12th February, 2024

DAA UNIT I VIBHA

Algorithm : Sequence of unambiguous instructions for solving a problem.

Problem

↓

Algorithm

↓

input ⟶ [computer] ⟶ output

→ Trial & Error

q.1 $12n^2 + 8 \in O(n^2)$

$\quad f(n) \qquad\qquad g(n^2)$

$f(n) \leq c \times g(n)$

$n \geq n_0$

$$12n^2 + 8 \leq 12n^2 + 8n^2$$
$$\leq 20n^2$$

$n_0 = 1$

$c = 20$

q2. $100n + 5 \in O(n)$

$$100n + 5 \leq 100n + 5n$$
$$\leq 105n$$

$n_0 = 1, \ C = 105$

q3. $13n^2 + n + 5 \in O(n^2)$

$$13n^2 + n + 5 \leq 13n^2 + n^2 + 5n^2$$
$$\leq 19n^2$$

$n_0 = 1, \ C = 19$

q4. $n^3 \in \Omega(n^2)$

$$f(n) \geq c\,g(n)$$

$$n^3 \geq n^2$$

$n_0 = 0, 1$

$c = 1$

q5. $n^2 + n \in \Theta(n^2)$

$\cancel{n^2 + n \leq n^2}$

$$n^2 \leq n^2 + n$$

$$n^2 + n \leq n^2 + n^2 \quad n_0 = 0$$

$$n^2 + n \leq 2n^2$$

$c_1 = 2$

$c_2 = 1$

Limit Theorem

$$\lim_{n \to \infty} \frac{t(n)}{g(n)} = \begin{cases} 0, & \text{odg } t(n) < g(n) \\ c > 0 & \Rightarrow t(n) = g(n) \\ \infty & \Rightarrow t(n) > g(n) \end{cases}$$

* Stirling's Formula

$$n! \approx (2\pi n)^{1/2} \left(\frac{n}{e}\right)^n$$

q.1 Compare ~~gos~~ of log n and n
$t(n)$ $g(n)$

$$\lim_{n \to \infty} \frac{\log n}{n} = \frac{1/n}{1} = \frac{1}{n} = 0$$

$$\Rightarrow t(n) \ll g(n)$$

little o

$$\Rightarrow t(n) \in o(n)$$

$$\log n \in o(n)$$

q2. $t(n) = \frac{1}{2}(n)(n-1)$ $g(n) = n^2$

$$\lim_{n \to \infty} \frac{1}{2} \frac{\cancel{n}(n-1)}{n^{\cancel{2}}} = \lim_{n \to \infty} \frac{1}{2} - \lim \frac{1}{2n}$$

$$= \frac{1}{2} = c > 0$$

$$\Rightarrow t(n) \in \theta(n^2)$$

small theta

$$\frac{1}{2} n(n-1) \in \theta(n^2)$$

q3. $t(n) = \log n$ $g(n) = \sqrt{n}$ $n^{1/2}$ $\frac{1}{2\sqrt{n}}$

$$\lim_{n \to \infty} \frac{\log n}{n^{1/2}} = \frac{1/n}{1/2\sqrt{n}} = 2\frac{\sqrt{n}}{n} = 2\left(\frac{1}{2\sqrt{n}}\right)$$

$$= 0$$

little o

$$\log n \in o(\sqrt{n})$$

Efficiency of non recursive functions

Formulas

$$\sum_{i=l}^{u} 1 = u - l + 1$$

$$\sum_{i=1}^{n} 1 = n \in \Theta(n)$$

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

$$\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$$

$$\sum_{i=0}^{n} a^i = \frac{a^{n+1} - 1}{(a-1)}$$

q1. Max element

$$t(n) = \sum_{i=1}^{n-1} 1$$

$$= n - 1 - 1 + 1 = n - 1$$

$$\lim_{n \to \infty} \frac{n-1}{n} = 1 \Rightarrow n - 1 \in \Theta(n)$$

q2. unique element

$$t(n) = \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} 1$$

$$= \sum_{i=0}^{n-2} (n - 1 - i - \cancel{1} + \cancel{1})$$

$$= \sum_{i=0}^{n-2} (n - 1 - i)$$

$$= n-1 + n-2 + \cdots + 1$$

$$= \frac{(n-1)(n)}{2}$$

$$\lim_{n \to \infty} \frac{n^2 - n}{2(n^2)} = \lim_{n \to \infty} \frac{2n-1}{4n} = \lim_{n \to \infty} \frac{2}{4} = \frac{1}{2}$$

$$\frac{n(n-1)}{2} \in \Theta(n^2)$$

q.3 Metrix multiplication

$$t(n) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \sum_{k=0}^{n-1} 1$$

$$= \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} n$$

$$= \sum_{i=0}^{n-1} n^2 = n^3 \in \Theta(n^3)$$

Efficiancy of Recursive algo

q1. Factorial

$$f(n) = f(n-1) * n$$

ABDUL Bari

1.10.1 Compare $f(n) = 3n^{\sqrt{n}}$ $g(n) = 2^{\sqrt{n}\log_2 n}$

$$3n^{\sqrt{n}} \qquad 2^{\log_2 n^{\sqrt{n}}}$$

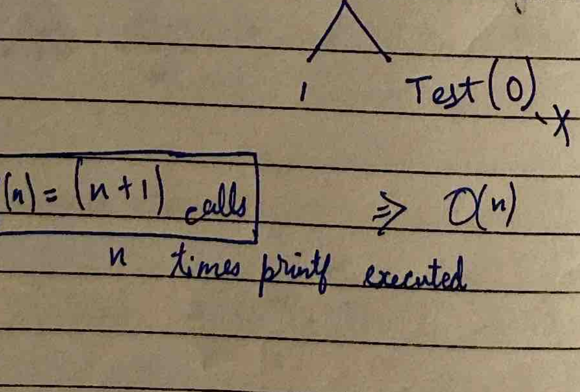
$$3n^{\sqrt{n}} \qquad \left(n^{\sqrt{n}}\right)^{\log_2 2}$$

$$3n^{\sqrt{n}} > n^{\sqrt{n}}$$

2.1.1 eg.

```
Void Test (int n)
{
    if (n>0)
    {
        printf("%d", n);
        Test (n-1);
    }
}
```

eg: Test(3)



$\boxed{f(n) = (n+1) \text{ calls}}$ $\Rightarrow O(n)$

n times printf executed

Date
Page

Recurrence Relation

```
time T(n) — void Test(int n)
            {
                if (n>0)
                {
   1     ———————    printf("%d \n", n);
   T(n-1) ————— Test(n-1);
                }
            }
```

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

$$T(n) = \begin{cases} 1 & n=0 \\ T(n-1)+1 & n>0 \end{cases}$$

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

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

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

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

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

⋮ continue for k times

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

Assume $n-k = 0$ $\therefore n = k$

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

$T(n) = T(0) + n \Rightarrow \boxed{T(n) = 1+n}$

$O(n)$

2.1.2 eg 2. $T(n)$ — Void Test (int n)

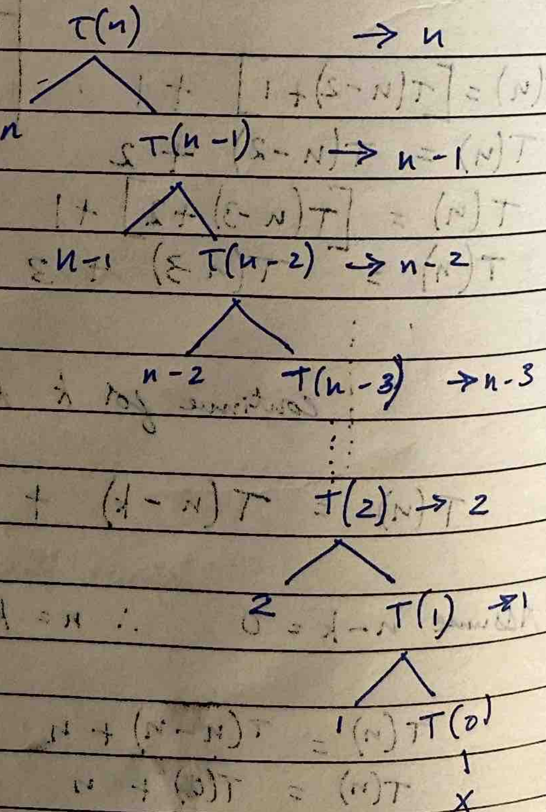
```
{
1 —   if (n > 0)
      {
n+1 —   for (i = 0; i < n; i++)
n   —       printf("%d", n);
T(n-1) —  Test(n-1);
      }
}
```

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

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

$$T(n) = \begin{cases} 0 & n = 0 \\ T(n-1) + n & n > 0 \end{cases}$$

Recursive Tree :



$$0 + 1 + 2 + \cdots + n-1 + n = \frac{n(n+1)}{2}$$

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

→ Back substitution method

$$T(n) = \begin{cases} 1 & n = 0 \\ T(n-1) + n & n > 0 \end{cases}$$

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

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

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

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

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

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

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

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

$$\vdots$$

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

assume $n - k = 0 \Rightarrow n = k$

$$T(n) = T(n-n) + (\cancel{n} - \cancel{n} + 1) + (\cancel{n} - \cancel{n} + 2) + \cdots (n-1) + n$$

$$T(n) = T(0) + 1 + 2 + 3 + \cdots + (n-1) + n$$

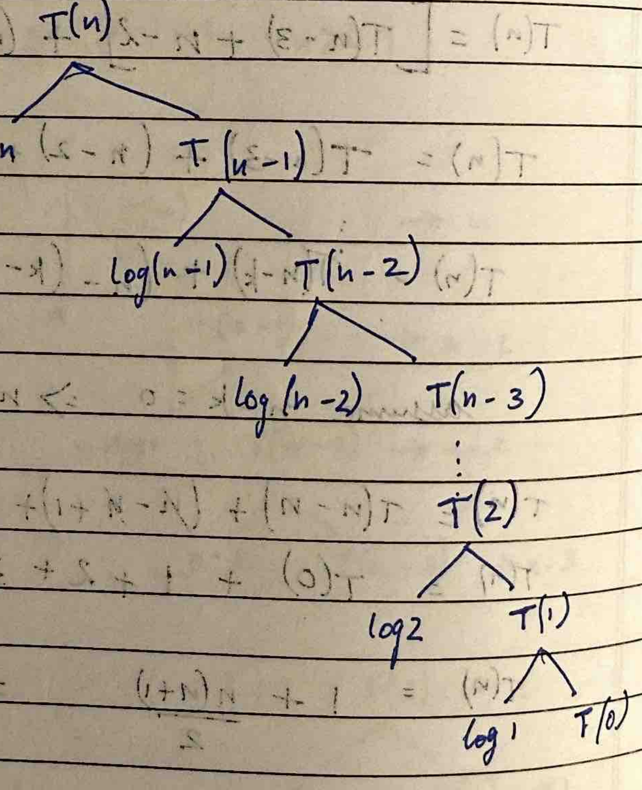
$$T(n) = 1 + \frac{n(n+1)}{2} = \Theta(n^2)$$

2.1.3 eg3

```
Void Test (int n)                      — T(n)
{
    if (n > 0)
    {
        for (i = 1; i < n; i = i*2)
            printf("%d", i);           — log n
        Test (n-1);                    — T(n-1)
    }
}
```

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

$$T(n) = \begin{cases} 1 & n = 0 \\ T(n-1) + \log n & n > 0 \end{cases}$$



$\log n + \log(n-1) + \dots + \log 2 + \log 1$

$= \log [n(n-1) \dots 2(1)]$

$= \log n!$

upper bound $\log n^n$ $\Rightarrow O(n \log n)$

→ Substitution Method / Induction Method

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

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

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

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

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

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

$n - k = 0$

$n = k$

$T(n) = T(0) + \log n!$

$O(n \log n)$

* $T(n) = T(n-1) + 1$ — $O(n)$
* $T(n) = T(n-1) + n$ — $O(n^2)$
* $T(n) = T(n-1) + \log n$ — $O(n \log n)$
* $T(n) = T(n-1) + n^2$ — $O(n^3)$

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

$T(n) = T(n-100) + n$ — $O(n^2)$

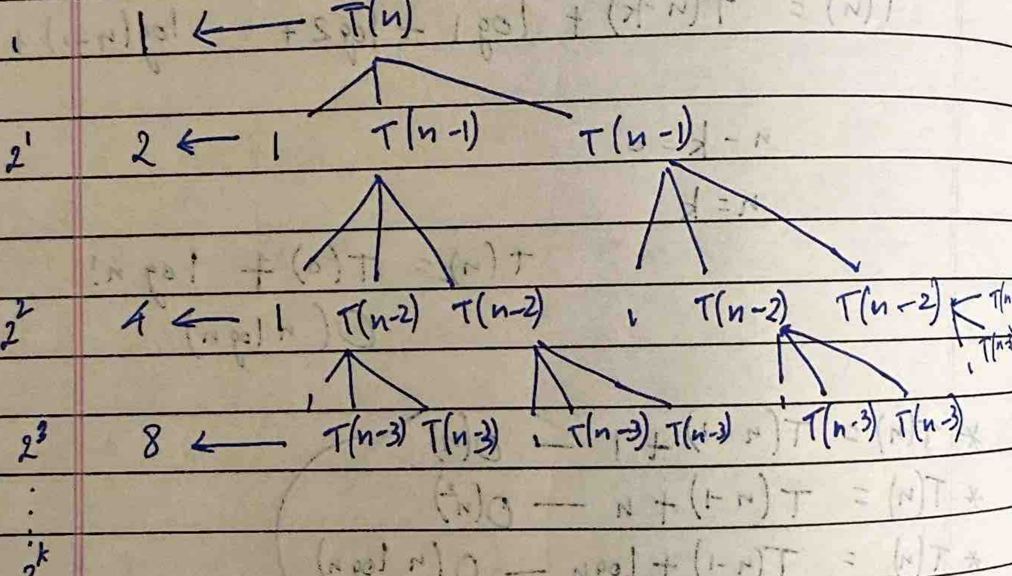
2.1.4

T(n) — Algo Test (int n)

```
{
    if (n>0)
    {
1           printf("%d", n);
T(n-1)      Test(n-1);
T(n-1)      Test(n-1);
    }
}
```

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

$$T(n) = \begin{cases} 1 & n = 0 \\ 2T(n-1) + 1 & n > 0 \end{cases}$$



GP $\Rightarrow 1 + 2^1 + 2^2 + 2^3 + \dots + 2^k = 2^{k+1} - 1$

$\frac{a(r^{k+1} - 1)}{r - 1}$

assume $n - k = 0 \Rightarrow n = k$

$2^{n+1} - 1$ $\quad O(2^n)$

Date
Page

→ Back substitution

$$T(n) = \begin{cases} 1 & n = 0 \\ 2T(n-1)+1 & n > 0 \end{cases}$$

$T(n) = 2T(n-1) + 1$ — ①

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

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

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

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

$\vdots$

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

assume $n-k=0 \Rightarrow n=k$

$= 2^n T(0) + 1 + 2 + 2^2 + \dots + 2^{k-1}$

$= 2^n \times 1 + 2^k - 1$

$= 2^n + 2^n - 1$

$= 2^{n+1} - 1$ $\quad O(2^n)$

$\theta(2^n)$

18th February, 2024

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

$T(n) = T(n-1) + n \longrightarrow O(n^2)$

$T(n) = T(n-1) + \log n \longrightarrow O(n \log n)$

$T(n) = 2T(n-1) + 1 \longrightarrow O(2^n)$

$T(n) = 3T(n-1) + 1 \longrightarrow O(3^n)$

$T(n) = 2T(n-1) + n \longrightarrow O(n2^n)$

2.2 # Master Theorem for Decreasing function

$$T(n) = a\,T(n-b) + f(n)$$

$a > 0,\ b > 0$ and $f(n) = O(n^k)$ where $k \geq 0$

$T(n) = T(n-1) + 1 \xrightarrow{\times n} O(n)$

$T(n) = T(n-1) + n \xrightarrow{\times n} O(n^2)$

$T(n) = T(n-1) + \log n \xrightarrow{\times n} O(n \log n)$

$T(n) = 2T(n-2) + 1 \rightarrow O(2^{n/2})$

Master's Theorem

- if $a = 1$: $O(n^{k+1})$, $O(n * f(n))$
- if $a > 1$: $O(n^k a^{n/b})$, $O(f(n)\, a^{n/b})$
- if $a < 1$: $O(n^k)$, $O(f(n))$

2.3.1 # Dividing Functions

Recursion Tree

$$T(n) = \begin{cases} 1 & n = 1 \\ T(n/2) + 1 & n > 1 \end{cases}$$

```
T(n)    — Algo Test (int n) {
              if (n>1) {
1       ————      printf("%d", n);
T(n/2)  ————      Test (n/2);
              }
          }
```

$T(n)$ — 1

$T(n/2)$ — 1

$T(n/2^2)$ — 1

$\vdots$

$T(n/2^k)$ — k

$n/2^k = 1$

k steps $\quad \therefore \frac{n}{2^k} = 1$

$$2^k = n$$

$$k = \log_2 n$$

$O(\log n)$

Substitution :

$T(n) = T(n/2) + 1 \qquad \Big| \qquad T(n/2) = T(n/2^2) + 1$

$$T(n) = [T(n/2^2) + 1] + 1$$

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

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

$$\vdots$$

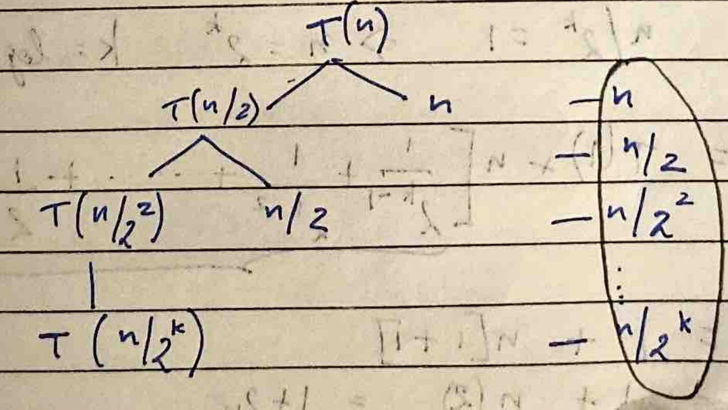
$$T(n) = T(n/2^k) + k$$

assume $\frac{n}{2^k} = 1 \Rightarrow n = 2^k \;,\; k = \log n$

$$T(n) = T(1) + \log n$$

$$O(\log n)$$

2.3.2 $\quad T(n) = \begin{cases} 1 & n = 1 \\ T(n/2) + n & n > 1 \end{cases}$



$$T(n) = n + \frac{n}{2} + \frac{n}{2^2} + \cdots + \frac{n}{2^k}$$

$$T(n) = n\left[1 + \frac{1}{2} + \frac{1}{2^2} + \cdots + \frac{1}{2^k}\right]$$

$$T(n) = n \sum_{i=0}^{k} \frac{1}{2^i}$$

$$= n(1)$$

$$T(n) = n$$

$$O(n)$$

Substitution: $T(n) = T(n/2) + n$ $\quad\Big|\quad T(n/2) = T(n/2^2) + n/2$

$$T(n) = \left[T(n/2^2) + \frac{n}{2}\right] + n$$

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

$$T(n) = T(n/2^3) + \frac{n}{2^2} + \frac{n}{2} + n$$

$$\vdots$$

$$T(n) = T(n/2^k) + \frac{n}{2^{k-1}} + \frac{n}{2^{k-2}} + \cdots + \frac{n}{2} + n$$

assume $n/2^k = 1 \Rightarrow n = 2^k \quad k = \log n$

$$T(n) = T(1) + n\left[\underbrace{\frac{1}{2^{k-1}} + \frac{1}{2^{k-2}} + \cdots + \frac{1}{2}}_{} + 1\right]$$

$$= 1 + n[1+1]$$

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

$$\Rightarrow O(n)$$

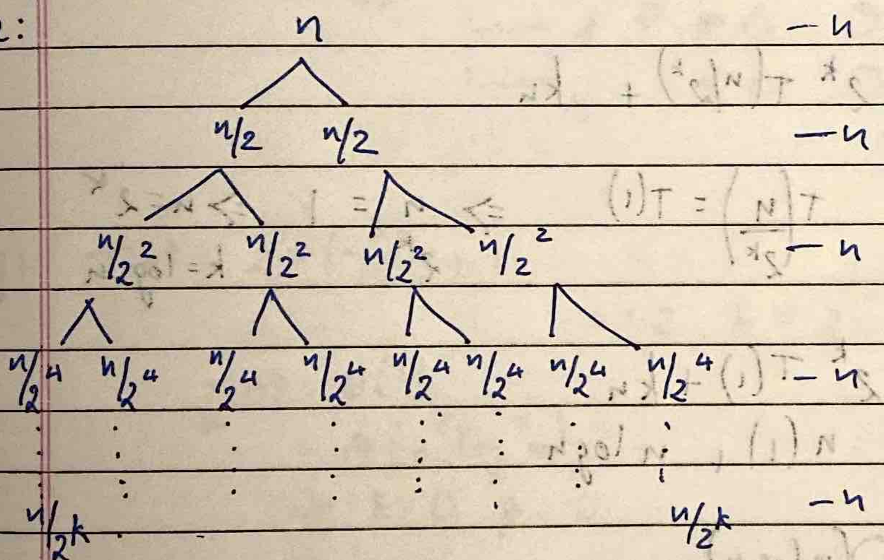
2.3.3 (*)

```
void Test(int n){                          —— T(n)
    if (n>1){
        for (i = 0; i<n; i++)   } —n
            stmt;
        Test(n/2);                         — T(n/2)
        Test(n/2);                         — T(n/2)
    }
}
```

$$T(n) = 2\,T(n/2) + n$$

$$T(n) = \begin{cases} 1 & n = 1 \\ 2\,T(n/2) + n & n > 1 \end{cases}$$

Tree:



$\Rightarrow n(k)$

assume $n/2^k = 1$

$k = \log n$

$\Rightarrow O(n \log n)$

Substitution $T(n) = \begin{cases} 1 & n = 1 \\ 2T(n/2) + n & n > 1 \end{cases}$

$T(n) = 2\,T(n/2) + n$

$= 2\left[2T\left(\frac{n}{2^2}\right) + \frac{n}{2}\right] + n$

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

$= 2^2\left[2T\left(\frac{n}{2^3}\right) + \frac{n}{2^2}\right] + 2n$

$T(n/2) = 2T\left(\frac{n}{2^2}\right) + \frac{n}{2}$

$T(n/2^2) = 2\,T(n/2^3) + \frac{n}{2^2}$

$T(n) = 2^3\,T(n/2^3) + 3n$.

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

$T\left(\frac{n}{2^k}\right) = T(1) \Rightarrow \frac{n}{2^k} = 1 \Rightarrow n = 2^k$

$k = \log n$

$T(n) = 2^k\,T(1) + kn$

$= n(1) + n\log n$

$O(n\log n)$

Date
Page

2.4.1 Masters Theorem for Dividing Functions

$$T(n) = a\,T(n/b) + f(n)$$

$a \geq 1$

$b > 1$

$$f(n) = \Theta(n^k \log^p n)$$

find ① $\log_b a$
② k

Case 1: if $\log_b a > k$ then $\Theta(n^{\log_b a})$

Case 2: if $\log_b a = k$
- if $p > -1$ $\Theta(n^k \log^{p+1} n)$
- if $p = -1$ $\Theta(n^k \log \log n)$
- if $p < -1$ $\Theta(n^k)$

Case 3: if $\log_b a < k$
- if $p \geq 0$ $\Theta(n^k \log^p n)$
- if $p < 0$ $O(n^k)$

eg 1. $T(n) = 2T(n/2) + 1$

$a = 2,\ b = 2$ $\quad \log_2 2 = 1$

$f(n) = \Theta(1)$
$= \Theta(n^0 \log^0 n)$
$\Rightarrow k = 0 \quad p = 0$

$\log_b a = \log_2 2 > 0$

$\Rightarrow \Theta(1)$

eg 2. $T(n) = 4T(n/2) + n$

$\log_2 4 = 2 > k = 1 \quad p = 0$

$\Theta(n^2)$

eg3. $T(n) = 8T(n/2) + n^1$

$\log_2 8 = 3 > k = 2 \quad p = 0$

$\Theta(n^3)$

eg4. $T(n) = 4T\left(\frac{n}{2}\right) + n$

$\Theta(n^2)$

eg5. $T(n) = 2T(n/2) + n$

$\log_2 2 = 1 \quad k = 1 \quad p = 0$

$\Theta(n \log n)$

eg6 $T(n) = 4T(n/2) + n^2 \log^5 n$

$\log_2 4 = 2 = k = 2$

$\Theta(n^2 \log^6 n)$

eg7. $T(n) = 8T(n/2) + n^3$

$\log_2 8 = 3 \quad k = 3$

$\Theta(n^3 \log n)$

eg8 $T(n) = 2T(n/2) + \frac{n^1}{\log n}$

$\log_2 2 = 1 \quad k = 1 \quad p = -1$

$\Theta(n \log\log n)$

eg 9. $T(n) = 2T(n/2) + n^1 \log^{-2} n$

$\log_2 2 = 1 \quad k = 1 \quad p = -2$

$\Theta(n)$

eg 10. $T(n) = T(n/2) + n^2$

$\log_2 1 = 0 < k = 2$

$\Theta(n^2)$

eg 11. $T(n) = 2T(n/2) + n^2 \log^2 n$

$\log_2 2 = 1 < k = 2$

$\Theta(n^2 \log^2 n)$

eg 12. $T(n) = 4T(n/2) + \frac{n^3}{\log n}$

$\log_2 2 = 2 < k = 3$

$\Theta(n^3)$

2.4.2

Case 1.

$T(n) = 2T(n/2) + 1 \longrightarrow \Theta(n^1)$

$T(n) = 4T(n/2) + 1 \longrightarrow \Theta(n^2)$

$T(n) = 4T(n/2) + n^1 \longrightarrow \Theta(n^2)$

$T(n) = 8T(n/2) + n^2 \longrightarrow \Theta(n^3)$

$T(n) = 16T(n/2) + n^2 \longrightarrow \Theta(n^4)$

Case 3.

$T(n) = T(n/2) + n^1 \longrightarrow \Theta(n)$

$T(n) = 2T(n/2) + n^2 \longrightarrow \Theta(n^2)$

$T(n) = 2T(n/2) + n^2 \log n \longrightarrow \Theta(n^2 \log n)$

$$T(n) = 4T(n/2) + n^3 \log^2 n \rightarrow \Theta(n^3 \log^2 n)$$
$$T(n) = 2T(n/2) + \frac{n^2}{\log n} \rightarrow \Theta(n^2)$$

Case 2.
$$T(n) = T(n/2) + 1 \rightarrow \Theta(\log n)$$
$$T(n) = 2T(n/2) + n^1 \rightarrow \Theta(n \log n)$$
$$T(n) = 2T(n/2) + n^1 \log n \rightarrow \Theta(n \log^2 n)$$
$$T(n) = 4T(n/2) + n^2 \rightarrow \Theta(n^2 \log n)$$
$$T(n) = 4T(n/2) + (n \log n)^2 \rightarrow \Theta(n^2 \log^3 n)$$
$$T(n) = 2T(n/2) + \frac{n^1}{\log n} \rightarrow \Theta(n \log \log n)$$
$$T(n) = 2T(n/2) + \frac{n}{\log^2 n} \rightarrow \Theta(n)$$

14th February, 2024

2.5 # Root function

```
Void Test (int n) {          — T(n)
    if (n > 2)
    {
        Stmt ;               — 1
        Test (√n) ;          — T(√n)
    }
}
```

$$T(n) = \begin{cases} 1 & n = 2 \\ T(\sqrt{n}) + 1 & n > 2 \end{cases}$$

$T(n) = T(\sqrt{n}) + 1$ | $a = 1 \quad \cancel{b = 1/2} \quad d = 0$

$T(n) = T(n^{1/2}) + 1$

$T(n) = T(n^{1/2^2}) + 2$

$T(n) = T(n^{1/2^3}) + 3$

$\vdots$

$$\sqrt{2} \times \frac{\sqrt{2}}{\sqrt{2}} = \frac{2}{\sqrt{2}} \Rightarrow \frac{n}{\sqrt{n}}$$

Date ______
Page ______

$$T(n) = T(n^{1/2^k}) + k$$

assume $n = 2^m$

$$T(2^m) = T(2^{m/2^k}) + k$$

assume $T(2^{\frac{m}{2^k}}) = T(2^1)$

$\therefore \frac{m}{2^k} = 1$

$m = 2^k$, $k = \log_2 m$

$\because n = 2^m \Rightarrow m = \log_2 n$

$\therefore k = \log_2 \log_2 n$

$\Theta(\log_2 \log_2 n)$

2.6.1 # Iterative Binary Search

```
int BinSearch (A[0...n-1], n, key)
{
    l = 0 , h = n-1
    while (l <= h)
    {
            mid = (l + h)/2;
            if (key == A[mid])
                       return mid;
            if (key < A[mid])
                       h = mid - 1;
            else
                       l = mid + 1;
    }
    return 0;
}
```

$C_{avg} \approx \log_2 n$

2.6.2 # Binary Search Recursive

```
Algo RBinSearch (l, h, key)                      — T(n)
{
  if (l == h)
  {
     if (A[l] == key)                            — 1
         return l;
     else
         return 0;
  }
  else {
     mid = (l+h)/2;                              — 1
     if (key == A[mid])                          — 1
         return mid;
     if (key < A[mid])                           — 1
         return (l, mid-1, key);     ┐
     else                            ├ T(n/2)
         return (mid+1, h, key);     ┘
  }
}
```

$$T(n) = \begin{cases} 1 & n = 1 \\ T(n/2) + 1 & n > 1 \end{cases}$$

$a = 1 \quad b = 2 \quad d = 0$

$a = b^d$

$1 = 2^0 = 1 \Rightarrow \Theta(\log n)$

2.7.1 # 2-way Merges Sort (iterative)

	1	2	3	4	5	6	7	8
A	9	3	7	5	6	4	8	2

8 elements

1st pass 39   59   46   28 — n elements merged

2nd pass 3 5 7 9   2 4 6 8 — n "

3rd pass 2 3 4 5 6 7 8 9 — n "

$8 \to 8/2 = 4 \to 4/2 \to 2 \to 2/2 = 1$

$\log_2 8 = 3$ → no of passes

no. of passes = $\log n$

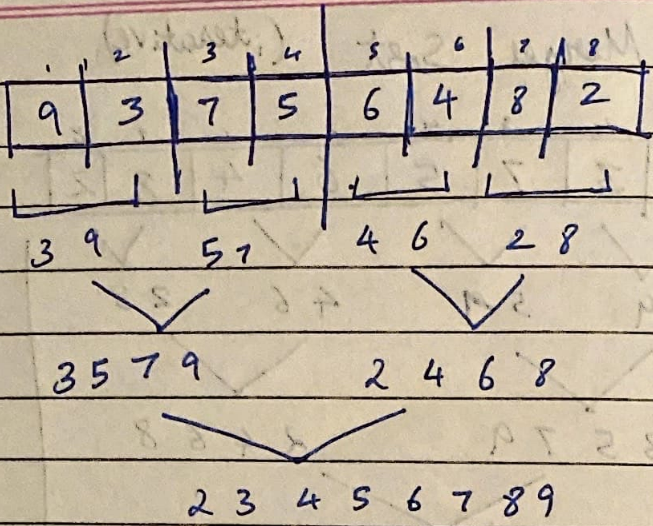
each pass n merges ⇒ $\Theta(n \log n)$

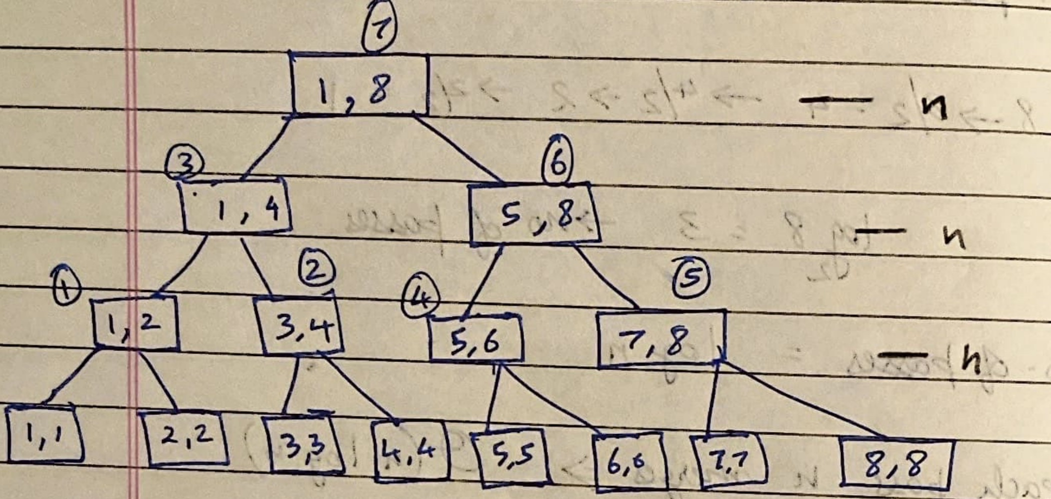
2.7.2 # MergeSort (recursive) (divide & conquer)

```
T(n)   - Algorithm MergeSort (l, h)
         {
             if (l < h)
             {
1      -         mid = (l+h)/2;
T(n/2) -         MergeSort (l, mid);
T(n/2) -         MergeSort (mid+1, h);
n      -         Merge (l, mid, h);
                        n elements
         }
```

size of A, size of B

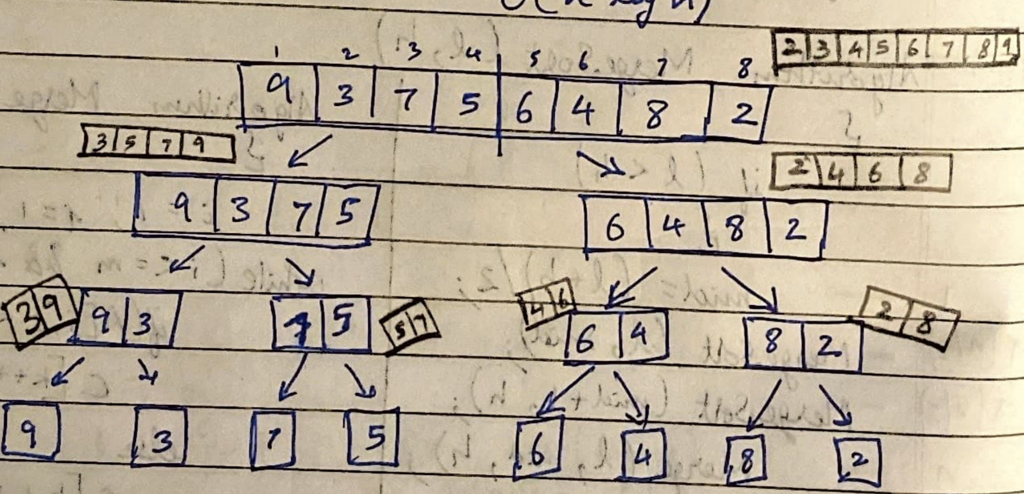
```
Algorithm Merge (A, B, m, n)
{
    i = 1, j = 1, k = 1;
    while (i <= m && j <= n) {
        if (A[i] < B[j])
            C[k++] = A[i++];
        else
            C[k++] = B[j++];
    }
    for (; i <= m; i++)
        C[k++] = A[i];
    for (; j <= n; j++)
        C[k++] = B[j];
}
```





n log n

$\Theta(n \log n)$



Recurrence relation : $T(n) = 2T(n/2) + n$

$$T(n) = \begin{cases} 1 & n = 1 \\ 2T(n/2) + n & n > 1 \end{cases}$$

$a = 2 \quad b = 2 \quad d = 1$

$a = b^d$ ✓ $\Rightarrow \Theta(n \cdot \log n)$

2.7.3 # Pros and Cons of Merge sort

Pros :
1. Large size list
2. Linked List
3. External sorting
4. Stable

Cons :
1. Extra space
2. No small problem
3. Recursive

21st February, 2024

2.8.1 # Quick sort

pivot → l … h

0	1	2	3	4	5	6	7	8	9
10	16	8	12	15	6	3	9	5	∞

0	1	2	3	4	5	6	7	8	9
6	5	8	9	3	10	15	12	16	∞

i $\quad$ i

sorted

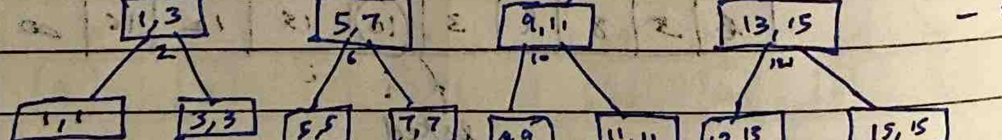
```
Partition (A, l, h)
{
  pivot = A[l];
  i = l; j = h;

while (i ≤ j)
 {
   while (A[i] ≤ pivot)
        i++;
   while (A[j] > pivot)
        j--;
   if (i < j)
        swap (A[i], A[j]);
 }
 swap (A[l], A[j]);      pivot swapping
 return j;
}
```

```
Quicksort (A, l, h)
{
   if (l < h)
   {
      j = partion (A, l, h);
      Quicksort (A, l, j);
      Quicksort (A, j+1, h);
   }
}
```

2.8.2 # Analysis

1 2 3 . . . . . 7 8 9 15

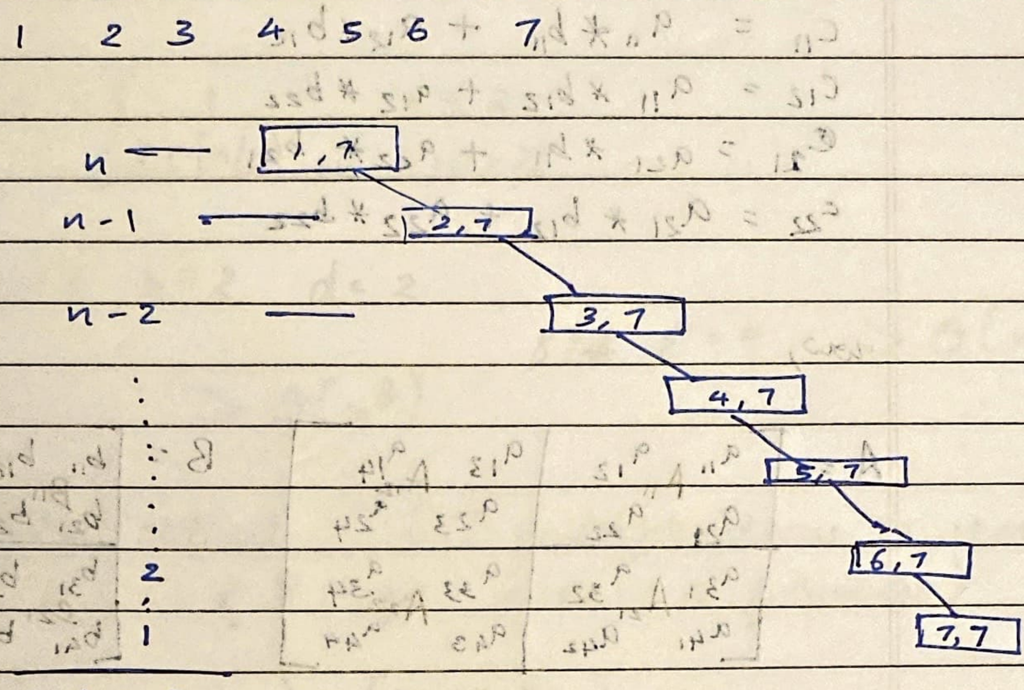
Best Case



$\frac{n}{2^k} = 1$

$k = \log_2 n$ $\Rightarrow$ Best case $= O(n \log n)$

Worst Case



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

$\Rightarrow$ Worst Case : $O(n^2)$

to improve worst case : 1. Select Middle element as pivot

2. Select Random element as pivot

2.9 # Strassens Matrix Multiplication

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \times B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = C = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}$$

```
for (i = 0; i<n; i++)
{
    for (j = 0; j < n; j++)
    {
        c[i, j] = 0;
        for (k = 0; k<n; k++)
            c[i][j] += A[i][j] * B[i][j];
    }
}
```

$O(n^3)$

$c_{11} = a_{11} * b_{11} + a_{12} * b_{12}$

$c_{12} = a_{11} * b_{12} + a_{12} * b_{22}$

$c_{21} = a_{21} * b_{11} + a_{22} * b_{21}$

$c_{22} = a_{21} * b_{12} + a_{22} * b_{22}$

now,

$$A = \left[\begin{array}{cc|cc} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ \hline a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{array}\right] \quad \begin{array}{cc} A_{11} & A_{12} \\ A_{21} & A_{22} \end{array}$$

$$B = \left[\begin{array}{cc|cc} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ \hline b_{31} & b_{32} & b_{33} & b_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{array}\right] \quad \begin{array}{cc} B_{11} & B_{12} \\ B_{21} & B_{22} \end{array}$$

```
Algo MM (A, B, n)
{
    if (n ≤ 2){
        c11 = a11 * b11 + a12 * b12;
        c12 = a11 * b12 + a12 * b22;      4 formulas
        c21 = a21 * b11 + a22 * b21;
        c22 = a21 * b12 + a22 * b22;
    }
    else {
        mid = n/2
                          matrix addn.
        MM(A11, B11, n/2) + MM(A12, B21, n/2);
        MM(A11, B12, n/2) + MM(A12, B22, n/2);
        MM(A21, B11, n/2) + MM(A22, B21, n/2);
        MM(A21, B12, n/2) + MM(A22, B22, n/2);
    }
}
```

Date ______
Page ______

$$T(n) = \begin{cases} 1 & n \leq 2 \\ 8T(n/2) + n^2 & n > 2 \end{cases}$$

$a = 8 \quad b = 2 \quad d = 2$

$8 > 2^2 = 4 \quad \Rightarrow \Theta(n^3)$

$\Rightarrow \Theta(n^3)$

→ Strassen's Method (7 multiplications instead of 8 for those 4 formulas)

$P = (A_{11} + A_{22}) \cdot (B_{11} + B_{22})$
$Q = (A_{21} + A_{22}) \cdot B_{11}$
$R = (A_{11}) \cdot (B_{12} - B_{22})$
$S = A_{22} \cdot (B_{21} - B_{11})$
$T = (A_{11} + A_{12}) \cdot B_{22}$
$U = (A_{21} - A_{11}) \cdot (B_{11} + B_{12})$
$V = (A_{12} - A_{22}) \cdot (B_{21} + B_{22})$

$C_{11} = P + S - T + V$
$C_{12} = R + T$
$C_{21} = Q + S$
$C_{22} = P + R - Q + U$

$$T(n) = \begin{cases} 1 & n \leq 2 \\ 7T(n/2) + n^2 & n > 2 \end{cases}$$

$a = 7 \quad b = 2 \quad d = 2$

$7 > 2^2$

$\Rightarrow O(n^{\log_2 7}) = O(n^{2.81})$