Time Complexity

Date: / /

① Frequency Count Method →

I

1) Sum (A, n)

{ s = 0; ————————— 1 → bothered abt this

for (i = 0; i < n, i++) ——— n+1

loop goes for n times   s = s + a[i]; ——→ n

return s; ——— 1

}

Total time → $1 + n+1 + n + 1$

$$f(n) = 2n + 3$$

$$T = O(n)$$

2)

for (i = 0, i < n, i++) —— n+1

inside loop n time [for (J = 0; J < n, J++)   n × n+1

inside loop n times [C[i][J] = A[i, J] + B[i, J]   n×n

$$f(n) = 2n^2 + 2n + 1$$

$$= O(n^2)$$

3)

for (i = 0; i < n; i++)

    for (J = 0; J < i, J++)

    } }

Analysing

i	J	Times
0	0 ×	0
1	0 ✓ 1 ×	1
2	0 ✓ 1 ✓ 2 ×	2
3	0 ✓ 1 ✓ 2 ✓ 3 ×	3

m $\qquad$ n times

total time $\Rightarrow 1+2+3+4+ \cdots n$

$= \frac{n(n+1)}{2}$ Date: $O(n^2)$

4) $p=0$

for (i=1; p<=n; i++)

{ p=p+i }

Analyse

i	p
1	$0+1=1$
2	$1+2=3$
3	$1+2+3=6$
k	$1+2 \cdots +k$

Assume $p>n$

$p = \frac{k(k+1)}{2}$

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

$k^2 > n$

$k > \sqrt{n}$ → stop after this cond^n

$T = O(\sqrt{n})$

5) for (i=1; i<n; i=i*2)

{ };

Analyse.

i	t
1	2^0
1×2	2^1
2×2	2^2
2×2×2	2^3
	2^k

$2^k < n$

$k < \log_2 n$

$O(\log_2 n)$

6) for (i=0; i<n; i++)

{ } $\rightarrow n$

for (j=0; j<n; j++)

{ } $\rightarrow n$

$O(n)$ $\quad f(n)=2n$

7) for (i=0; i<n; i++) $\rightarrow n+1$

{ for (j=1; j<n; j=j*2) $\rightarrow n \times \log n$

} }

$O(n \log n)$

II Analysis of ~~for~~ & while

Date: / /

① $i=0$ $\boxed{n+1}$

~~while (i<n)~~

{ i++; }

$O(n)$

② $i=1$ $k=1$

while (k<n)

{ k = k+i i++ }

Analysing

i	k
1	1+1
2	1+1+2
3	1+1+2+3
:	
~~m~~	~~1~~+1+2+3+4---+m

$k = \frac{m(m+1)}{2}$

Stop when

$k < n$

$\frac{m(m+1)}{2} < n$

$m^2 < n$

$m < \sqrt{n}$

$O(\sqrt{n})$

② for wf analyse by taking eg

$O(1)$ → constant

$O(n)$ → linear

$O(\log n)$ → logarithmic

$O(n^2)$ → Quad

$O(n^3)$ → cubic

$O(2^n)$ → ...

Comparison of classes →

$1 < \log n < \sqrt{n} < n < n\log n < n^2 < n^3 \dots < 2^n < 3^n \dots < n^n$

Date: / /

1) Big O → worst case

if $f(n) \le c \times g(n) \quad \forall\, n \ge n_0$ & c +ve.

then $f(n) = O(g(n))$

2) Theta → average

$c_1 g(n) \le f(n) \le c_2 g(n)$

$f(n) = \Theta(g(n))$

3) big Omega → best

$c_1 g(n) \le f(n)$

then $f(n) = \Omega\, g(n)$

Method to compare →

1) $n^2 \quad n^3$

take log

$2\log n < 3\log n$

2) $n^2 \log n \quad n(\log n)^{10}$

take log

$\log n^2 \log n \quad \log n (\log n)^{10}$

$\underline{2\log n} + \log\log n \quad \underline{\log n} + 10\log\log n$

$2\log n > \log n$

$\log\log n < 10\log\log n$ → this is smaller

$n^2\log n > n(\log n)^{10}$

$\log_a b$

$\log ab = \log a + \log b$

$\log a/b = \log a - \log b$

$\log a^b = b\log a$

$a^b = n$

then $b = \log_a n$

$a^{\log_c b} = b^{\log_c a}$

Recurrence relation

Date: / /

1) $T(n) \rightarrow$ time taken for f^n

void Test (int n) — $T(n)$
{ if (n > 0) — 1
for (i=0; i<n; i++) — n+1
print — n
Test (n-1) — $T(n-1)$

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

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

$T(n)$
n — $T(n-1)$
n-1 — $T(n-2)$
n-2 — $T(n-3)$
$T(0)$

Time $\rightarrow \frac{n(n+1)}{2}$

$T(n) = n^2$

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

2) By subst.

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

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

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

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

$T(n) = T(n-k) + (n-(k-1)) + n-(k-2) \dots (n-1) + n$

Assume $n-k=0$ because it goes till $n=0$

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

$= T(0) + 1 + 2 \; --- \; (n-1) + n$

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

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

Q) $T(n)$

Test(n) ——— $T(n)$

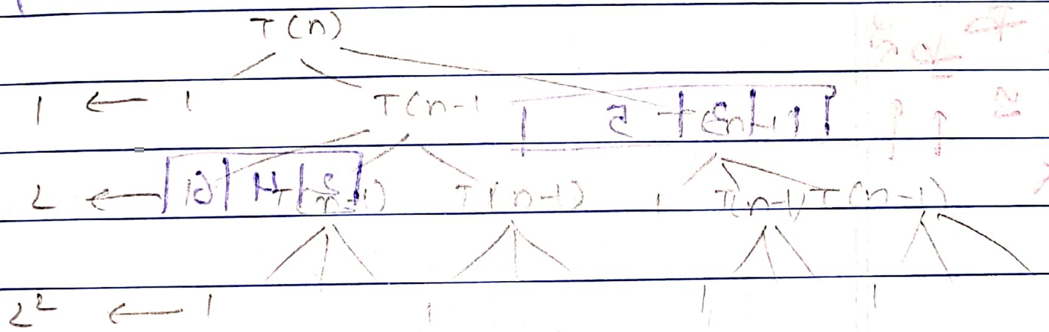
{ if $n>0$

{ print

Test(n-1) ——— $T(n-1)$

Test(n-1) ——— $T(n-1)$

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



$1 + 2 + 2^2 \; --- \; 2^k = 2^{k+1} - 1$ (GP)

$= O(2^n)$

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

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

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

$= 2^k \, T(n-k) + k$

$n - k = 0$

$= 2^n \, T(0) + n$

$= O(2^n)$