I)
VSCS201

DAA - Design Analysis and Algorithms

Author - [illegible], Karomachi

1 Dec 2022

* Ipo cycle :- the f (input process output)
- any product is to design, analysed based on a algorithmes.

Defn - Algorithms is define as the step by step procedure to required to produce or complete a task

2 Dec 2022

Q Write the Algorithm to display sum of 3 No.

→ Accept a, b, c
Total = a+b+c
Display Total.

Best Algorithmes - → less time to complete / execute / run
→ occupies less space / memory

→ The every program takes some time to complete. then we can use the time module (labalary) For the calculating time, which the the module. program taken for complition.

→ some time system system clock was not an. fixe right place.

$$\begin{bmatrix} 3 & 4 \\ 5 & 6 \\ 7 & 8 \end{bmatrix}_{3\times 2}$$

m1 = [[3,4], [5,6], [7,8]]
m2 = [[1,2], [3,4], [8,9]]

6 loops required

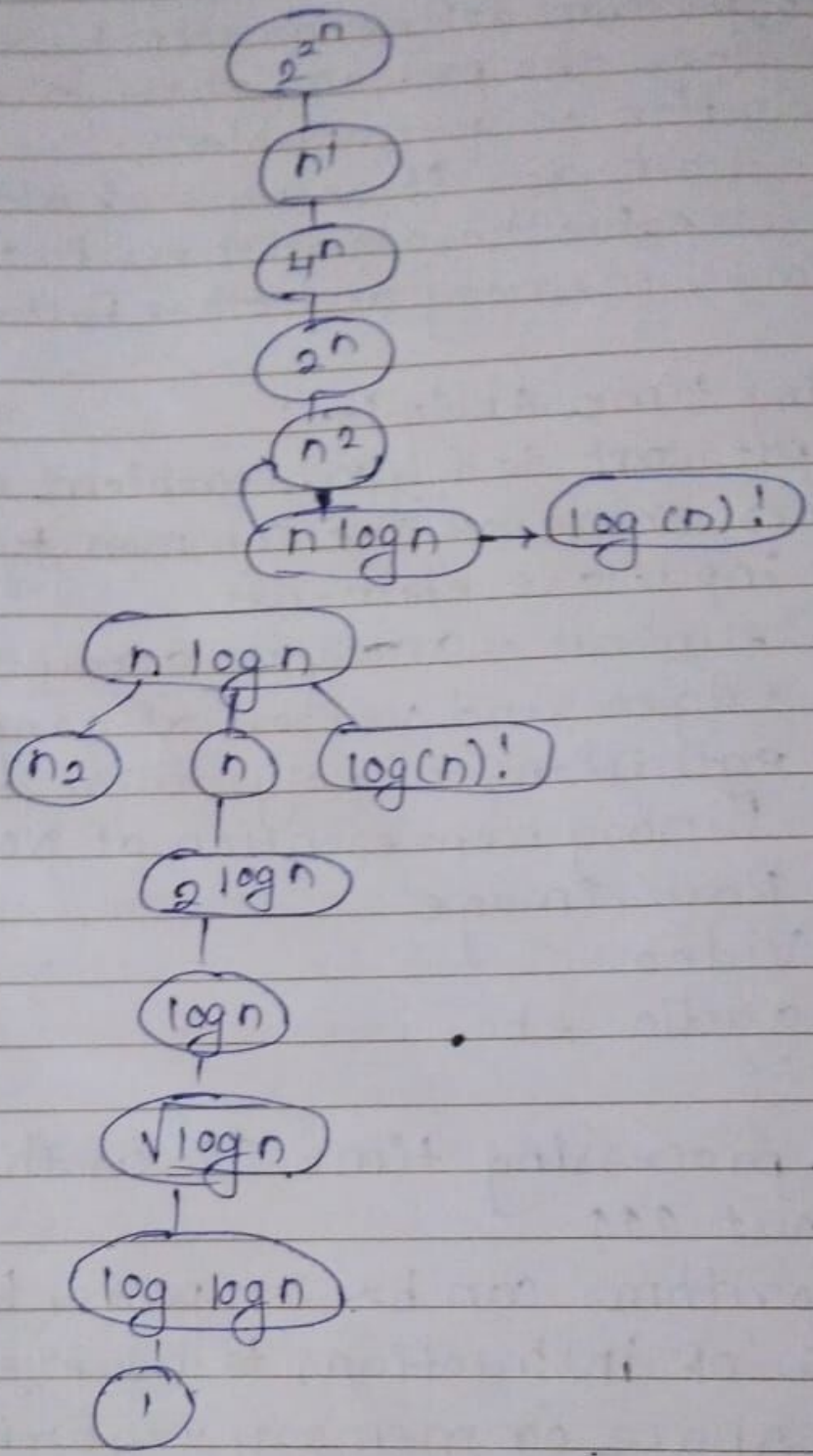
```
for x in range (len[m]):
    for y in range (len(m.[0]):
```

- An Algorithm defines as step by step instruction to slove the problem. there are many solution on given problem.
- The main goal of analysis of alogorithms is to compaire the alogorithms in terms of runing time, memory and other factors

* Runing time Analysis :
- input wort to a given problem can be of many types some of the common type of input
  1) input are elements
  2) elements of ~~array~~ analysis array
  3) edges and vertex of graph
  4) equations of polynomial degree
  5) Binary represention of No.
  6) Raw Image
  7) Video
  8) audio etc.

- The processing time ↑↑ as the size of the input ↑↑↑
- Algorithms can be compaires based on time, No. of instructions to be executed, amount of space or memory requried for execution
- The rate of growth of an algorithms is a fn of input

* Comanlyused rate of growth.



3 Dec 2022 (types)

3 cases of the Analysis of the Algorithms:-

- Best case
- Worst case
- Average case.

Q Write a python code to print factorial of a No. for n = 5

```
import timeit
result = 1
i = 1
n = int(input("Enter a Number"))
while i <= n :
    result = result * i
    i = i + 1
print ("Factorial of", n, result)
print ("Running time", timeit.timeit())
```

- Rate of growth of input.

47, 52, 55, 57, 62, 69

Q Write a python program to sort a list of data.

```
import timeit
w1 = [62, 52, -55, 0, 57, 8]  # w1, wort case
w2 = [-55, 0, 8, 52, 57, 62]  # w2, Best case
w3 = [0, 8, -55, 52, 57, 62]  # w3, Average case
print ("w1", w1)
print ("Sorted w1", w1.sort())
print ("Execution time of w1", timeit.timeit())
print ("w2", w2)
print ("Sorted w2", w2.sort())
print ("Execution time of w2", timeit.timeit())
print ("w3", w3)
print ("Sorted w3", w3.sort())
print ("Execution time of w3", timeit.timeit())
```

① Definations of case

Types of Analysis

- to analyis the algorithm we need to know the what input takes less time, and what input the Algorithm takes long time
- Algorithmes are given as simple instructions & complete algorithmes make up of many instructions Accordinaly we have 3 types of analyiss

(1) Best case - It defines the I/P in which Algorithm takes lowest time it has the I/P where the algorithm runs fastest.

(2) Wort case - It defines the I/P for which the Algorithm takes long time. it has the I/P which algorithm runs slower.

(3) Average case - It provides a predication of runing time of the algorithm. it assusms the I/P is random.

* Asymptotic Notation - construct the Eqn

$f(n)$ $n^2 + 500$ $n = 5 \quad = 525$

$n = 100 \quad = 600$

$f(n) = n + 100n + 500$

- In All the 3 cases Best, wort, Avg. we need to find upper bound & lower bound
- lower bound < Averagetime > upper bound time
- every algorithm is represented in a form of Function
- Call the Function $f(n)$.

- Based on function f(n) we have Big-O-Notation we write as $O(n)$ which gives tight upper bound of the given function f(n)

2) Big-Θ-Notation Fn Θ(n)
- gives tight lower bound of the given function f(n)

3) Big-omega-Notation $\Omega(n)$
- gives Average analysis of the given function f(n).

List processing # method 1

program to obtain sum of elements in a given list.

```
import timeit
mylist = [44, 23, 0, -85, 1234]
total = 0
for data in mylist :
        total = total + data.
print ("sum of element in list", total)
print (" Running time timeit. timeit())
```

— sum of element in list 1216

running time 0.02492259

Method 2

```
import timeit
mylist = [44, 23, 0, -85, 1234]
total = 0
x = len (mylist).
x = x - 1
i = 0
while i <= x :
        total = total + mylist [i]
        i = i + 1
print ("sum of elements in list", total)
print ( " Runing time timeit. timeit () )
```

— sum of elements in list = 1216

time = 0.02200

#

```
import timeit
mylist = [44, 23, 0, -85, 1234)
total = 0
for data in mylist :
        total = total + data.
print ( "sum of elements", total)
```

```
# method 3
import timeit
total = 0
n = len(mylist)
for i in range(0, n):
        total = total + mylist[i]
print("sum of the list", total)
print("Runing time", timeit.timeit)
```

sum = 1216 time = 0.02170

Q. WAP to find. ~~Compare~~ the time of elements for the list size 10

~~# method 1~~

Q. WAP to count the even & odd elements from the list.

```
# program to count even & odd.
import timeit
mylist = [23, -12, 48, 999, 84, 51].
odd_ctr = 0
even_ctr = 0
for data in mylist:
    if data % 2 == 0:
        even_ctr = even_ctr + 1
    else:
        odd_ctr = odd_ctr + 1
print("odd count", odd_ctr)
print("even count", even_ctr)
```

3

3