A.I.S.S.M.S
INSTITUTE OF INFORMATION TECHNOLOGY
Kennedy Road, Near R.T.O., Pune - 411 001.

Assignment - 2
C.S - 5084

1] given running times

a) n^2 → $f_1(n) = n^2 \in g_1(n) = n^2$
b) n^3 → $f_2(n) = n^3 \in g_2(n) = n^3$
c) $100n^2$ → $f_3(n) = 100n^2 \in g_3(n) = (n^2)$ 100 ⇒ const.
d) $n \log n$ → $f_4(n) = n \log n \in g_4(n) = n \log(n)$
e) 2^n → $f_5(n) = 2^n \in g_5(n) = 2^n$

1) lets double the input size
∴ n will be 2n

∴ $F_1(n) = (2n)^2 = 4n^2 \in g_1(n) = n^2$ ∵ 4 cost.
$F_2(n) = (2n)^3 = 8n^3 \in g_2(n) = n^3$ ∵ 8 const.
$F_3(n) = 100(2n)^2 = 400n^2 \in g_3(n) = n^2$ ∵ 4×100 cost.
$f_4(n) = 2n \log 2n \in g_4(n) = n \log n$
$F_5(n) = 2^{(2n)} = 4 2^n \in g_5(n) = 2^n$ ∵ 4 cost.

the g(n) would be same for all but algorithms gets slower as follows.

f_1 gets slower by [4] times
f_2 gets slower by [8] times
f_3 gets slower by [4] times.
f_4 gets slower by [2] times approx.
& f_5 gets slower by [4] times.

2) lets increase size of input by 1

$F_1(n+1) = (n+1)^2 = n^2+2n+1$
$F_2(n+1) = (n+1)^3 = n^3+3n^2+3n+1$
$F_3(n+1) = 100(n+1)^2 = 100n^2 + 200n + 100$
$F_4(n+1) = (n+1)\log(n+1)$
$F_5(n+1) = 2^{(n+1)} = 2 \cdot 2^n$

if we just take difference from original term.

$F_1(n+1) - f_1(n) = n^2+2n+1 - n^2 \Rightarrow 2n+1$
$F_2(n+1) - f_2(n) = n^3+3n^2+3n+1 - n^3 \Rightarrow 3n^2+3n+1$
$F_3(n+1) - f_3(n) = 100n^2+200n+100 - 100n^2 \Rightarrow 200n+100$
$F_4(n+1) - f_4(n) = \log(n+1)$
$F_5(n+1) - f_5(n) = 2 \cdot 2^n - 2^n = 1(2^n)$

F_1 gets 2 times slower
F_2 gets ~~increased by~~ slower by 3 times
F_3 gets slower by 2 times slower.
F_4 gets logarithmically slower.
F_5 doesn't affect.

(we ignored the constants)

3] ascending order of growth rate.

~~$F_3(n) = n + 10$~~
$F_2(n) = \sqrt{2n}$ the $F_3(n) = n + 10$
$F_6(n) = n^2 \log n$
$F_1(n) = n^{2.5}$
$F_4(n) = 10^n$
$F_5(n) = 100^n$

F_2 & F_3 has linear growth,
F_6 has logarithmic growth.
F_1 has polynomial growth
then F_4 & F_5 has logarithmic growth.

6] For the given steps of algo.

```
For
   for
   ——
   ——
   endfor
Endfor
```

a] here for loop is running 2 times

For each step of outer loop (n) inner loop runs runs (n−i) times

∴ running time is $O(n^2)$

(b) to show that same f is $O(n^2)$ $\Omega(f(n))$

As per

as per the definition of Big omega (for lowerbounds)

$f(n)$ is $\Omega(g(n))$ if there exist const $c>0$, $n_0 \geq 0$ such that $f(n) \geq c \cdot g(n) \geq 0$ $\forall n \geq n_0$

here in this case to consider lower bound the outer loop must run at least

'One time, for which inner loop will run $n-1$ time. apparently order remains same for algo. i.e n^2

$\therefore$ f is $\Omega(n^2)$

[7] ~~approximise the algo. we can perform operations.~~

[8] a] for budget of $k=2$ jars,
first jar we'll drop from $\frac{1}{2}^{st}$ rung.
second jar will drop from middle of rung.
if the jar doesn't break, then we drop the second jar from middle of rung to 1^{st} rung.
thus, the $f(n)$ will be slightly slower than linear f^n which would be logarithmic

b] with budget of $k > 2$ jars.
Strategy would be to follow binary search algorithm. but here partition of ht. is $k-2$
At each step we drop the jar from middle of rung then repeat the proceces till the jar does not break.
Thus, this function also grows logarithmically & will be slower than previous steps or functions. such that f_k is slower than $f_{k-2} \ldots f_1$