Fundamentals of Data Structures
Algorithm Analysis
Example test questions for the course

These questions or similar questions have been used in tests in previous years. The course has been taught using various program languages, as a consequence the program text in these examples may be written in Java, Pascal, Turing or pseudo code.

---

1. Use big Theta notation to describe the worst-case running times of each of the following operations.

   A Removing an item with a given key from an AVL tree. _____

   B Rebalancing an AVL tree after an item with a given key has been inserted but the tree has not yet been rebalanced. _____

   C Rebalance an AVL tree after an item with a given key has been removed but the tree has not yet been rebalanced. _____

   D Inserting an item with a given key into a skip list. _____

   E Removing an item with a given key from a skip list. _____

   F Performing a union operation on two sets A and B assuming that there does not exist a total order on the elements in A and B. _____

   G Performing a union operation on two sets A and B assuming that there exists a total order on the elements in A and B. _____

   H Performing a depth-first traversal on a directed graph. _____

   I Using the Floyd-Warshall algorithm to determine whether there exists a path between two given vertices in a directed graph. _____

---

2. Consider the following algorithm.

```
for i ← 1 to 100 do
    for k ←1 to n do
        j ← 1 ; m ← n
        while j < m do
            m ← (m + j) / 2
        end while
    end for
end for
for i ← 1 to 30 do
    for j ← 1 to n do
        k ← i + j + n
    end for
end for
for i ← 1 to 70 do
    j ← 2*n + i
end for
```

What is its exact running time with respect to n?
What is its big O running time? Prove it using its mathematical definition.
What is its big Omega running time? Prove it using its mathematical definition.

What is its big Theta running time? Prove it using its mathematical definition.

---

3. Give the big O for the following algorithms.

A Circle the appropriate time complexity with respect to N.

```
procedure A(S : string)
    int N := length(S)
    int j := 2 * N
    loop
        exit when j <= 0
        int i := N
        loop
            exit when i <= 1
            i := i div 2
        end loop
        j := j - 1
    end loop
end A
```

$O(N^2)$ $O(N)$

O(log N) $O(N \log N^2)$

O(N log N) O(1)

infinite $O(2^N)$

None of the above

B Circle the appropriate time complexity with respect to N.

```
procedure B(S : string)
    int N := length(S)
    int i := 1000
    int k := 0
    while i > 1
        for j: 1..N*N
            if (j < N) then
                k += 1
            end if
        end for
        i := i - 1
    end loop
end B
```

$O(N^2)$ O(N)

O(log N) $O(N \log N^2)$

O(N log N) O(1)

infinite $O(2^N)$

None of the above

C Circle the appropriate time complexity with respect to N.

```
procedure C(S, Q : string)
    int N := length(S)
    int P := length(Q)
    for k: 1..P*P
        int j := 1
        while j <= P
            j := j * 2
        end loop
    end for
end C
```

$O(N^2)$ O(N)

O(log N) $O(N \log N^2)$

O(N log N) O(1)

infinite $O(2^N)$

None of the above

D Circle the appropriate time complexity with respect to N.

```
procedure D(S : string)
    int N :int := length(S)
    int k := 0
    for i := 1..N
        k += 1
    end for
    for i := 1.. N
        for j := 1..N
            k += 1
        end for
    end for
    for i := 1..N
        for j := 1..N
            k += 1
        end for
    end for
end D
```

$O(N^2)$ $O(N)$

$O(\log N)$ $O(N \log N^2)$

$O(N \log N)$ $O(1)$

infinite $O(2^N)$

None of the above

E Circle the appropriate time complexity with respect to N.

```
function Rec(N : int) : int
    Rec(N-1)
    if N == 1 then
        return 1
    end if
end Rec
```

$O(N^2)$ $O(N)$

$O(\log N)$ $O(N \log N^2)$

$O(N \log N)$ $O(1)$

infinite $O(2^N)$

None of the above

F Circle the appropriate time complexity with respect to N.

```
function Rec2(N : int) : int
    if N == 1 then
        return 1
    else
        Rec2(N-1)
        Rec2(N-1)
    end if
end Rec
```

$O(N^2)$ $O(N)$

$O(\log N)$ $O(N \log N^2)$

$O(N \log N)$ $O(1)$

infinite $O(2^N)$

None of the above

G Circle the appropriate time complexity with respect to N.

```
function FA (N : int) : int
    if N ≤ 1 then
        return 1
    else
        FA (N-N/2)
        FA (N-N/2)
    fi
end
```

$O(N^2)$ $O(N)$

$O(\log N)$ $O(N \log N^2)$

$O(N \log N)$ $O(1)$

infinite $O(2^N)$

None of the above

H Circle the appropriate time complexity with respect to N.

```
function FB (N : int) : int
    if N ≤ 1 then
        return 1
    else
        FB (N-100)
    fi
end
```

$O(N^2)$	O(N)
O(log N)	$O(N \log N^2)$
O(N log N)	O(1)
infinite	$O(2^N)$
None of the above	

I What is the big O of the following function? Write your answer in the box.

```
function FC (N : int) : int
    if N ≤ 1 then return 1
    else
      FC (N-1); FC (N-1); FC (N-1)
    fi
end
```

J Sort the following running times from largest (on the left) to smallest (on the right):

$O(N^2)$, $O(N \log N)$, $O(10^N)$, $O(1)$, $O(N^3)$, $O(\log N)$, $O(2^N)$, $O(N)$

K Circle true or false.

$5N^2 + 47N + 11000 + 6N^2$ has complexity $O(N^4)$ TRUE FALSE

L Circle true or false.

$5N^2 + 47N + 11000 + 6N^2$ has complexity $\text{Theta}(N^4)$ TRUE FALSE

---

4. Describe the execution time in big O notation for each of the following operations on a basic sequence when each of the following data structures are used to implement the basic sequence abstract data type.

	Circular Array	Singly Linked List	Doubly Linked List	Simple Array
Insert a new element at the head				
Remove an return the last element of the sequence				
Return the number of elements in the sequence				
Remove and return the first element of				

the sequence				

5. While experimental studies on running times are useful, they have three major limitations. Describe the three major limitations.

6. Describe what is meant by the big O notation in algorithm analysis.

7. Describe when we decide to use an O(n**2) algorithm as opposed to an O(n log n) algorithm.

8. How is the structure of a program related to the big O execution time analysis of the program? Consider both polynomial time and exponential time algorithms.

9. Show that the expression $10\ N^2 + 1000\ N + 1000000$ is $O\ (N^2)$.

10. Show that the expression $10\ N^2 + 1000\ N + 1000000$ is $O\ (N^2)$.

11. Describe what is meant by big Ω notation in algorithm analysis.

12. Show that the expression $10\ N^2 - 1000\ N - 1000000$ is $\Omega\ (N^2)$.