n^2)$