Algorithm Analysis
Handout 2
Deadline October 10

Exercise 2.1
For each of the following equations show whether it is true or false:

1. $x^2 = O(x^3)$
2. $x^3 = O(x^2)$
3. $5x^3 + 3x^2 = \Omega(x^4)$
4. $3x^2 + 5x + 2 = \Theta(x^2)$
5. $2^{n+1} = O(2^n)$
6. $3^n = O(2^n)$
7. $3^n = \Omega(2^n)$

Exercise 2.2
The Max-Heapify algorithm in your textbook has a recursive definition:

```
MAX-HEAPIFY(A, i)
 1  l ← LEFT(i)
 2  r ← RIGHT(i)
 3  if l ≤ heap-size[A] and A[l] > A[i]
 4     then largest ← l
 5     else largest ← i
 6  if r ≤ heap-size[A] and A[r] > A[largest]
 7     then largest ← r
 8  if largest ≠ i
 9     then exchange A[i] ↔ A[largest]
10          MAX-HEAPIFY(A, largest)
```

Give an iterative (loop – based) definition of the Max-Heapify algorithm.

Exercise 2.3
Using Figure 6.4 on page 161 in the textbook as a model, illustrate the operation of Heap sort on the array **A = [6, 12, 1, 28, 7, 19, 22, 8, 2]**

Exercise 2.1

1. True(Square is lower bound to cubic. Cubic is much larger than square.)
2. False(Cubic is upper bound to square.)
3. False(x^4 is upper bound to cubic. But the omega means is it upper bound of the given equation?, so it's not true.)
4. True(The theta means is it average of the given equation?, so the equation's degrees are same with square and square.)
5. True(2^(n+1) means 2*2^n, and 2 can be removed.)
6. False(3^n is much larger than 2^n)
7. True(3^n is upper bound to 2^n)

Exercise 2.2

```
while ( i < = heapsize) {
 le <- left(i)
 ri <- right(i)
 if (le<=heapsize) and (A[le]>A[i])
  largest <- le
 else
  largest <- i
 if (ri<=heapsize) and (A[ri]>A[largest])
  largest <- ri
 if (largest != i)
 {
   exchange A[i] <-> A[largest]
   i <- largest
 }
 else
  break
}
```

Exercise 2.3

[6, 12, 1, 28, 7, 19, 22, 8, 2]
[6, 28, 1, 12, 7, 19, 22, 8, 2]
[6, 28, 22, 12, 7, 19, 1, 8, 2]
[28, 6, 22, 12, 7, 19, 1, 8, 2]
[28, 12, 22, 6, 7, 19, 1, 8, 2]
[28, 12, 22, 8, 7, 19, 1, 6, 2] Since line 1 to here(line6), this process is build up the max heap
[22, 12, 19, 8, 7, 2, 1, 6] **[28]**
[19, 12, 6, 8, 7, 2, 1] **[22, 28]**
[12, 8, 6, 1, 7, 2] **[19, 22, 28]**
[8, 7, 6, 1, 2] **[12, 19, 22, 28]**
[7, 2, 6, 1] **[8, 12, 19, 22, 28]**
[6, 2, 1] **[7, 8, 12, 19, 22, 28]**
[2, 1] **[6, 7, 8, 12, 19, 22, 28]**
[1] **[2, 6, 7, 8, 12, 19, 22, 28]**

Each line(from line 7 to 14) is Max-heapify.(I removed the process building max heap.)

∴ **[1, 2, 6, 7, 8, 12, 19, 22, 28]**