DSA

INTERNSHALA

Introduction to Algorithms

What is an Algorithm?

- An algorithm is a finite set of instructions.
- It’s a textual representation of flow of logic.
- It’s independent of any programming language.
- There is no fixed standard syntax for writing algorithms. Various authors use different kinds of syntax.
- There are following three programming constructs in algorithms-
  - Sequence
  - Selection (decision making)
  - Iteration (looping)

Sequence in Algorithms

```
1. Set a = 100
2. Set b = 200
3. Set c = a + b
4. Print c
5. End
```

Selection in Algorithms

```
1. Set a = 100
2. Set b = 900
3. if a>b, then:
       Write: a is bigger
       else
       Write: b is bigger
4. END
```

```
1. Read a
2. Read b
3. if a>b, then:
        write: a is bigger
        else
             if b>a, then
             write: b is bigger
             else
             write: a and b are equal
4. END
```

We can use [] to write comments bw algos.

Loops in algorithm

[Algo to print all natural numbers from 1 to 100 using repeat while loop]

```
1. Set num = 1
2. Repeat steps 3,4 while (num<=100)
3. Write: num
4. Set num = num + 1
5. END
```

[Algo to print all the even numbers from 2 to 100]

```
1. Set num = 2
2. Repeat steps 3,4 while (num<=100)
3. Write: num
4. Set num = num + 2
5. END
```

[Algo to print all natural numbers from 1 to 100 using repeat for loop]

```
1. Repeat 2 for num = 1 to 100
2. Write: num
3. END
```

[Algo to print all the odd numbers from 1 to 100]

```
1. Repeat 2 for num = 1 to 100 by 2
2. Write: num
3. END
```

[Finding sum of Array elements]

```
1. set c =1        [initializing value of counter with 1]
2. repeat steps 3,4 while (c<=10)
3. set sum = sum + sc[c]
4. set c = c + 1
5. write: sum
6. END
```

[using repeat for loop]

```
1. Repeat for c = 1 to 10
2. Set sum = sum + sc[c]
   [end of for loop]
3. Write: sum
4. END
```

[finding the biggest element in array]

```
1. Set biggest = arr[1]
2. Repeat for c = 2 to N
3. If arr[c] > biggest, then:
         Set biggest = arr[c]
         [end of if]
   [end of for]
4. Write: biggest
5. END
```

[Difference between return and end statement]

END statement means the end of program.

RETURN means that the function is returning some value to calling function.

CALL means a function is being invoked.

[Matrix addition]

```
1. Set I = 1
2. Set J = 1
3. Repeat steps 4,7,8 while I<=3
4. Repeat steps 5,6 while J<=3
5. Set c[I][J] = a[I][J] + b[I][J]
6. Set J = J + 1
   [end of J loop] [inner loop for columns]
7. Set I = I + 1
8. Set J = 1
   [end of I loop] [outer loop for rows]
9. END
```

[Matrix Multiplication]

```
   Set i = 1  [for traversing rows of first matrix]
   Set j = 1  [for traversing columns of second matrix]
   Set k = 1  [for traversing columns of first and rows of second
   matrix]
1. Repeat for i = 1 to 3
2. Repeat for j = 1 to 3
3. Repat for k = 1 to 3
4. Set sum = sum + a[i][k]*b[k][j]
   [end of innermost loop k]
5. Set c[i][j] = sum
6. Set sum = 0
   [end of middle loop j]
   [end of outer most loop i]
7. END
```

Complexity of Algorithms

- Time Complexity
    - The time taken by the algorithm to complete it’s task
- Space Complexity

    - Amount of space taken by the algorithm to complete it’s task

Big O Notation

- This is a mathematical notation to evaluate algorithms
- This is to observe the way time and space requirements grow as the input size grows.

Why do we need it?

- It’s not practical to talk about the exact runtime of an algorithm.
- Runtime is dependent on the CPU speed.
- We use Big O Notation to judge how runtime grows.
- Size of the input is used represented by n.

Terminology

- Runtime will grow “on the order of the size of the input”

  $$O(n)$$

- Runtime will grow “on the order of square of the size of the input”

  $$O(n^2)$$