14/8/21

DATE □□ □□ □□□□

Tutorial Sheet - 1

(1) int a = 0, b = 0;
for (i = 0; i < N; i++) a += rand();
for (j = 0; j < M; j++) b += rand();

Time complexity = O(N + M);
Space Comp = O(1) Ans.

(2) int sum = 0, i;
for (i = 0; i < n; i = i + 2) { sum += i; }

↳ loop will sum ~~add~~ even ~~#~~ times.

∴ 0 + 2 + 4 - - - - 2n.

= ~~2 (n)(n+1)~~ / 2 =

Time complexity

O(n/2) ⇒ O(n) Ans

(3) int sum = 0, i;
for (i = 0; i < n; i = i * 2) { sum += i }

Time complexity = ~~O(n²)~~ O(log n) Ans

(4) int sum = 0, i
for (i = 0; i*i < n; i++)

O(log n) Ans.

(5) int j = 1, i = 0;
while (i <= n) {
i = i + j;
j++;
}

Time = O(n)
Space = O(1)
Ans

Vardhman PAGE

⑤ void recursion(int n){

if(n==1) return; → T(1)

recursion(n-1); → T(n-1)

print(n); → T(1)

recursion(n-1); → T(n-1).

}

$$T(n) = T(n-1) + 1$$

↳ $T(n-1) = T(n-2) + 1$

$T(n) = T(n-2) + 2$ $\quad T(1) = 1$

$T(n) = T(n-k) + k$

↳ $O(n)$ Ans

⑥ The given Recursive fun is a binary search

∴ Time complexity = ~~O(n log n)~~

$$T\left(\frac{n}{2}\right) + 1$$

As we dividing by 2 every time

∴ $O(\log n)$

Ans

(6) (1) $T(n) = T(n-1) + 1$

$T(n-1) = T(n-2) + 1$

$T(n) = T(n-2) + 2$

$T(n) = T(n-k) + k$

$= O(n)$ Ans

(2) $T(n) = T(n-1) + n$

$T(n-1) = T(n-2) + (n-1)$

$T(n) = T(n-2) + n-1 + n$

$T(n-2) = T(n-3) + n-2 + n-1 + n$

$T(n) = T(n-3) + n-2 + n-1 + n$

$T(n) = T(n-k) + kn - \frac{(k-1)k}{2}$

(3) $T(n) = T(n/2) + 1$

$a=1, b=2, k=0, p=0$

$a = b^k$

$O(n^k \log^{p+1} n)$

$O(\log n)$ Ans

(4) $T(n) = 2T(n/2) + 1$

$a=2, b=2, k=1$

$2 > 2^1$ $(a > b^k)$

$O(n^{\log_b a}) = O(n)$

Ans

(5) $T(n) = 3T(n-1)$

$T(0) = 1$ (given)

$T(n-1) = 3T(n-2)$

$T(n) = 3(3T(n-2))$

$T(n) = 3^k T(n-k)$

$n-k = 0$

$\boxed{n = k}$

$3^n \times 1 = T(n)$

$T(n) = 3^n$

$O(3^n)$ Ans

DATE

(9) $O(n)$

(10) $O(N * N)$

(11) $O(N \log N)$

(12) X will be always a better choice for large inputs

(13) $O(\log N)$

(14)

(15) $f_1(n) = n^{\sqrt{n}}$
$f_2(n) = 2^n n$
$f_3(n) = (1.00001)^n$
$f_4(n) = n^{(1.0+2^{(n+1)})}$

$f_4 > f_2 > f_3 > f_1$

(16) $f(n) = 2^{(2n)}$

$\Omega(2^n n)$

(19) $T(n) = 7T(n/2) + 3n^2 + 2$

using Master's Theorem

$a = 7, \quad b = 2, \quad k = 2$

$a > b^k$

$7 > 2^2$: True

$T(n) = \Theta(n^{\log_2 7})$

$= \Theta(n^{2.8})$

(17) $T(n) = 2T(n/2) + n^2$,

$a = 2$, $b = 2$, $k = 2$

$2 < 2^2$ $\boxed{p = 0}$

$T(n) = O(n^k \log^p n)$

$T(n) = O(n^2)$ Ans

(18)
```
int gcd (int n, int m) {
    if (n * m == 0) return m;
    if (n < m) swap (n, m);
    while (m > 0) { n = n - m;
                    swap (n, m); }
    return m; }
```

↳ here n is gradually / exponentially decreasing

$\therefore$ $O(\log n)$ Ans

(19)
```
int a = 0, b = 0;
for (i = 0; i < N; i++) {          → O(n)
    for (j = 0; j < N; j++)        → O(n)
        a = a + j;
}                                  O(n^2)
for (k = 0; k < N; k++)            ← O(n)
    b = b + k;
```

$\therefore$ $O(n^2) + O(n) + O(n)$

$O(n * n + n + n)$

$O(n^2 + n)$

$= O(n^2)$ Ans