WIA2005 Algorithm Design &Analysis
Semester 2
Tutorial 1

---

1. The following is an insertion sort algorithm.

```
def InsertionSort(A):
    for j in range(1,len(A),1):
        key = A[j]
        # insert A[j] into the sorted sequence A[1..j-1]
        i=j-1
        while (i>=0 and A[i]> key):
            A[i+1]=A[i]
            i=i-1
        A[i+1]= key
```

Illustrate the insertion sort operation on array A = 41, 51, 69, 36, 51, 68.

41,(51),69,36,51,68

j=1   key=51   i=0   tak lalu while (!41>51)   A[i+1]=51

41,51,(69),36,51,68

j=2   key=69   i=1   tak lalu while (!51>69)   A[i+1]=69

41,51,69,(36),51,68

j=3   key=36   i=2   lalu while (69>36)

A[i+1]=36
A[i]=69
letak no. besar ke tmpt no. kecik
new A[i+1]=69
i=1

41,51,69,69,51,68   key=36
A[i+1]=69
A[i]=51
letak no. besar ke tempat no. kecik
new A[i+1]=51
i=0

41,51,51,69,51,68   key=36
A[i+1]=51
A[i]=41
letak no. besar ke tempat no. kecik
new A[i+1]=41

i=-1

keluar loop
A[i+1]=36

36,41,51,69,(51),68

j=4 key=51 i=3 lalu while (69>51)
36,41,51,69,(51),68
A[i+1]=51//i+1 will always refer to the key value
A[i]=69
new A[i+1]=69
i=2

keluar dari loop since (51=51)
A[i+1]=51//overwrite 69

36,41,51,51,69,(68)

j=5 key=68 i=4 lalu while (69>68)
36,41,51,51,69,(68)
A[i+1]=68
A[i]=69
new A[i+1]=69//copy A[i] value
i=3

36,41,51,51,69,69 key=68

keluar dari loop since (51<68)//lagi kecik dari the key
A[i+1]=68

36,41,51,51,68,69

2. Modify the insertion sort algorithm to sort array into decreasing order.

```
def InsertionSort(A):
  for j in range(1,len(A),1):
    key=A[j]
    i=j-1
    while (i>=0 and A[i]<key):
      A[i+1]=A[i]
      i=i-1
    A[i+1]=key
  return A

print(InsertionSort(A))
```

3. Write a pseudocode for linear search for the following requirement:
   Input: A sequence of n numbers A = $\langle a_1, a_2, \ldots, a_n \rangle$ and a value v.
   Output: An index i such that v = A[i] or the special value NIL if v does not appear in A.

```
Declare empty array List[]
For i=0 to Array.length
    If A[i] ==v
        List.add(A[i])

If List.length>0
    return list
else
    return NIL
```

```
for i=0 to length(A)
        if A[i]==v
            return i
    return NIL
```

4. Express the function n^3 / 1000 – $100n^2$ – 100n + 3 in terms of Θ-notation.

   $\Theta(e) = \Theta(\max(n^3/1000, -100n^2, -100n, 3))$.

   $\Theta(1) < \Theta(n) < \Theta(n^2) < \Theta(n^3)$

   Then

   $\Theta(e) = \Theta(\max(n^3/1000, -100n^2, -100n, 3)) = \Theta(n^3)$

5. For the following pairs of functions, f(n) and g(n), determine if they belong to Case 1: f(n) = O(g(n)) or Case 2: g(n) = O(f(n)). Formally justify your answer.

   a. f(n) = 3n +2 , g(n) = n
      Case 2: g(n) = O(f(n))
      because $f(n) \leq cg(n)$ for $n > n_0$
      For example for n=1000
      f(n) = 302 while g(n) =100

   b. f(n) = $(n^2 - n)/2$, g(n) = 6n
      Case 2: g(n) = O(f(n))
      because $O_f(\max(n^2,n)) = O_f(n^2)$
      $O_g(n) < O_f(n^2)$ for $n > n_0$

   c. f(n) = $n+2\sqrt{n}$, g(n) = n^2
      Case 1: f(n) = O(g(n))
      because $O_f(\max(n, \sqrt{n})) = O_f(n)$
      $O_g(n^2) > O_f(n)$ for $n > n_0$

   d. f(n) = $n^2 + 3n + 4$, g(n) = n^3

      Case 1: f(n) = O(g(n))

because $O_f(\max(n^2, n)) = O_f(n^2)$
$O_g(n^3) > O_f(n^2)$ for $n > n_0$

6. Given the iterative function below (in Java), calculate their time complexity.

a.
```
function1 (){
      for (int i = 1; i <= n; i ++) {
            printf("Hello world");
      }
}
```

T(n) = O(n)

b.
```
function2(){
      for (int i = 1; i <=n; i ++) {
                  for (int j = 1; j <=n; j ++) {
                        printf("Hello world");
        }
    }
```

T(n) = O(n^2)

c.
```
function3 (){
      for (int i = 1; i² <= n; i ++) {
            printf("Hello world");
      }
}
```

T(n) = O($n^{0.5}$)

d.
```
function4 (){
      for (int i = 1; i <= n; i = i*2) {
            printf("Hello world");
      }
}
```

T(n) = O($\log_2$(n))

e.
```
function3(){
      for (int i = n/2; i <=n; i ++) {
            for (int j <= 1; j <=n/2; j = 2*j) {
                  for (int k = 1; k <= n; k*2) {
                        printf("Hello world");
                        }
                  }
            }
      }
}
```

T(n) = O(n[$\log_2$(n)]^2)= O(n[$\log_2^2$(n)])