Lecture: 1

20 Feb

Algo: Sequence of steps to solve a problem that will take input and will produce at least one output.

Representation:

Analyse
- Pseudo code (Generic, abstract)
- flowchart (Symbolic, execution/flow)
- Program (concrete rep language machine understanding)

1: **Accuracy** (Desired) in terms of testing / Blackbox white box

Given input	Know output	Software output
0	1	0 x
5	120	120 ✓

Test case ⇒ Requirements

unit testing: check one module

integration:

f(int, int)

f(float, float)

2: **Efficiency:**

→ time

→ space

→ empirical analysis

→ Analytical analysis

Add first 100 natural numbers:

Algo 1:

```
Sum = 0          S1
for(i=1; i<100; i++)   S2 S3 S4
    Sum += i;    S5
display Sum;     S6
```

$S_1 + S_2 + S_3 + S_4 + S_5 + S_6$

$1 + 1 + 100 + 100 + 100 + 1$

$1 + 1 + 1 + 2 + 2 + 1$

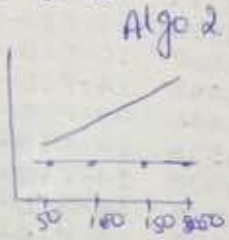
$1 + 1 + 100 + 200 + 200 + 1$

$503 \Rightarrow$ B.op

Algo 2: $\Rightarrow$ constant complexity

```
Display 100*(100+1)/2
        S1  S2  S3 S4
```

= 4 B.op



Empirical Analysis:

Implement (Time, cost)

Differences in terms of hardware (Processor, RAM, I/O)

" " " " Software (OS, tools (V. Studio, turbo))

" " " - Language (Java, C++, Assembly) + Programming style (loop, structure)

" " " " network (Bandwidth)

" " " System state

Analytical Analysis:

Unit of time in terms of Basic operations:

Sequential:

```
Sum ← 0;  => 1
Sum = Sum + 10; => 2
Sum = Sum * 2; => 2
display Sum; => 1
        = 6 B.op
```

Conditional: Best case

```
a = 2;  => 1
if (a > 10) => 1/2 Bop
{
  Sum ← 0;
  Sum = Sum + 10;
  Sum = Sum * 2;
}
display Sum;
```

```
a = 20 => 1
if (a > 10) => 1
{
  Sum ← 0; => 1
  Sum = Sum + 10; => 2
  Sum = Sum * 2; => 2
  display Sum; => 1
}
Sum = 10; => 1
display Sum; => 1
```

worst case: 1 + 1 + 6

Best case: 1 + 1 + 2

→ if-else is more powerful than switch statement.

→ Range is given in if-else

Loop:

```
for(i = 1 to n)
  display Ali;
```

```
for (int i=1; i<=n; i++)
    cout << "Ali";
```

$= 1 + n + 2n + n$

$= 1 + 4n$

$\Rightarrow 4n$

$O(n)$

$\sum_{low}^{up} c$

$(up - low + 1)$

$1 + \sum_{i=1}^{n} (1 + 2 + 1)$ (condition → 1+2+1 → Body of loop; increment)

$= 1 + \sum_{i=1}^{n} (4)$

$= 1 + 4 \sum_{i=1}^{n} (1)$

$= 1 + 4(n - 1 + 1)$

$= 1 + 4n$

=>2

2

=>1

1

lement.

5) for (int i=1; i<=n; i++)
  for (int j=i; j<=n; j++)
    for (int k=j; k<=n; k++)
      Print "Ali"

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

$= 1 + \sum_{i=1}^{n} \left(1+2+1+\sum_{j=i}^{n}\left(1+2+1+\sum_{k=j}^{n}(1+2+1)\right)\right)$

$= 1 + \sum_{i=1}^{n} \left(4+\sum_{j=i}^{n}\left(4+\sum_{k=j}^{n}(4)\right)\right)$

$= 1 + \sum_{i=1}^{n} \left(4+\sum_{j=i}^{n}(4+4(n-j+1))\right)$

$= 1 + \sum_{i=1}^{n} \left(4+\sum_{j=i}^{n}(4+4n-4j+4)\right)$

$= 1 + \sum_{i=1}^{n} \left(4+\sum_{j=i}^{n}(8+4n-4j)\right)$

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

$= 1 + \sum_{i=1}^{n} \left(4+\sum_{j=i}^{n} 8 + \sum_{j=i}^{n} 4n - \sum_{j=i}^{n} 4j\right)$

$= 1 + \sum_{i=1}^{n} \left(4+8(n-i+1)+4n(n-i+1)-4\sum_{j=i}^{n} j\right)$

$= 1 + \sum_{i=1}^{n} \left(4+8n-8i+8+4n^2-4in+4n-4\left(\sum_{j=1}^{n} j - \sum_{j=1}^{i} j\right)\right)$

$= 1 + \sum_{i=1}^{n} \left(4+8n-8i+8+4n^2-4ni+4n-4\left(\frac{n(n+1)}{2}-\frac{i(i+1)}{2}\right)\right)$

$= 1 + \sum_{i=1}^{n} \left(4+8n-8i+8+4n^2-4ni+4n-2\left(n(n+1)-i(i+1)\right)\right)$

$= 1 + \sum_{i=1}^{n} \left(4+8n-8i+8+4n^2-4ni+4n-2n^2-2n+2i^2+2i\right)$

$= 1 + \sum_{i=1}^{n} \left(12+10n-6i+2n^2-4ni+2i^2\right)$

$= 1 + \sum_{i=1}^{n} 12 + \sum_{i=1}^{n} 10n - \sum_{i=1}^{n} 6i + \sum_{i=1}^{n} 2n^2 - \sum_{i=1}^{n} 4ni + \sum_{i=1}^{n} 2i^2$

$= 1 + 12(n-1+1) + 10n(n-1+1) - 6\sum_{i=1}^{n} i + 2n^2(n-1+1) - 4n\sum_{i=1}^{n} i + 2\sum_{i=1}^{n} i^2$

$= 1 + 12n + 10n^2 - 3\left(\frac{n(n+1)}{2}\right) + 2n^3 - 2n\left(n(n+1)\right) + 2\left(\frac{n(n+1)(2n+1)}{6}\right)$

$= 1 + 12n + 10n^2 - 3n^2 - 3n + 2n^3 - 2n^3 - 2n^2 + \frac{(n^2+n)(2n+1)}{3}$

$= 1 + 12n + 10n^2 - 3n^2 - 3n + 2n^3 - 2n^3 - 2n^2 + \frac{(2n^3+n^2+2n^2+n)}{3}$

Lecture: 4

13 Mar

1:
```
for (int i=1; i<=n; i++)        // n
    Print "Ali";
```
$= O(n)$

2: nested loop:
```
for (int i=1; i<=n; i++)        // n
    for (int j=1; j<=n; j++)    // n
        Print "Ali";
```
$= O(n^2)$

3:
```
for (int i=1; i<=n; i++)        // n
    for (int j=1; j<=i; j++)    // dependant i
        Print "Ali";
```

i =	1	2	3	----	n
j =	1 time	2 times	3 times	----	n times

$= 1 + 2 + 3 + \dots + n$

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

4:
```
for (int i=1; i<=n; i++)        // n
    for (int j=i; j<=n; j++)    // Dependent i
        Print "Ali";
```

i =	1	2	3	----	n
j =	1 - n times	2 - n' times	3 - n times	----	n - n 1 time

$= n + n - 2 + (n-3) + \dots + 3 + 2 + 1$

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

5:

```
for(int i=1; i*i<=n; i++)        → i<=√n → (√n)
    Print "Ali";
```

$= O(\sqrt{n})$

6:

```
for(int i=1; i<=n; i++)          → n
    for(int j=1; j<=i; j++)      → Dependent i when j<=n ⇒ 100n²
        for(k=1; k<=100; k++)    → 100
            Print "Ali"
```

i = 1 2 3 ---- n

j = 1 time 2times 3times ---- n times

k = 1×100 2×100 3×100 ---- 3n×100

$1\times100 + 2\times100 + 3\times100 + \dots + n\times100$

$= 100(1+2+3+\dots+n)$

$= 100\left(\frac{n(n+1)}{2}\right)$

$= O(n^2)$

7:

```
for(int i=1; i<=n; i++)          → (n) Dependent i
    for(int j=1; j<=i²; j++)     → n/2
        for(int k=1; k<=n/2; k++)  => will alway execute n/2
            Print "Ali";
```

i = 1 2 3 ------ n

j = 1 time 4 times 9 times ------ n^2 times

k = $\frac{1}{2}^{xn}$ $\frac{2}{2} \Rightarrow 1$ $\frac{3}{2}$ ---------- $\frac{n}{2}$

$\frac{1}{2}$ $\frac{8}{2}$ $\frac{27}{2}$ $\frac{n^3}{2}$

$= \frac{1}{2} + \frac{8}{2} + \frac{27}{2} + \dots + \frac{n3}{2}$

8:

```
for(
```

9:

```
for(
```

i = 1

= 2

$= \frac{1}{2}(1 + 8 + 27 + \cdots + n^3)$

$= O(n^3)$

$= \frac{1}{2}(1 + 2 + 3 + \cdots + n)$

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

$= O(n^2)$

$i = 1 \quad 2 \quad 3 \quad \cdots \quad n$

$j = 1^2 \text{ times} \quad 2^2 \text{ times} \quad 3^2 \text{ times} \quad \cdots \quad n^2 \text{ times}$

$k = 1^2 \times n/2 \quad 2^2 \times n/2 \quad 3^2 \times n/2 \quad \cdots \quad n^2 \times n/2$

$= 1 \times n/2 + 2^2 \times n/2 + 3^2 \times n/2 + \cdots + n^2 \times n/2$

$= \frac{n}{2}\left(1 + (2)^2 + (3)^2 + \cdots + (n)^2\right)$

$= \frac{n}{2}\left(\frac{n(n+1)(2n+1)}{6}\right)$

$= O(n^4)$

8:

```
for(int i=n/2; i<=n; i++)          → n/2
    for(int j=1; j<=n/2; j++)      → n/2
        for(int k=1; k<=n; k++)    → n
            Print "Ali"
```

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

$= O(n^3)$

9:

```
for(int i=1; i<=n; i=i*2)
    Print "Ali"
```

$i = 1 \quad 2 \quad 4 \quad 8 \quad 16 \quad \cdots \quad n$

$= 2^0 \quad 2^1 \quad 2^2 \quad 2^3 \quad 2^4 \quad \cdots \quad 2^k$

$2^k = n$

$\log 2^k = \log n$

$K \log 2 = \log n$ $\quad \because \boxed{\log 2 = 1}$

$K = \log_2 n$

when i = i x 3:

i = 1 3 9 27 ..... n

$= 3^0 \quad 3^1 \quad 3^2 \quad 3^3 \ldots\ldots 3^K$

$3^K = n$

$\log(3)^K = \log n$

$K \log 3 = \log n$

$K = \log_3 n$

when i = i x j:

$\log_j(n)$

10:

```
for (int i = n/2 ; i<=n; i++)        → n/2
    for (int j = 1; j<= n/2; j++)    → n/2
        for (K = 1; K<= n; K = K*2)  → log2 n
                => Print "Ali";
```

$= \frac{n}{2} \times \frac{n}{2} \times \log_2 n$

$= n^2 \log_2 n$

11:

```
for (int i = n/2 ; mi<=n; i++)       → n/2
    for (j = 1; j<= n; j = j*2)      → log2 n
        for (int K = 1; K<=n; K = K*2) → log2 n
            Print "Ali"
```

$= \frac{n}{2} \times \log_2 n \times \log_2 n$

$= n(\log_2 n)^2$

12:

```
for (int i =
    for (
```

i = 1

j = 1 - n

(1, 2, 3, ... n) (

n times

$= n + \frac{n}{2} +$

$= n(1 +$

$= n \log$

13:

```
A( )
{
  i=1
wh
```

i = 1

S = 1

14:

```
A( )
{
  int n= 2^k
    for(int i=1; i<=n; i++)
      {
      int j=2;
        while (j<=n)
          { j= j^2;
          }
        Print "Ali";
      }
}
```

K = 1	2	3	K
$n = 4 = 2^2$	$n = 16 = 4^2$	$n = 2^8$	$n * K + 1$ times
$j = 2, 4$	$j = 2, 4, 16$	$j = 2, 2^2, 2^4, 2^8$	↓
$n * 2$ times	$n * 3$ times	$n * 4$ times	compute

$$n = 2^{2^k}$$

$$\log n = 2^k$$

$$k = \log(\log(n))$$

$$\boxed{n * K + 1}$$

$$= O(n * \log(\log n))$$

Two methods — By Basic definition
— By applying limits

By Applying limits:

$$\lim_{n\to\infty} \frac{f_n}{g_n} = v$$

$$0 \le v < \infty$$

$$f_n = O_g(n)$$

e.g:

$$\lim_{n\to\infty} \frac{10n^2+5n}{n^2}$$

$$= \lim_{n\to\infty} \left(\frac{10n^2}{n^2} + \frac{5n}{n^2}\right)$$

$$= \lim_{n\to\infty} 10 + \lim_{n\to\infty} \frac{5}{n}$$

$$= 10 + \frac{5}{\infty}$$

$$= 10$$

$$\Rightarrow 10n^2+5n = O(n^2)$$

e.g:

$$\lim_{n\to\infty} \frac{(10n^2+5n)}{n}$$

$$= \lim_{n\to\infty} 10n^2+5n$$

$$= 10(\infty) + 5$$

$$= 10\infty + 5$$

$$= \infty + 5 = \infty$$

$$10n^2+5n \neq O(n)$$

Lecture: 5

15 Mar

$\log_2 n = \frac{\ln n}{\ln 2}$

for large value of n

$\lg n < \sqrt{n} < n < n\lg n < n^2 < n^3 < 2^n < n! < n^n$

Asymptotic Analysis:

How we can map complexity of Algorithm.

O - Notation:

(i) $f_n \Rightarrow$ Complexity of your algorithm

$10n^2 + 5n$

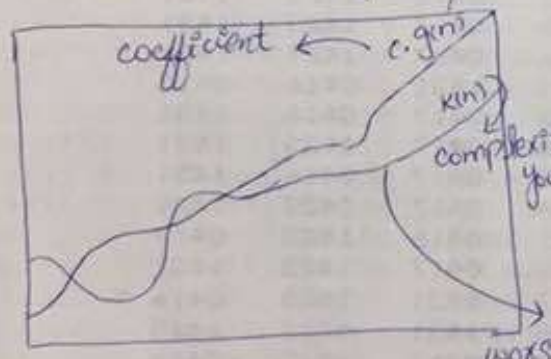
(ii) $g_n \Rightarrow$ Standard functions

$\lg n, \sqrt{n}, n^2, \ldots$

(iii) Co-efficient C

$c \cdot g(n)$

(iv) $n \geq n_0$



worst case complexity of your Algo do not exceed the standard function

$10n^2 + 5n \leq 13n^2$

$n = 1$

$15 \leq 13$ ×

$n = 2$

$50 \leq 52$ ✓

$10n^2 + 5n \leq 11n^2$

$n = 1$

$15 \leq 11$ ×

$n = 2$

$50 \leq 44$ ×

$n = 3$

$105 \leq 99$ ×

$n = 4$

$180 \leq 176$ ×

$n = 5$

$275 \leq 275$ ✓ $n \geq 5$

$10n^2 + 5n = O(n^2)$

$f(n) \leq cg(n), n \geq n_0$

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

$n \geq 2$ ↓ upper bond

e.g: $\lim_{n\to\infty} \frac{10n^2+5n}{n^3}$

$= \lim_{n\to\infty} \left(\frac{10n^2}{n^3} + \frac{5n}{n^3}\right)$

$= \lim_{n\to\infty} \left(\frac{10}{n} + \frac{5}{n^2}\right)$

$= \lim_{n\to\infty} \frac{10}{n} + \lim_{n\to\infty} \frac{5}{n^2}$

$= 0 \Rightarrow 10n^2+5n = O(n^3)$

By Basic Definition:

we show that

e.g $3n^2+10n = O(n^2)$

Consider:

$10 \le n$

$f(n)$ $\quad$ $g(n)$

xing by n

$10n \le n^2$

Add $3n^2$

$3n^2 + 10n \le 3n^2 + n^2$

$= 4n^2$

$3n^2 + 10n \le c \cdot n^2$

e.g: $n \le n^2 \Rightarrow$ xing by 10 / $n \ge 1$

$10n \le 10n^2$

Adding $3n^2$

$3n^2 + 10n \le 3n^2 + 10n^2$

$3n^2 + 10n \le 13n^2$ → for $n \ge n_0$ where $c = 13, n_0 = 1$

$3n^2 + 10n \le c \cdot n^2$

by Basic definition:

$3n^2 + 10n = O(n^2)$

$$= \lim_{n\to\infty} \frac{\frac{1}{n}(\ln 2)}{\ln 2 \cdot 2n}$$

$$= \frac{\ln 2}{n} \times \frac{1}{\ln 2 \cdot 2n}$$

$$= \frac{1}{n \cdot 2n}$$

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

$$= \frac{1}{\infty}$$

$$= 0 + 1 + 0$$

$$= 1$$

$\Rightarrow \lg(n) + n^2 + n = O(n^2)$

$\lg(n) + n^2 + n = \Omega(n^2)$

$\lg(n) + n^2 + n = \Theta(n^2)$

$n \lg n = O(n^2)$

$$= \lim_{n\to\infty} \frac{n \lg n}{n^2}$$

$$= \lim_{n\to\infty} \frac{\lg n}{n} = \frac{\infty}{\infty} \text{ (undefined)}$$

$\because \lg n = \frac{\ln n}{\ln 2}$

$$= \lim_{n\to\infty} \frac{\ln n / \ln 2}{n}$$

$$= \lim_{n\to\infty} \frac{\ln n}{\ln 2} \times \frac{1}{n}$$

$$= \lim_{n\to\infty} \frac{d/dn \ln n}{d/dn (\ln 2 \cdot n)}$$

$$= \lim_{n\to\infty} \frac{1/n \cdot \ln 2}{\ln 2 (1)}$$

$$= \lim_{n\to\infty} \frac{\ln 2}{n} \cdot \frac{1}{\ln 2}$$

$$= \frac{1}{\infty} = 0$$

$\Rightarrow n \lg(n) = O(n^2)$

$n \lg(n) \neq \Omega(n^2)$

$n \lg(n) \neq \Theta(n^2)$

Algo.

$$= \lim_{n\to\infty} \frac{\ln n}{n} \cdot \frac{1}{\ln n}$$

$$= \frac{1}{\infty} = 0$$

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

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

$$= \lim_{n\to\infty} \frac{\frac{n^2}{2} + \frac{n}{2}}{n^2}$$

$$= \lim_{n\to\infty} \frac{\frac{n^2}{2}}{n^2} + \frac{\frac{n}{2}}{n^2}$$

$$= \lim_{n\to\infty} \frac{n^2}{2} \cdot \frac{1}{n^2} + \lim_{n\to\infty} \frac{n}{2} \cdot \frac{1}{n^2}$$

$$= \frac{1}{2} + \frac{1}{2\cdot\infty}$$

$$= \frac{1}{2} + \frac{1}{\infty}$$

$$= \frac{1}{2} + 0$$

$$= \frac{1}{2} \Rightarrow (0 \le c < \infty) \Rightarrow (0 < c \le \infty) \Rightarrow (0 < c < \infty)$$

$$\frac{n(n+1)}{2} = O(n^2) \qquad \frac{n(n+1)}{2} = \Omega(n^2) \qquad \frac{n(n+1)}{2} = \Theta(n^2)$$

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

$$= \lim_{n\to\infty} \frac{n^2}{2^n}$$

$$= \lim_{n\to\infty} \frac{\infty}{\infty} \text{ undefined}$$

By applying L. Hopital Rule

$$= \lim_{n\to\infty} \frac{\frac{d}{dn}(n^2)}{\frac{d}{dn}(2^n)}$$

$= \lim_{n\to\infty} \frac{2n}{2^n \cdot \ln 2}$

$= \frac{\infty}{\infty}$ (undefined)

By applying derivative

$= \lim_{n\to\infty} \frac{2n}{2^n \cdot \ln 2} \Rightarrow \lim_{n\to\infty} \frac{2}{2^n \cdot (\ln 2)^2}$ → constant value

$\lim_{n\to\infty} \frac{2}{2^n \cdot \ln 2 \cdot \ln 2}$

$= \frac{2}{2^\infty}$

$= \frac{2}{\infty}$

$= 0$

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

$n^2 \neq \Omega(n^2)$

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

$\lg(n) + n^2 + n = \Theta(n^2)$

$= \lim_{n\to\infty} \frac{\lg(n)}{n^2} + \frac{n^2}{n^2} + \frac{n}{n^2}$

$= \lim_{n\to\infty} \frac{\lg(n)}{n^2} + \lim_{n\to\infty} \frac{n^2}{n^2} + \lim_{n\to\infty} \frac{n}{n^2}$

$= \frac{\infty}{\infty} + 1 + \frac{1}{\infty}$

$= 1 + 0$

$= 1$

$= \lim_{n\to\infty} \frac{\lg(n)}{n^2} \rightarrow \lim_{n\to\infty} \frac{\ln n}{\ln 2}$

$= \lim_{n\to\infty} \frac{\frac{d}{dn}(\lg(n))}{\frac{d}{dn}(n^2)}$ $= \lim_{n\to\infty} \frac{\ln n / \ln 2}{n^2/1}$

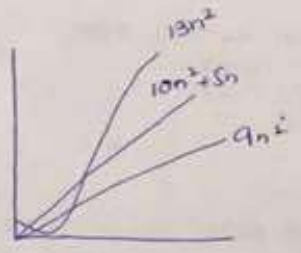
$= \lim_{n\to\infty} \frac{1}{2n}$ $= \lim_{n\to\infty} \frac{\ln n}{\ln 2 \cdot n^2}$

$= \lim_{n\to\infty} \frac{\frac{d}{dn}(\ln n)}{\frac{d}{dn}(\ln 2 \cdot n^2)}$

Ω (Omega Notation): Lecture: 6 20 Mar

Standard function is below your Alg.

We talk about best case.



$0 \le c \le \infty$
$f(n) = O(g(n))$
$0 < c \le \infty$
$f(n) = \Omega(g(n))$
$0 < c < \infty$
$f(n) = \Theta(g(n))$

$10n^2+5n = O(13n^2)$

$10n^2+5n = \Omega(9n^2)$

$\lg(n) = O(n)$

$$\lim_{n\to\infty} \frac{\log(n)}{n} = \frac{\infty}{\infty} \text{ (undefined)}$$

L. Hopital Rule

Since $\lg(n) = \frac{\ln n}{\ln 2}$

By applying L. Hopital Rule

$$\lim_{n\to\infty} \frac{\ln n / \ln 2}{n/1}$$

$$\lim_{n\to\infty} \frac{\ln n \cdot 1}{\ln 2 \cdot n}$$

$$\lim_{n\to\infty} \frac{\frac{d}{dn}(\ln n)}{\frac{d}{dn}(\ln 2 \cdot n)}$$

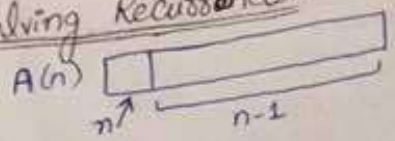
$$\lim_{n\to\infty} \frac{\frac{1}{n}(\ln 2)}{\frac{\ln 2}{1} \cdot \frac{dn}{dn}}$$

$$\lim_{n\to\infty} \frac{\frac{\ln 2}{n}}{\frac{\ln 2}{1}}$$

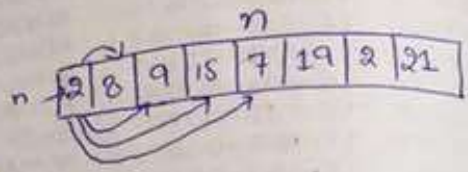
29 Mar

Lecture: 8

Solving Recurrences:



$T(0) = 0$
$T(n) = T(n-1) + c$ } Decrease & Conquer



$T(n) = T(n-1) + nc$ } Decrease & conquer
$T(0) = 0$

$T(n) = T(n-1) + c$
$T(n-1) = T(n-2) + c$
$T(n-2) = T(n-3) + c$
$T(n-3) = T(n-4) + c$
⋮

$T(3) = T(2) + c$
$T(2) = T(1) + c$
$T(1) = T(0) + c$

$T(n) = 0 + c + c + c + \cdots + c$

$T(n) = nc$

$T(n) = O(n)$

$\Rightarrow 2^n$

```
f(n)
{
if (n==1)
  return 2;
else
  return 2*f(n-1);
}
```

2^n
$\Rightarrow 2^4$
$2 \cdot 2^3$
$\downarrow$
$2 \cdot 2^2$
$\downarrow$
$2 \cdot 2^1$
$\downarrow$
2^1

$T(n) = T(n-1) + 1$ → one operation

Merge Sort: When value of start is less than value of End.

```
ms(A,S,E)
{
  if (S<E)
    mid = (S+E)/2;
    ms(A,S,m)
    ms(A,mid+1,E)
    Merge(A,S,m,E)
}
```


```
Merge(A,S,m,E)
{
  B[S..E]
  i=K=S; j=mid+1;
  while (i<=mid && j<=E)
  {
    if (A[i]<A[j])
      B[K++] = A[i++];
    else
      B[K++] = A[j++]
  while (i<=mid)
  B[K++] = A[i++];
  while (j<=E)
    B[K++] = A[i++];
  for (i=S to E)
    A[i] = B[i];
```

Divide & conquer:

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

↓ for 1st half ↓ for 2nd half ↓ merge

$T(n) = 2T(n/2) + Cn$ ← Cost of selection (Div+merg)

$T(1) = C$ — No. of subproblem

Size of subproblem

Lecture: 7 Recursive call: 22 Nov

Binary Search:

Array should be in sorted form.

A	9	11	15	20	25	30	35	60	*
	0	1	2	3	4	5	6	7	8
S=									E=

```
B.S(A, S, E, n) if(S<=E)
mid = (S+E)/2
  if (n == A[mid])
      Print mid
else if (n < A[mid])
      B.S(A, S, mid-1, n)
else
      B.S(A, mid+1, E, n)
```

(0+8)/2 = 4

n == A[4]

n = A[mid]

n > A[mid]

n < A[mid]

Divide & conquer:

~~Merge Sort~~ $T(n) = T(n/2) + c$

(input size)

$T(1) = c;$

Factorial:

```
f(n)
{
  if(n==1)
    return 1;
  else
    return n*f(n-1);
}
```

⇒ f(5)

(5==1)

⇒ NO

5 * f(5-1)

⇒ 5 * f(4)

⇒ 4 * f(3)

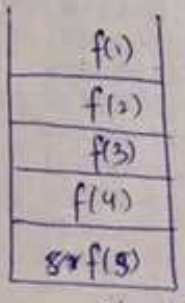
⇒ 3 * f(2)

⇒ 2 * f(1)

return 1;

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

$T(1) = 1$



Stack

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

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

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

$\vdots$

$T(3) = T(2) + 3c$

$T(2) = T(1) + 2c$

$T(1) = T(0) + 1c$

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

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

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

$\Rightarrow \Theta(n^2)$

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

$\frac{n(n+1)}{2n^2} \Rightarrow \lim_{n\to\infty} \frac{n(n+1)}{2n^2}$

$= \frac{1}{2}\left(\frac{n}{2n^2} \cdot \frac{(n+1)}{n^2}\right) \overset{\lim}{=} \frac{n+1}{2n}$

$\lim_{n\to\infty} \left(\frac{n}{2n} + \frac{1}{2n}\right)$

$= \lim_{n\to\infty} \left(\frac{1}{2} + \frac{1}{2n}\right)$

$= \lim_{n\to\infty} \frac{1}{2} + \lim_{n\to\infty} \frac{1}{2n}$

$= \frac{1}{2} + \frac{1}{2\infty}$

$= \frac{1}{2} + \frac{1}{\infty}$

$= \frac{1}{2} + 0$

$= \frac{1}{2}$

$\Rightarrow \frac{n(n+1)}{2} = \overset{0}{\Theta}(n^2)$

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

$\frac{n(n+1)}{2} \neq \Omega(n^2)$

g 2 = 1

12:

```
for(int i=1; i<=n; i++)        → n
    for(int j=1 ; j<=n; j=j*i)
        Print "Ali";
```

i =	1	2	3	-----	n
j =	1-n	1-n	1-n	----	n-n
	(1,2,3,...n)	(1,4,6,...n)	(1,6,9,...n)	....	n-n
	n times	n/2 times	n/3 times		1 time

$$= n + \frac{n}{2} + \frac{n}{3} + \cdots + 1$$

$$= n\left(1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}\right)$$

$$= n \log n$$

13:

```
A( )
{
  i=1 ; S<=i ;
  while (S<=n)
      {
        i++;
        S= S+i;
          Print "Ali";
      }
}
```

i =	1	2	3	4 -----	K
S =	1	3	6	10 ----	n

$n = 1 - K$

$n = \frac{K(K+1)}{2} \Rightarrow \frac{K^2}{2} + \frac{K}{2}$ Ignore

$n = K^2$

$K = \sqrt{n}$

$= O(\sqrt{n})$

Lecture: 9

3 Apr

$T(0) = 0$

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

$T(n) = T(n-1) + c\log n$ $\qquad \log(a \cdot b) = \log a + \log b$

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

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

$\vdots \qquad \vdots$

$T(3) = T(2) + c\log(3)$

$T(2) = T(1) + c\log(2)$

$T(1) = T(0) + c\log(1)$

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

$T(n) = c\log(n * n-1 * n-2 * \cdots * 3 * 2 * 1)$

~~$T(n) = c\log \frac{n(n+1)}{2}$~~ $\qquad T(n) = c\log(n!)$

$T(1) = c$

$T(n) = 2T(n/2) + c, \quad n > 1$

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

$T(n/4) = 2T(n/8) + c$

$T(n) = 2T(n/2) + c \quad \Rightarrow \quad 2T(n/2^1) + c \quad (2^0)c$

$T(n) = 2\left[2T(n/4) + c\right] + c$

$= 4T(n/4) + 3c \quad \Rightarrow 2^2 T(n/2^2) + 3c \quad (2^0 + 2^1)c$

$T(n) = 4\left[2T(n/8) + c\right] + 3c$

Complexity of Merge Sort:

$T(1) = C$

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

$T(n/2) = 2T(n/4) + c\,n/2$

$T(n/4) = 2T(n/8) + c(n/4)$

$T(n) = 2T(n/2) + cn \longrightarrow 2^1 T(n/2^1) + 2^0 cn$

$T(n) = 2\left[2T(n/4) + c\,n/2\right] + cn \Rightarrow 2^1 T(n/2^1) + 2^0 cn$

$T(n) = 4T(n/4) + c\,n/2 + cn$

$= 4T(n/4) + 2cn \Rightarrow 2^2 T(n/2^2) + 2^1 cn$

$T(n) = 4\left[2T(n/8) + cn/4\right] + 2cn + cn$

$= 8T(n/8) + 3cn \Rightarrow 2^3 T(n/2^3) + 3cn$

$\vdots$

$T(n) = 2^k T(n/2^k) + kcn$ $\quad \because n/2^k = 1$

$n = 2^k$

$k = \log n$

$= 2^k T(1) + kcn$

$= nc + \log n \cdot cn$

$= cn + cn\log n$

$= O(n\log n)$

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

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

a	b^x	$a = b^x$
2	2^1	$\Theta(n^x \log n)$

$2 = 2 \Rightarrow$

$\Theta(n^1 \log n) \Rightarrow \Theta(n \log n)$

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

$a = 2 \quad b = 2 \quad x = 3$

a	b^x	
2	2^3	$a < b^x$

$2 < 8 \Rightarrow \Theta(n^x)$

$= \Theta(n^3)$

$= 8T(n/8) + 7c \Rightarrow 8T(n/2^3) + 7c$

$\downarrow$

$(2^0 + 2^1 + 2^2)c$

$\vdots$

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

$= n^1(1) + kc$

$= nc + kc$

$= nc + \log n\, c$

~~$= c(n + \log n)$~~

~~$= O(\log n)$~~ $O(n)$

$T(n/2^k) = 1$

$n = 2^k$

$k = \log n$

$T(n) = 2^k T(n/2^k) + (2^0 + 2^1 + 2^2 + \cdots + 2^{k-1}) \cdot c$

$= 2^k T(n/2^k) + (2^k - 1) \cdot c$

$= n\,T(1) + nc - c$

$= nc + nc - c$

$= O(n)$

Master Theorem:

Recurrences:

$T(n) = 1T(n/2) + cn^0$

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

a	b^n
1	2^0
1 = 1	$\Rightarrow a = b^n$ $[\theta(n^n \log n)]$

$\downarrow$

$\theta(n^0 \log n) \Rightarrow \theta(\log n)$

Lecture: 12

10 Apr

Quick Sort: Best Case:

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

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

$T(n/4) = 2T(n/8) + (n/4 - 1)$

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

$= 2\left[2T(n/4) + (n/2 - 1)\right] + (n-1)$

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

$= 2^2 T(n/2^2) + 2n - 3 \Rightarrow (2^0 + 2^1)$

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

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

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

$= O(n \log n)$

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

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

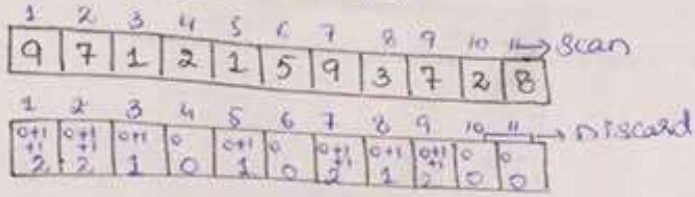
$T(n/4) = 2T(n/8) + n/2$

$T(n) = 2\left[2T(n/4) + n/2\right] + n$

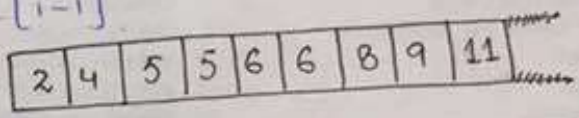
$= 2^2 T(n/2^2) + n + n$

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

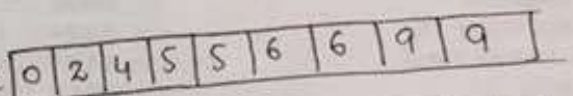
Count Sort: (small No's)



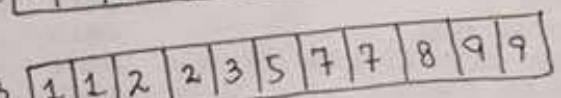

```
Count (A, n, max)
  {
    c[max] = 0;
  for(i=1 to max)
    c[max] = 0;
  for(i=1 to n)
      c[A[i]] = c[A[i]] + 1;        c[A[i]] += 1;
  for(i=2 to m)                     c[A[i]] +
    c[A[i]] = c[A[i]] + c[A[i]];  ↑ c[A[i+1]];
      c[i] = c[i] + c[i-1]
```




```
for(i=1 to 1)  => for (i=1 to 1)
```




```
  A[i] = c[A[i]];
  c[A[i]] = B[c[i]];
    c[A[i]]--;
```




```
for(i=n down to 1)
B[c[A[i]]] = A[i];
      c[A[i]] -= 1;
```

Substitution Method: Complexity of Binary Search:

$T(1) = c$

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

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

$T(n/4) = T(n/8) + c$

$T(n) = T(n/2) + c \Rightarrow T(n/2^1)$

$T(n) = [T(n/4) + c] + c$

$T(n) = T(n/4) + 2c \Rightarrow T(n/2^2)$

$T(n) = [T(n/8) + 2c] + 2c$

$T(n) = T(n/8) + 3c \Rightarrow T(n/2^3)$

$\vdots \qquad \vdots$

$T(n) = T(n/2^k) + kc \quad \because n/2^k = 1$

$= T(1) + kc$

$= c + kc$

$= O(k)$

$= O(\log(n))$

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

$n = 2^k$

$\log(n) = \log 2^k$

$\log(n) = k \log 2$

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

$k = \log(n)$

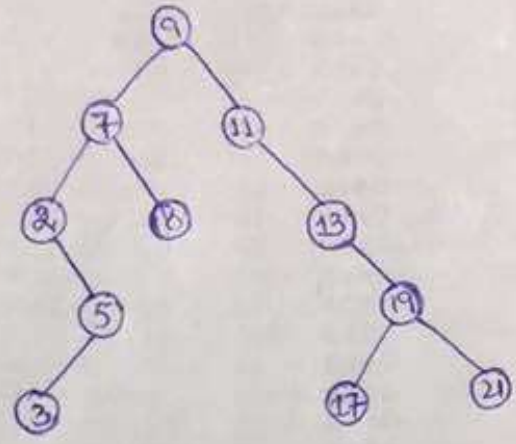
Lecture: 11

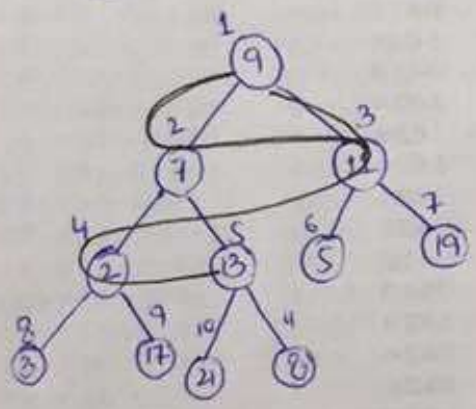
5 Apr

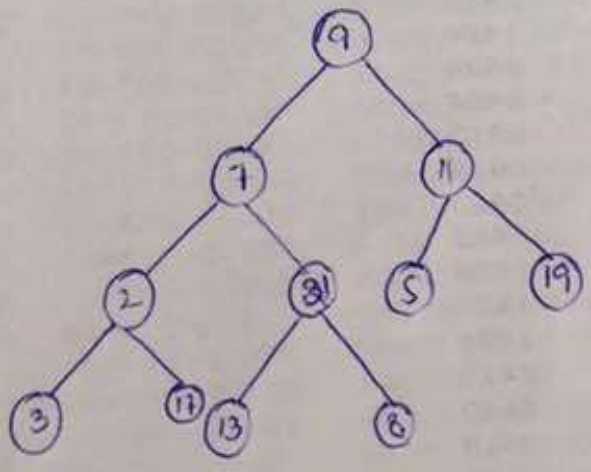
Quiz:

Sort the following Array using heap sort.

9	7	11	2	13	5	19	3	17	21	8







9	3	7	2	1

Iteration: 1:

3	9	7	2	1

3	7	9	2	1

3	7	2	9	1

3	7	2	1	9

Iteration: 2:

3	7	2	1	9

3	2	7	1	9

3	2	1	7	9

Iteration: 3:

2	3	1	7	9

2	1	3	7	9

Iteration: 4:

1	2	3	7	9

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

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

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

$= n + n\log n$

$= O(n \log n)$

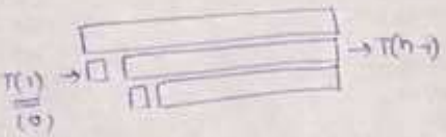
Worst Case:

$T(n) = T(0) + T(n-1) + (n-1)$

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

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

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



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

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

$T(1) = T(0) + 0$

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

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

$= O(n^2)$

Bubble Sort:

```
for(int i=0; i<n-1; i++)
for(int j=0; j<n-i-1; j++)
  {
    if(a[j] > a[j+1])
      swap(a[j], a[j+1])
  }
```

$O(n^2)$

Lecture: 13

12 Apr

Insertion Sort:

9	6	5	0	8	2	7	1	3

```
Insertion Sort(A, n)
{
  for(j=2; j<n; j++)
      key = A[j];
      i = j-1;
  while (i>0 && A[i]>key)
      A[i+1] = A[i];
      i = i-1;
      A[i+1] = key;
}
```

Worst Case:

$1 + 1 = 2 \quad 2(1)$

$2 + 2 = 4 \quad 2(2)$

$3 + 3 = 6 \quad 2(3)$

$\vdots$

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

$$2(1) + 2(2) + 2(3) + \cdots + 2(n-1)$$

$$\Rightarrow 2(1 + 2 + 3 + \cdots + (n-1))$$

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

$$= O(n^2)$$

Selection Sort:

2	5	9	3	1	7	8

```
selection sort(A,n)
{
  int min;
  for(int i=0; i<=n-1; i++)
      min= i;
  for(int j= j+1; j<n; j++)
     {
       if(A[j] < A[min])
         {
           min= j;
         }
         if(min != j)
           {
             swap(A[i] ↔ A[min])
           }
     }
```

Sequential Sort:

```
Sequential Sort(A, n)
{  for(int i=0; i<=n-1; i++)
   for(int j=0; j<=0; j++)
    if ( A[i] > A[i+1])
        swap( A[i], A[i+1])
```

9	6	5	0	1

i=0
A[i] = 9
A[i+1] = 6

6	9	5	0	1

```
Sequential Sort (A,n)
{
 for(int i=0; i<=n-1; i++)
   {
    for(int j= i+1; j<=n; j++)
      {  if (A[j] > A[i])
         {
           temp = A[i];
            A[i] = A[j];
            A[j] = temp;
         }
      }
   }
```

Complexity:

i=0 1 2 ---- n-1

j= n n-1 n-2 --- 1

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

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

$= O(n^2)$

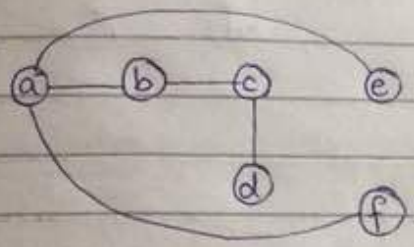
Lecture # 14 — After Mids

Graph:

$G_1 = (V, E)$

$V = \{a, b, c, d, e, f\}$

$E = \{(a, b), (b, c), (c, d), (a, e), (a, f)\}$.



Directed:

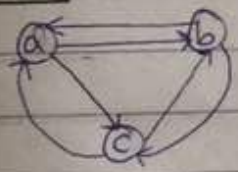
$a_i . a_j \neq a_j . a_i \quad i \neq j$

Undirected:

$a_i . a_j = a_j . a_i \quad i \neq j$

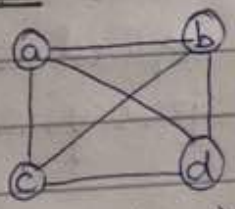
Complete: Every node connected with every node.

Directed:



$\frac{n(n-1)}{3(3-1)} = 6$

Undirected:



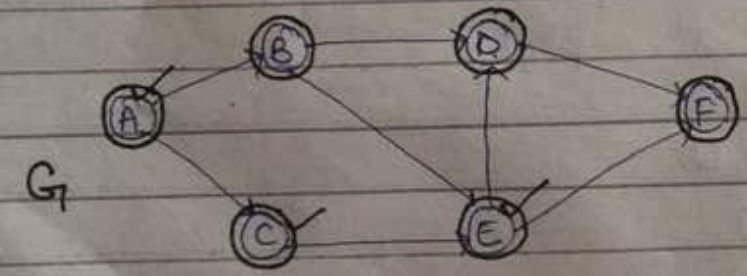
$\frac{n(n-1)}{2} = \frac{4(4-1)}{2}$

$= 6$

DFS: Depth first Search

```
DFS (G, s)
for each vertex v ∈ G(V)
    state(v) ← Ready

State (s) ← wait
push (S, s)
while (S ≠ ϕ)
    u ← pop (S)
for each vertex V ∈ Adj(u)
    if (state (V) = = Ready)
    push (S, V)
    State (V) ← wait
    State (u) ← processed.
```



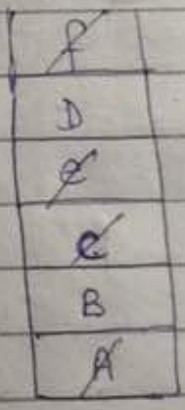
G

Starting vertex = A

U = A → B, C

U = C → E

U = E → D, F



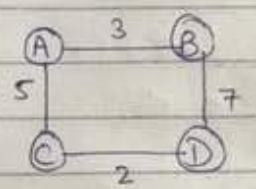
S

A	C	E	f	D	B

Lecture # 15

Minimum Spanning Tree:-

* Spanning Tree is weighted always and its edges can be connected in different ways.

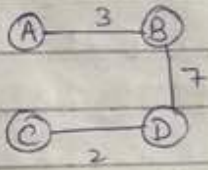


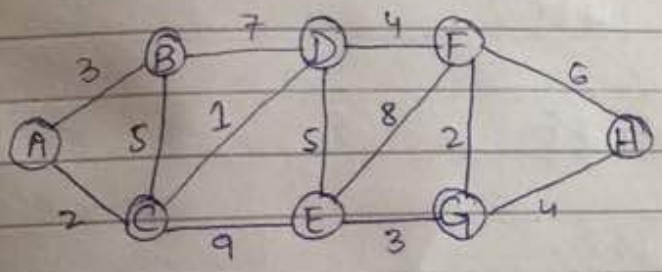
1) Kruskal:-

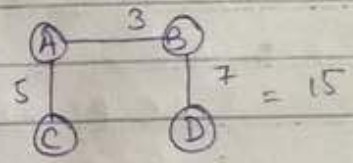
i) Sort edges

ii) consider each edge (min to max) if donot form cycle.

⇒ Concept of union set.

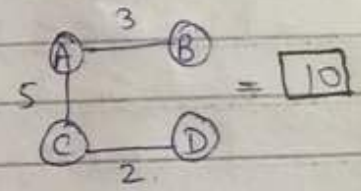


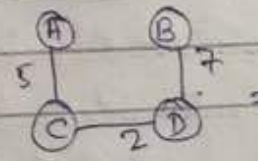


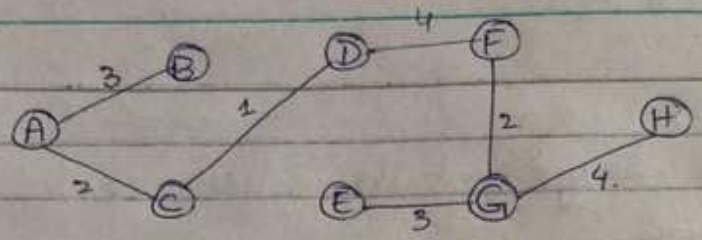


① Sort:-

C, D	1	E, F	8 X
F, G	2	C, E	9 X
A, C	2		
A, B	3		
D, F	4		
G, H	4		
D, E	5 X		
B, C	5 X		
F, H	6 X		
B, D	7 X		







$= 3+2+1+4+3+2+4$

$= 19.$

UV = {{A}, {B}, {C}, {D}, {E}, {F}, {G}, {H}}

1) C, D → 1

= {{A}, {B}, {C, D}, {E}, {F}, {G}, {H}}

2) F, G → 2.

= {{A}, {B}, {C, D}, {E}, {F, G}, {H}}.

3) A, C, 2

= {{A, C, D}, {B}, {E}, {F, G}, {H}}.

4) A, B, 3

= {{A, B, C, D}, {E}, {F, G}, {H}}.

5) E, G, 3

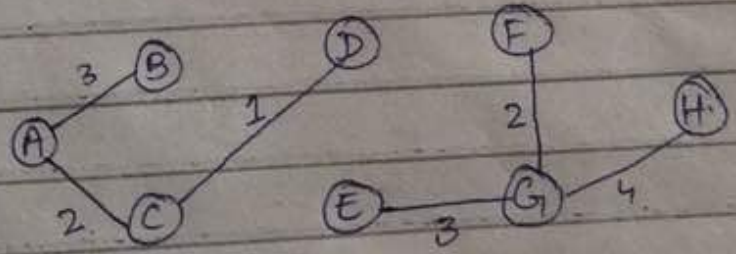
= {{A, B, C, D}, {E, F, G}, {H}}.

6) D, F, 4

= {{A, B, C, D, E, F, G}, {H}}.

7) G, H, 4

= {{A, B, C, D, E, F, G, H}}.



= 19.

Kruskal Algo:

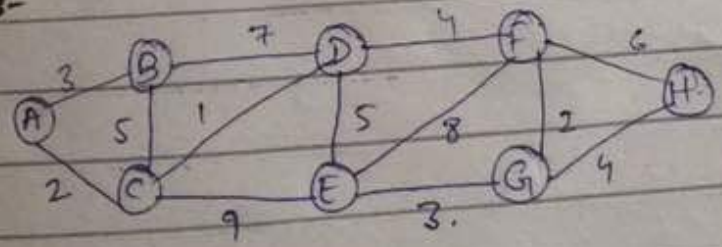
```
Kruskal (G, w)                                   T = φ
for each vertix v ∈ V                                          ] O(|V|)
    UV = UV ∪ {v}                                              ]
for each edge (u, v, w) ∈ E                                    ] n log n
    Enqueue (Q, u, v, w)                                       ] E log E
while (|UV| > 1)                                                          ]
(u,v,w) = Dequeue (Q)                                                     ] |E|
    if u, v belong to different set US1 & US2 of UV set                   ]
        UV = UV - US1
        UV = UV - US2
        UV = UV ∪ (US1 ∪ US2)
        T = T ∪ (u, v, w)
    return T;
```

UV = { }
UV = { {a} }
UV = { {a}, {b} }

u ∈ US1 && v ∈ US2 &&
US1 ≠ US2 && US1, US2 ∈ UV

UVset ↑
$|V| + |E| + |E \log E|$
↓ construction of MST ← sorting
$O(|V| + E \log E)$.

ii) **Prims :-**



Algo:

```
Prims(G, s)
T = ϕ
S = { }
S = S ∪ s
V = V - s
while (V ≠ ϕ)
  for each u ∈ S
   for each v ∈ Adj(u)
   d ← ∞
  if (d > u,v,w)
   d ← w
    p = u
    q = v
  S = s ∪ q
  U = v - q
  T = T ∪ (p, q, d)
return T;
```

V = {A, B, C, D, E, F, G, H}

S = { }

S = {A}

A

Complexity:-

O[V * V]

Algo: 1:

```
BFS (T, P)
n = length(T)
m = length(P)

for (i=0 to n-m)
  for( j = 1 to m)
    if (T(i+j) = P(j))
      if(j=m) *
       return i;
else
    brute inner loop
return -1;
```

```
001
1101001        Search 001
Mis-match.
 001
1101001  (Mismatch).
  001
1101001  (1st match, rest x)
   001
1101001  (1st x)
    001
1101001  (1st ✓, 2nd ✓, 3rd ✓)
```

3 = m
length of pattern.

* If characters in text are less than of pattern, then there will be no possibility left to further check.

Finite State Automata:

$\Sigma = \{5, 10\}$

Q = no. of state = 4.

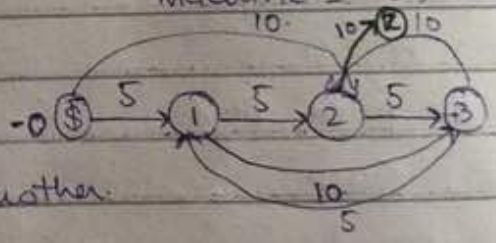
Starting state = -sign.

Final state = + sign.

Transition Table = change from one state to another.

Product = 15 Rs

Machine = 5, 10 coins.



	5	10
0	1	2
1	2	3
2	3	R(∞)
3	1	2

Transition Table

$\Sigma = \{a, b, c\}$

abba

abba

abba

abba

ab ✓

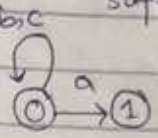
abba

ac

bba

$\Sigma \rightarrow a = a$, $b = b$, $c = c$

Check the prefix of S.P. suffix of set.

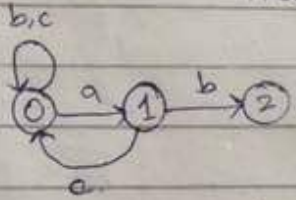


$a \rightarrow a = aa$ $aa = a$ ✓ $= 1$

$\rightarrow b = ab$ ✓ $= 2$

$\rightarrow c = ac$ $ac = c = 0$

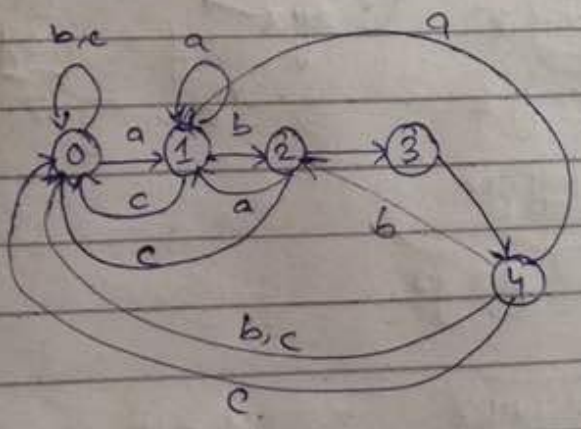
Since prefix, suffix doesn't match, so reduce both.



$ab \rightarrow a = aba = 1$

$\rightarrow ab = abb = 3$

$c = abc = 0$



$abba = 4$

$abbb = 0$

$abbc = 0$

	a	b	c
0	1	0	0
1	1	2	0
2	1	3	0
3	4	0	0
4	1		

Lecture # 17

FSA: $\Sigma = \{x, y, z\}$

pattern = xyxyz

length of pattern = 5

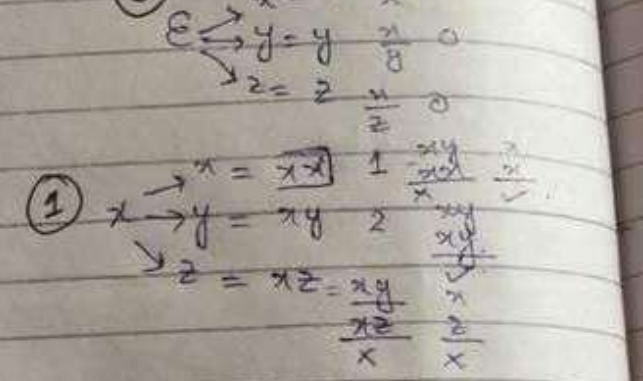
States will be from 0 to 5.

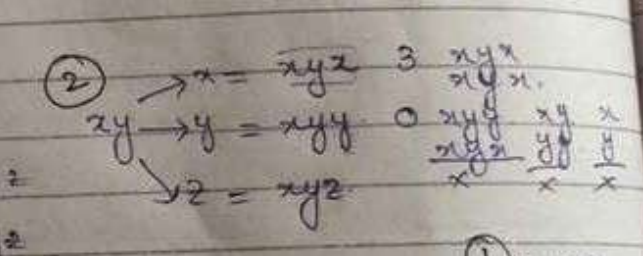
0 will be initial state

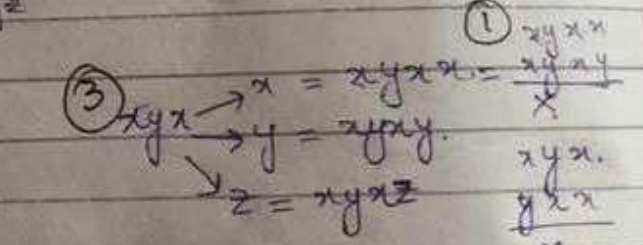
5 " " final "

$T = \{xxyyzxyxyzxxz\}$.

0	x	y	z
0	1	0	0
1	1	2	0
2	3	0	0
3	1	4	0
4	3	0	5
5	1	0	0







xyxyz

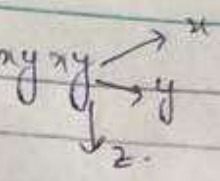
xyxyz

xyxyz

xyxyz

② xyxyz xy.xy = 4

③ xyxyz xyxz xyxyz xyxyz



④

① xyxyx

xyxyz → xyxyx → x

xyxy z → yxyx → ✗

xyx yz → xyx → ✓

② xyxyy

xyxyz → xyxyy → ✗

xyxy z → yxyy → ✗

xyx yz → xyy → ✗

xy xyz → yy → ✗

x yxyz → y → ✗

③ xyxyz

xyxyz → ✓

⑤ xyxyz → x, y, z

xyxyzx

xyxyzy

xyxyzz

① xyxyzx → xyxyz

xyzx → yxyz → ✗

xyzx → yxy z → ✗

xyz → xy yz

xyxyzx → yxyz

xyxyz

yxy → xyz

yxy.

xyz

```
Matcher (T, δ, m)
{ n ← length(T)
   q=0;     // Starting from initial state.
for(i=0 to n) ⟶ O(n).
    c = T[i]   // Read character from text.
    q= δ(q,c)   // See transition.
  if( q=m) // if state is final.
  print found at  i-m;  // stating index of pattern
                  return 1;
}
```

Lecture # 18

1, Dynamic Programming:

1, Decrease & Conquer
2, Divide & Conquer (Merge Sort)
3, Graph.

Divide a problem into overlapped subproblems.

i, Recurrence Relation
ii, Recursive Algorithm
iii, Solve for an instance
iv, Memoized Algo.
v, Pattern, ~~Divide~~ Develop iterative Algo.

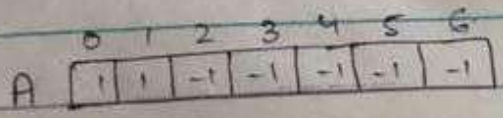
Example:

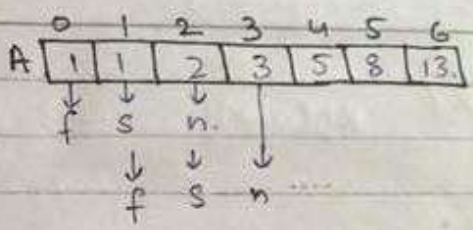
Series:

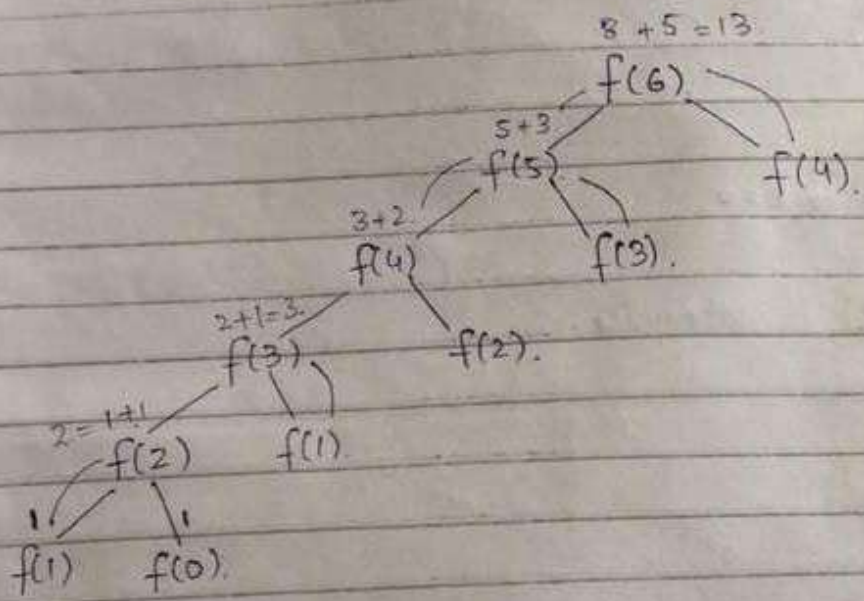
$$f(n) = \begin{cases} 1 & , \quad n \leq 1 \\ f(n-1) + f(n-2) & , \quad \text{otherwise} \end{cases}$$

n	0	1	2	3	4	5	6
f(n)	1	1	2	3	5	8	13

```
f(n)
{
  if (n ≤ 1)
    return 1
else
    return f(n-1) + f(n-2)
}
```







Iterative Algo:

```
{
  f = 1
  s = 1
  for ( i = 2 to n)
    n = f + s
    f = s
    s = n
}
```

```
A[0] = 1;
A[1] = 1;
for ( i = 2 to n)
  A[i] = -1;
```

Display n; (n)

Memoize Algo:

```
if (n)
{
if (A[n] == -1)
  return A[n]
else
A[n] = f(n-1) + f(n-2)
  return A[n];
}
```

Binomial Co-efficient:

$$B\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

$$\Rightarrow \frac{6!}{0!(6-0!)} = 1$$

$$\Rightarrow \frac{6!}{6![6-6)!} = 1.$$

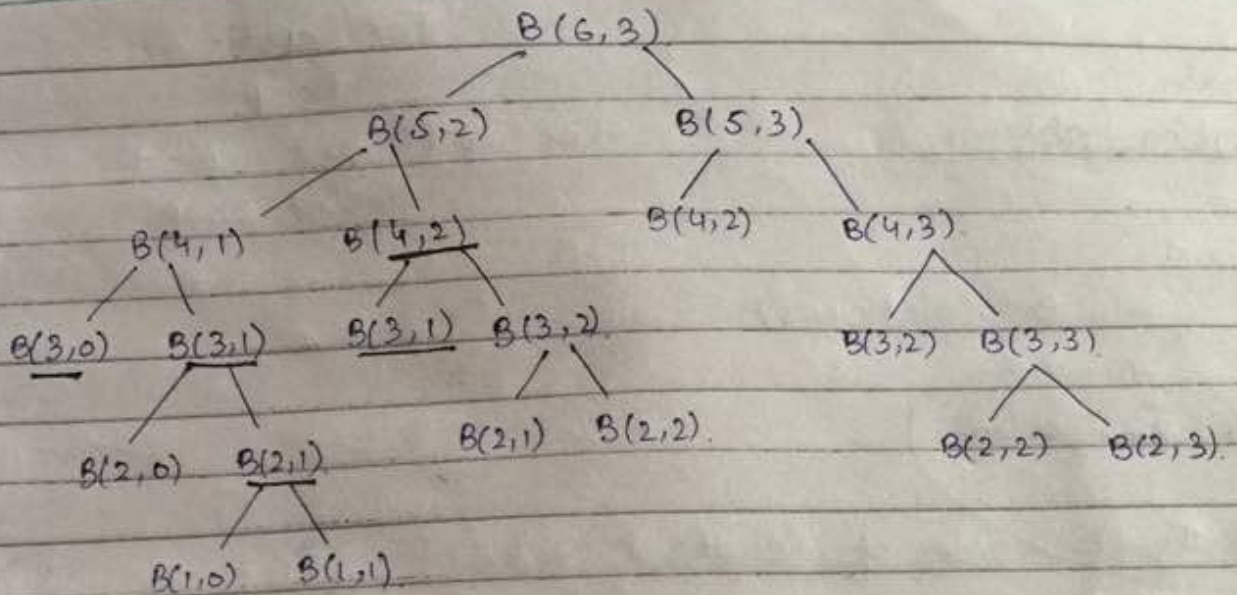
$$B(n,k) \begin{cases} 1 & , \quad n=k \text{ or } k=0 \\ B(n-1,k-1) + B(n-1,k) & \text{otherwise.} \end{cases}$$

Algo:

```
B(n,k)
if(n=k or k=0)
  return 1
else
return B(n-1,k-1) + B(n-1,k)
```

B(6,3)

	K →						
	0	1	2	3	4	5	6
0	1		i-1,j-1	i-1,j			
n↓ 1	1	1					
2	1	2	1				
3	1	3	3	1			
4	1	4	6	4	1		
5	1	-1	10	10	-1	1	
6	1	-1	-1	20	-1	-1	1.



Memoize Algo:

```
BC(n, k)
if(A[n][k] != -1)
    return A[n][k]
else
    A[n][k] = BC(n-1, k-1) + BC(n-1, k)
    return A[n][k]
```

```
for (i=0 to n)
    for (j=0 to k)
        if (i=j or j=0)
            A[i][j] = 1
        else
            A[i][j] = -1;
```

~~for (i=0 to n)~~

Iterative Algo:

A	B	C
2 3	3 5	5 6

① $(A*B)*C$.

$2\times3\times5$

$A*B = 2\ 5 = 30$.

$(A*B)*C = 2\times5\times6 = 60$

$= 30+60 = 90$

$[2\times3 + 3\times5]$

$6 + 15$

$[21] * C$.

$105 + 126$

231.

$21 \times 5 = 105$; $21 \times 6 = 126$; $105 + 126 = 231$

$45 \times 2 = 90$; $45 \times 3 = 135$; $90 + 135 = 225$

② $(B\times C) * A$

$B\times C = 3\times5\times6 = 90$.

$A*(B\times C) = 2\times3\times6$. 36

$= 90 + 36$

$= 126$.

$[3*5] + [5*6]$.

$15 + 30$

$45 * A$

$90 + 135$

225.

0,1 | 0,2

1,1 | 1,2

2,1 | 2,2

[2 3] [3 8]

$\Rightarrow A_1\ A_2\ A_3\ A_4$

$(A_1 * A_2) * (A_3 * A_4)$

$A_1 * (A_2 * A_3) * A_4$

Ways:-

$(A_1)\ (A_2\ A_3\ A_4)$.

$(A_1)\ ((A_2\ A_3)\ A_4)$. — (1)

$(A_1)\ (A_2\ (A_3\ A_4))$ — (2)

$(A_1\ A_2)\ (A_3\ A_4)$ — (3)

$(A_1\ A_2\ A_3)\ A_4$ — (4)

$((A_1\ A_2)\ A_3)\ A_4$ — (4)

$(A_1(A_2\ A_3)\ A_4)$ — (5)

$A_1 \dots A_4 = A_{1..1} \cdot A_{2..4}$
$= A_{1..2} \cdot A_{3..4}$
$= A_{1..3} \cdot A_{4..4}$

$A_{i..j} = A_{1..k} \cdot A_{k+1..j}$
$i \le k < j$

$A_{[1..1]} \; A_{k=1} \; (A_{2..n}) \; A_{[2..n]} \; A_{(k+1..j]}$
$A_1 \uparrow (A_2 \dots A_n)$
$A_{1..2} \; A_{k=2} \; A_{3..n}$
$A_1 A_2 \uparrow (A_3 \dots A_n)$
$(A_1 A_2 A_3) A_4 \dots A_n)$
$A_{[1..(n-1)]} \; k=n-1$
$(A_1 \dots A_{n-1}) \uparrow (A_n)$
$A_{[n..n]}$

* $m[i,i] = 0$
$\min[i \le k < j]($
$m[i,j] = m[i..k] + m[k+1,j] +$
$P_{i-1} P_k P_j)$

$A_1 \; A_2 \; A_3 \; A_4$
$2\;3 \quad 3\;4 \quad 4\;5 \quad 5\;2$
$P_0 P_1 \quad P_1 P_2 \quad P_2 P_3 \quad P_3 P_4$

	1	2	3	4
1	0 1	24 5	64 8	76 10
2		0 2	60 6	64 9
3			0 3	40 7
4				0 4

$m[1..2] = k=1 \quad m[1..1] + m[2..2] + P_{i-1} \cdot P_k \cdot P_j$
$0 + 0 + 2 \cdot 3 \cdot 4$
$= 24$

$m[2..3] = k=2 \quad m[2..2] + m[3..3] + P_{i-1} \cdot P_k \cdot P_j$
$0 + 0 + P_1 \cdot P_2 \cdot P_5$
$3 \cdot 4 \cdot 5$
$= 60$

$m[3..4] = k=3.$

$m[3..3] + m[4..4] + P_{i-1} \cdot P_j \cdot P_k.$

$0 + 0 + P_2 \cdot P_4 \cdot P_3.$

$4 \cdot 2 \cdot 5$

$= 40.$

$m[1..3]$ $k=1$

$m[1..1] + m[2..3] + P_{i-1} \cdot P_j \cdot P_k.$

$0 + 60 + P_0 \cdot P_1 \cdot P_3$

$60 + 2 \cdot 3 \cdot 5$

$60 + 30$

$= 90$

$k=2$

$m[1..2] + m[3..3] + P_{i-1} \cdot P_j \cdot P_k.$

$24 + 0 + P_0 \cdot P_3 P_2.$

$24 + 2 \cdot 4 \cdot 5$

$= 64.$

$m[2..4]$ $k=2.$ ~~$k=3$~~

$m[2..2] + m[3..4] + P_{i-1} \cdot P_j \cdot P_k.$

~~6~~$0 + 40 + P_1 P_4 P_2.$

~~6~~$0 + 40 + 3 \cdot 4 \cdot$ ~~8~~ $2.$

40~~0~~ $+$ ~~24~~ 24

~~125~~ $64.$

$k=3$

$m[2..3] + m[$~~4~~$..4] + P_{i-1} \cdot P_j \cdot P_k.$

$60 +$ ~~4~~$0 + P_1 \cdot P_4 \cdot P_3.$

~~6~~$0 + 3 \cdot$ ~~4~~$\cdot 5$

~~60~~$. 90.$

	A_1	A_2	A_3	A_4	A_5
A_1	0	120 (k=1)	88 (k=1) 5	158 (k=3) 8	160 (k=3) 10
A_2		0	48 (k=2) 2	(k=3) 1046	114 (k=3) 9
A_3			0	(k=3) 84.3	(k=3) 787
A_4				0	42 (k=4) 4
A_5					0

$m[1..2] = m[1..1] + [2..2] + P_0 P_1 P_2$.

$k=1 \quad = 0 + 0 + 5 \times 4 \times 6$

$= 120.$

$m[2..3] = m[2..2] + m[3..3] + P_1 P_2 P_3$.

$k=2 \quad = 0 + 0 + 4 \times 6 \times 2$.

$= 48.$

$m[3..4] = m[3..3] + m[4..4] + P_2 P_3 P_4$

$k=3 \quad = 0 + 0 + 6 \times 2 \times 7$

$= 84.$

$m[4..5] = m[4..4] + m[5..5] + P_3 P_4 P_5$

$k=4 \quad = 0 + 0 + 2 \times 7 \times 3$

$= 42.$

$m[1..3] = m[1..1] + m[2..3] + P_0 P_1 P_3$.

$k=1 \quad = 0 + 48 + 5 \times 4 \times 2$

$= 48 + 40$

$= 88.$ ✓

$m[1..4] = m[1..1] + m[2..4] + P_0 P_1 P_4$

$k=1 \quad = 0 + 64 + 2 \times 3 \times 2$

$= 64 + 12$

$= 76$ ✓

$k=2 = m[1..2] + m[3..4] + P_0 P_2 P_4$

$= 24 + 40 + 2 \times 4 \times 2$

$= 64 + 16$

$= 80$

$k=3 = m[1..3] + m[4..4] + P_0 P_3 P_4$

$= 64 + 0 + 2 \times 5 \times 4$

$= 64 + 40$

$= 104.$

Lecture # 20

$A_1 \quad A_2 \quad A_3 \quad A_4 \quad A_5$

$5 \times 4 \quad 4 \times 6 \quad 6 \times 2 \quad 2 \times 7 \quad 7 \times 3.$

$P_0 \quad P_1 \quad P_2 \quad P_3 \quad P_4 \quad P_5$

$m[i..i] = 0$

$m[i..j] = \min_{i \le k < j} (m[i..k] + m[k+1..j] + P_{i-1} P_k P_j).$

$k = 2 \quad = m[1..2] + m[3..4] + P_0 P_2 P_4$

$= 120 + 84 + 5 \times 6 \times 7$

$= 120 + 84 + 210$

$= 414$

$k = 3 \quad = m[1..3] + m[4..4] + P_0 P_3 P_4$

$= 88 + 0 + 5 \times 2 \times 7$

$= 88 + 70$

$= 158$ ✓

$m[2..5] \quad = m[2..2] + m[3..5] + P_1 P_2 P_5$

$k = 2 \quad = 0 + 78 + 4 \times 6 \times 3$

$= 78 + 72$

$= 150$

$k = 3 \quad = m[2..3] + m[4..5] + P_1 P_3 P_5$

$= 48 + 42 + 4 \times 2 \times 3$

$= 48 + 42 + 24$

$= 114$

$k = 4 \quad = m[2..4] + [5..5] + P_1 P_4 P_5$

$= 104 + 0 + 4 \times 7 \times 3$

$= 104 + 84$

$= 188$

28
× 3
84

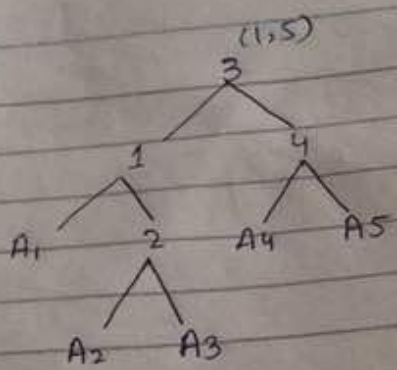
```
m[i,j] ← ∞
 for k=1 , j-1.
 do t ← m[i,k] + m[k+1, j] + Pi-1 . Pk. Pj
  if ( t < m[i,j])
then  m[i,j] ← t ,  s[i.j] ← k.
```

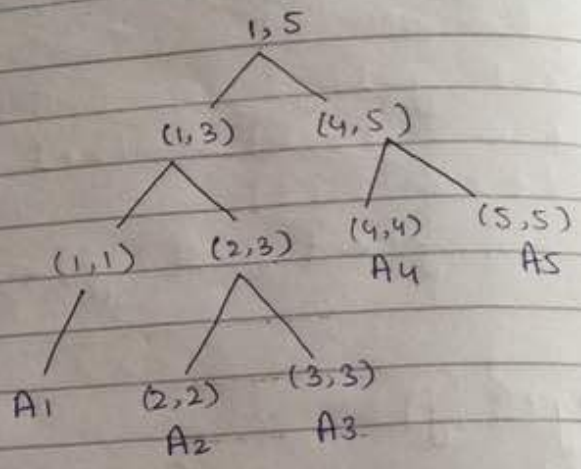
```
Multiply ( i , j)
if (i = j)
 then return A[i]
else  k ← s[i,j]
 X ← MULTIPLY (i , k)
 Y ← MULTIPLY ( k+1 , j)
return X.Y
```

A1 A2 A3 A4 A5

((A1 (A2 A3)) (A4 A5))

4.6.2	= 48
2.7.3	= 40
4.2	= 42
2.3	= 30
5.4 4.2	160
5.4.2	
5.2 2.3	
5.2.3	
5.3	





$S_1 =$
$S_2 =$

Lecture # 21

Longest Common Subsequence:

$A[i,j] = A[i-1, j-1] + 1$, $S_1[i] = S_2[j]$.

$A[i,j] = \max A[i-1, j], A[i, j-1]$ otherwise.

$A[i-1, j-1]$	$A[i-1, j]$
$A[i, j-1]$	$A[i, j]$

S_1 = A C D X Y

S_2 = A G X D F

		A	C	D	X	Y
	0	0	0	0	0	0
A	0	1	← 1	← 1	← 1	← 1
G	0	↑ 1	← 1	← 1	← 1	← 1
X	0	↑ 1	← 1	← 1	2	← 2
D	0	↑ 1	← 1	2	← 2	← 2
F	0	↑ 1	← 1	↑ 2	← 2	← 2

AD

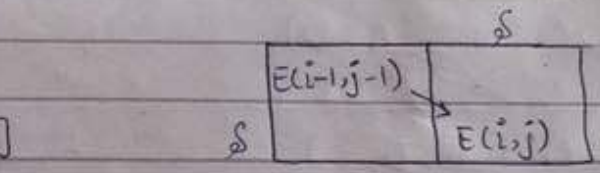
```
BT (i,j)
while (i>0 || j>0)
{ if ( A[i,j] = A[i-1,j-1] +1)
        S = S1[i] ∪ S
        i--;
        j--;
else
    if ( A[i,j] = A[i, j-1])
            j--;
else
        i--;
}
```

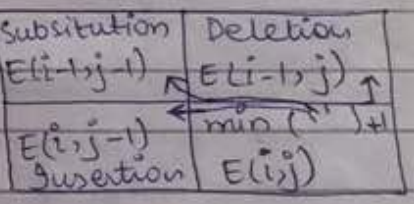
Lecture # 22

EDIT DISTANCE:

	NULL	F	R	E	Q	U	E	N	T
F Insert	0 →	1 →	2 →	3 →	4 →	5 →	6 →	7 →	8
NULL Sub.	1 ↓	1 ↘	2 ↘	3 ↘	4 ↘	5 ↘	6 ↘	7 ↘	8 ↘
R	2 ↓	2 ↘	2 ↘	2 ↘	3 →	4 →	5 ↘	6 →	7 →
E	3 ↓	3 ↘	3 ↘	3 ↘	2 ↓	3 →	4 →	5 →	6 →
Q	4 ↓	4 ↘	4 ↘	4 ↘	3 ↓	2 ↘	3 →	4 →	5 →
U	5 ↓	5 ↘	5 ↘	4 ↓	4 ↘	3 ↓	2 ↘	3 →	4 →
E	6 ↓	6 ↘	6 ↘	5 ↓	5 ↘	4 ↓	3 ↓	2 ↘	3 →
N	7 ↓	7 ↘	7 ↘	6 ↓	6 ↘	5 ↓	4 ↓	3 ↓	3 ↘
delete C	8	8	8	7	7	6	5	4	4
T Sub.									

$$E(i,j)\begin{cases} E(i-1,j-1) & A[i]=B[i] \\ \min\begin{pmatrix} E(i-1,j-1) \\ E(i-1,j) \\ E(i,j-1) \end{pmatrix} + 1 & \text{otherwise} \end{cases}$$





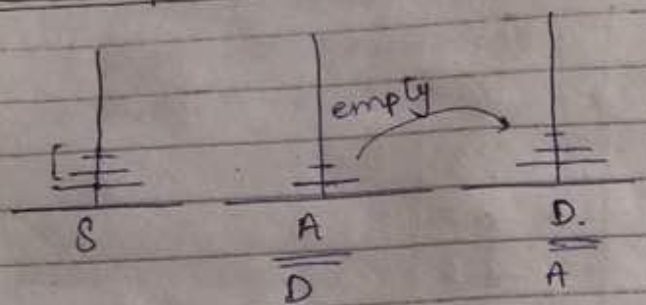
$\overbrace{CATS}^{i-1}$ $\overbrace{ca\,t\,i\,S}^{j-1}$ Subsitute.

$\overbrace{CATS}^{i}$ $\overbrace{CAT\,S}^{j-1}$ Insertion

$\overbrace{CATS}^{i-1}$ $\overbrace{CAT}$

Lecture No 23

Tower of Hanai:



S, S, D
M, S, A } n-1
S, D, A
L, S, D ⇒ Last one.
S, A, S
M, A, D
S, S, D

```
TOH(n)
if(n=1)
move(n,S,D,A)
else
move(n-1, S, A, D)
move(n, S, D, A)
move(n-1, A, D, S)
```

Knap Sack Problem:

items = $i_1, i_2, i_3 \ldots, i_n$

Values = $v_1, v_2, v_3, \ldots, v_n$

Weight = $w_1, w_2, w_3, \ldots, w_n$

Weight of KS = W

KS
- fractional KSP
- 0/1 KSP

Lecture # 24

Example:

n = 4 (# of elements)

8 = 5 pounds (max size)

Elements (size, value) = {(1,200), (3, 240), (2, 140), (5, 150)}.

Q① Assignment:

item = 1, 2, 3, 4, 5

weights = 3, 5, 3, 2, 2

Values = 5, 5, 6, 7, 4

W = 8

i/s	0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
1	0	0	0	5	5	5	5	5	5
2	0	0	0	5	5	5	5		10
3	0	0	0	6	6	6	11	11	11
4	0	0	7	7	7	13	13	13	18
5	0	0	7	7	11	18			

Q2

item = 1, 2, 3, 4

Weight = 4, 5, 3, 5

Values = 2, 3, 4, 5

W = 10.

Val. Weight.

5 3 3>1 3>2 3=3 ✓ (5)

5 5. 5>1 max((5+0, 5) 5), => max(5+0, 5), max(5+0, 5), max(6+0, 5), max(6+5, 5)

6 3 max(6+0, 5), max(6+0, 5), max(6+0, 5), max(6+5, 5)

7 2 max(7+0, 6), max(7+0, 6), max(7+6, 6), max(7+6, 6),

4 2 ~~max(4+0, 7), max(4+0, 7), max(4+7, 7), max(4+7, 7),~~

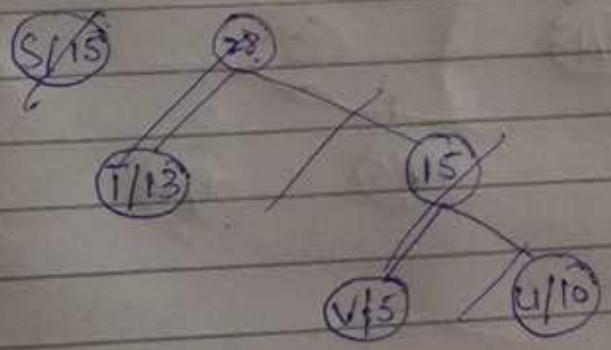
max(4+0, 7), max(4+0, 7), max(4+7, 7), max(4+7, 13)

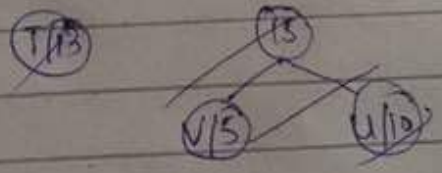
Lecture # 26

Huffman Coding:

Character	Frequency	Fixed-length codeword	Variable
P	30	000	01
Q	25	001	11
R	20	010	000
S	15	011	001
T	13	100	101
U	10	101	1000
V	5	110	1001

28
15





Huffman (c)

$n \leftarrow |c|$ → set of n characters

$Q \leftarrow c$ → min priority queue

for $i = 1$ to n

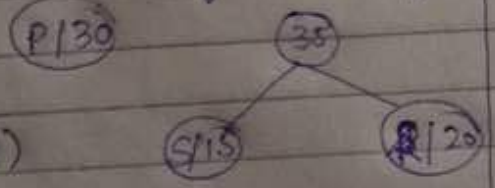
do $n-1$ → allocate a new node.

left $[z] \leftarrow x \leftarrow$ Extract-Min (Q)

Right $[z] \leftarrow y \leftarrow$ Extract-Min (Q).

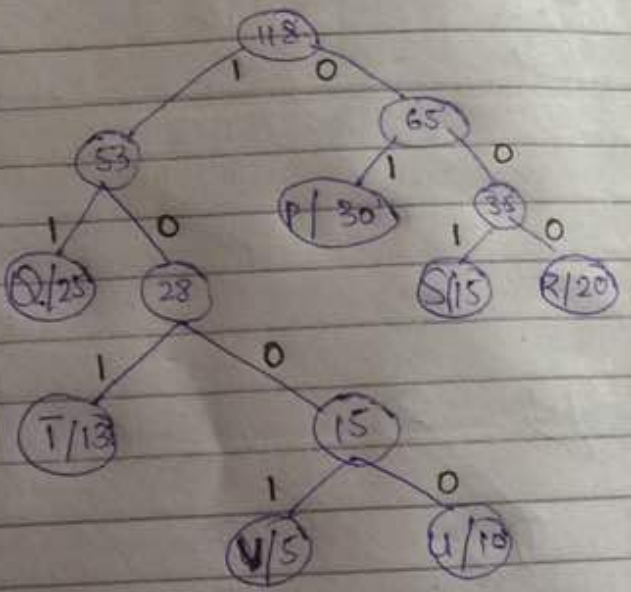
$f[z] \leftarrow f[x] + f[y]$

Shift it to Right



insert (Q, z)

Return Extract-Min (Q).



Lecture # 25

	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	2	2	2	2	2	2	2
2	0	0	0	0	2	3	3	3	3	5	5
3	0	0	0	4	4	4	4	6	7	10	11
4	0	0	0	0	5	6	7	8	9	10	11

Val Weight

1 2 4

2 3 5

3 4 3

4 5 4

2,6 = (3+0, 2)

1,6 = 2 + ~~0~~ = ~~2~~

item-1 weight

(2,6)

(1,6) = 2

V; (1; 1) max(2,3)

3+(0)

2,5 = 3.

3,5 = (4 + T[2][2]) = (4+0 = 4, 3)

3,9 = 5

3,10 = (4+7 = 11)

3,9 = (4+5

2,9 = 5

4,4 = 4 (5+0)

3,4 = 4

3,6 = 4

4,6 = (5+2=7)

4,5 = (5+3

3,7 = 6

4,7 = (5+3=8)

3,8 =

4,8 = (5+4 = 9

3,9

4,9 = 5+5 = 10

4,10 = (5+5

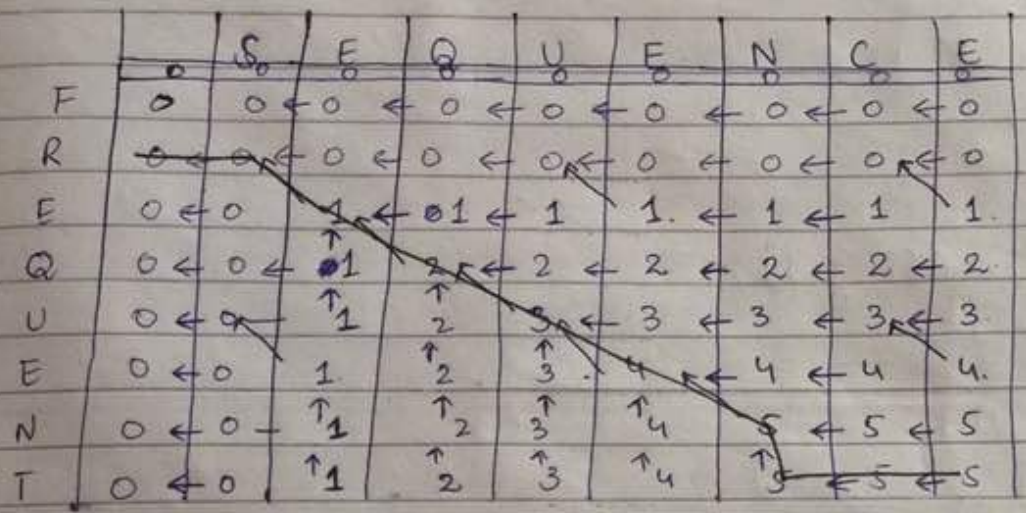
Items	Weight	Value	Density = V/W	Sort based on density.
i_1	5 ⇒	~~3~~ 1.25	= 2 × 0.6 — ⑥	
i_2	2	3	1.5 — ④	
i_3	3	7	2.33 — ②	
i_4	1	4	4 — ①	
i_5	3	5	1.66 — ③	
i_6	4	4	1 — ⑤	
		24.2.		

WKS = 15 kg.

i_1	2	1.2.
i_6	4	4
i_2	2	3
i_5	3	5
i_3	3	7
i_4	1	4

15 kg. | 14kg | 11kg | 8 kg | 6kg | 2kg.

S_1 = S E Q U E N C E
S_2 = F R E Q U E N T



EQUEN = LCS

Algo:

```
LCS (S1, S2)
for (i=0 to length (S1))
   A[0, i] = 0
for (j=0 to length (S2))
    A[i, 0] ← 0
for (i = 0 to length (S1))
  for (j=0 to length (S2))
    if (S1[i] = S2[j])
       A[i,j] = A[i-1, j-1] + 1.
    else
    A[i,j] = max (A[i-1,j] , A[i, j-1]).
```

20
×7

$m[1..5] = m[1..1] + m[2..5] + P_0 P_1 P_5$

$k=1 \quad = 0 + 114 + 5 \times 4 \times 3$

$= 114 + 60$

$= 174$

$k=2 \quad = m[1..2] + m[3..5] + P_0 P_2 P_5$

$= 120 + 78 + 5 \times 6 \times 3$

$= 120 + 78 + 90$

$= 288$

$k=3 \quad = m[1..3] + m[4..5] + P_0 P_3 P_5$

$= 88 + 42 + 5 \times 2 \times 3$

$= 88 + 42 + 30$

$= 160$ ✓

$k=4 \quad = m[1..4] + [5..5] + P_0 P_4 P_5$

$= 158 + 0 + 5 \times 7 \times 3$

$= 158 + 101$

$= 263$

Algo:

Matrix chain (p, N)

for i = 1 to N

do m[i, i] ← 0

for L = 2, N

do

for i = 1, n - L + 1

do j ← i + L - 1

$k = 2 = m[1..2] + [3..3] + P_0 P_2 P_3$

$= 120 + 0 + 5 \times 6 \times 2$

$= 120 + 60$

$= 180$

$m[2..4] = m[2..2] + m[3..4] + P_1 P_2 P_4$

$k = 2 \quad = 0 + 84 + 4 \times 6 \times 7$

$= 84 + 168$

$= 252$

$k = 3 \quad = m[2..3] + m[4..4] + P_1 P_3 P_4$

$= 48 + 0 + 4 \times 2 \times 7$

$= 48 + 56$

$= 104$ ✓

$m[3..5] = m[3..3] + m[4..5] + P_2 P_3 P_5$

$k = 3 \quad = 0 + 42 + 6 \times 2 \times 3$

$= 42 + 36$

$= 78$ ✓

$k = 4 \quad = m[3..4] + m[5..5] + P_2 P_4 P_5$

$= 84 + 0 + 6 \times 7 \times 3$

$= 84 + 126$

$= 210$

$m[1..4] = m[1..1] + [2..4) + P_0 P_1 P_4$

$k = 1 \quad = 0 + 104 + 5 \times 4 \times 7$

$= 104 + 140$

$= 244$

$168 + 84 = 252$

$42 \times 3 = 126$

Lecture # 19

For repetition.
Loop

For repitition's repitition.
Nested Loop.

Matrix Chain Multiplication Problem:

Matrix Multiplication:

A × B (Matrices)

same.

p q q r

Row column Row column.

C (Resultant) → order of Resultant Matrix.

p r

No. of multiplication = $p \times q \times r$

A[p][q], B[q][r], C[p][r]

for (i=1 to p) → for each row of A. → p

for (j=1 to r) → for each column of B → r

C[i][j]=0;

for (k=1 to q) → for multiplication → q

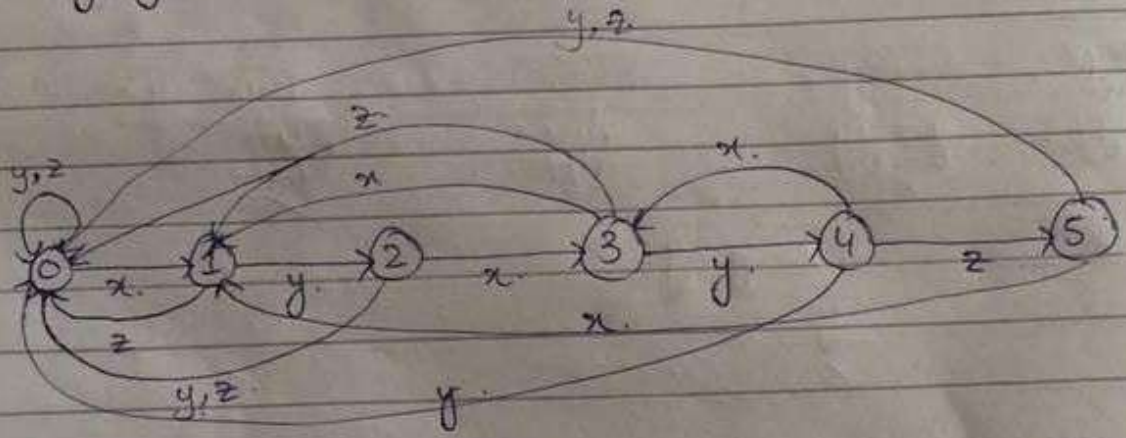
p*q*r

C[i][j] = C[i][j] + A[i][k] * B[k][j] →

xy xyz x 1
xy xy zx x.

xyxy z 0
x yxyzy

xy xy zz 0
xy xyz.



Algo: FSA(P, Σ) $|m| * |\Sigma| * |m|$.

m ← length(P) (m)

for (q=0 to m) q: states. |Σ|

for each character c in Σ

k ← min(m+1, q+2)

do

K = K-1 |m|

until Prefix P_k matches with suffix $q_k c$.

q_k: previous stage.
c: current character

δ(sigma)(q, c) ← k.

return δ;

Lecture NO 16

String Matching:

Application: Word processor (Spell check, Search/find word)
Digital Library
Search Query
Virus Scanning
Email filtering

Empty String ε " "
Length (String) = 6
no. of characters
String you want to search = pattern.
In which you want to search = Text.
Prefix = Starting characters [
Suffix = Ending characters]

[2 BSIT = BS
[1 BSIT = B

BSIT]2 = IT

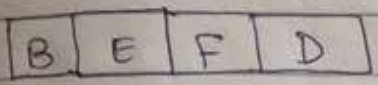
BSIT]3 = SIT

Four Algo's:
1. Brute force Algo
2. Robin Karp
3. FSA based String.
4. Knuth Moris Pratt.

Starting vertex: B

U = B → D
B → E

U = E → F



Algo:-

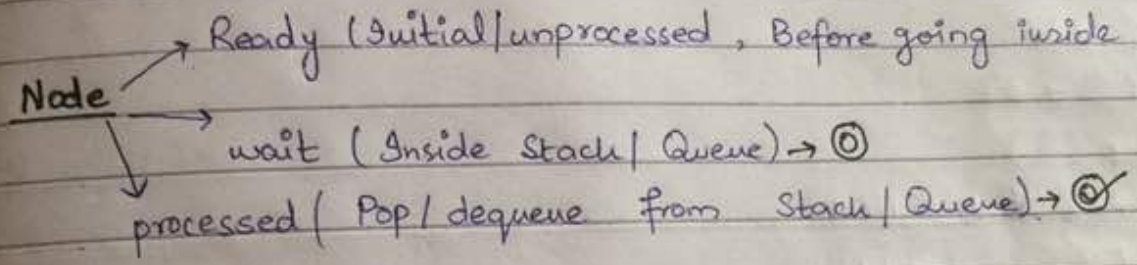
BFS:-

```
BFS (G, S)
for each vertex v ∈ G(V)
  State (v) ← Ready
state (s) ← wait
po Enqueue (Q, s)
while (Q, s)
   u ← dequeue (Q).
for each vertex v ∈ Adj (u)
  if (state (v) == Ready)
  Enqueue (Q, v)
  State (v) ← wait
  state (u) ← processed.
```

Graph Traversal:

↳ DFS (Stack)
Depth First Search

↳ BFS (Queue)
Breadth First Search



i) Initial State
ii) Mark all other state as Ready.
iii) Push/Enqueue the Initial State.
iv) Change starting state to wait.
v) While stack/Queue is not empty.

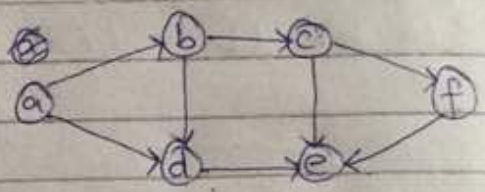
```
        U ← Pop (S)
            Dequeue (Q).
for each vertex V ∈ Adj(U)
    if (state(V) == Ready)
        Enqueue/push (S, V)
        State (V) ← wait
    State (U) ← processed
```

Weighted Graph:

$a_i . a_j = w$

Graph Representation:

↳ Adjacency Matrix
↳ Adjacency List



$$A_{ij} = \begin{cases} 1, & a_i . a_j \in E \\ 0, & \text{otherwise.} \end{cases}$$

	a	b	c	d	e	f
a	0	1	0	1	0	0
b	0	0	1	1	0	0
c	0	0	0	0	1	1
d	0	0	0	0	1	0
e	0	0	0	0	0	0
f	0	0	0	0	1	0

Adjacency List:

a → [b] [d]
b → [c] [d]
c → [e] [f]
d → [e]
e
f → [e]

* Less space required in adjacency list than adjacency matrix.

$$= \lim_{n\to\infty} \frac{\frac{1}{n}(\ln 2)}{\ln 2 \cdot 2n}$$

$$= \frac{\ln 2}{n} \times \frac{1}{\ln 2 \cdot 2n}$$

$$= \frac{1}{n \cdot 2n}$$

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

$$= \frac{1}{\infty}$$

$$= 0 + 1 + 0$$

$$= 1$$

$\Rightarrow \lg(n) + n^2 + n = O(n^2)$

$\lg(n) + n^2 + n = \Omega(n^2)$

$\lg(n) + n^2 + n = \Theta(n^2)$

$n \lg n = O(n^2)$

$$= \lim_{n\to\infty} \frac{n \lg n}{n^2}$$

$$= \lim_{n\to\infty} \frac{\lg n}{n} = \frac{\infty}{\infty} \text{ (undefined)}$$

$\because \lg n = \frac{\ln n}{\ln 2}$

$$= \lim_{n\to\infty} \frac{\ln n / \ln 2}{n}$$

$$= \lim_{n\to\infty} \frac{\ln n}{\ln 2} \times \frac{1}{n}$$

$$= \lim_{n\to\infty} \frac{d/dn \ln n}{d/dn (\ln 2 \cdot n)}$$

$$= \lim_{n\to\infty} \frac{1/n \cdot \ln 2}{\ln 2 (1)}$$

$$= \lim_{n\to\infty} \frac{\ln 2}{n} \cdot \frac{1}{\ln 2}$$

$$= \frac{1}{\infty} = 0$$

$\Rightarrow n \lg(n) = O(n^2)$

$n \lg(n) \neq \Omega(n^2)$

$n \lg(n) \neq \Theta(n^2)$

Algo.

$$= \lim_{n\to\infty} \frac{\ln n}{n} \cdot \frac{1}{\ln n}$$

$$= \frac{1}{\infty} = 0$$

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

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

$$= \lim_{n\to\infty} \frac{\frac{n^2}{2} + \frac{n}{2}}{n^2}$$

$$= \lim_{n\to\infty} \frac{\frac{n^2}{2}}{n^2} + \frac{\frac{n}{2}}{n^2}$$

$$= \lim_{n\to\infty} \frac{n^2}{2} \cdot \frac{1}{n^2} + \lim_{n\to\infty} \frac{n}{2} \cdot \frac{1}{n^2}$$

$$= \frac{1}{2} + \frac{1}{2\cdot\infty}$$

$$= \frac{1}{2} + \frac{1}{\infty}$$

$$= \frac{1}{2} + 0$$

$$= \frac{1}{2} \Rightarrow (0 \le c < \infty) \Rightarrow (0 < c \le \infty) \Rightarrow (0 < c < \infty)$$

$$\frac{n(n+1)}{2} = O(n^2) \qquad \frac{n(n+1)}{2} = \Omega(n^2) \qquad \frac{n(n+1)}{2} = \Theta(n^2)$$

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

$$= \lim_{n\to\infty} \frac{n^2}{2^n}$$

$$= \lim_{n\to\infty} \frac{\infty}{\infty} \text{ undefined}$$

By applying L. Hopital Rule

$$= \lim_{n\to\infty} \frac{\frac{d}{dn}(n^2)}{\frac{d}{dn}(2^n)}$$

$= \lim_{n\to\infty} \frac{2n}{2^n \cdot \ln 2}$

$= \frac{\infty}{\infty}$ (undefined)

By applying derivative

$= \lim_{n\to\infty} \frac{2n}{2^n \cdot \ln 2} \Rightarrow \lim_{n\to\infty} \frac{2}{2^n \cdot (\ln 2)^2}$ → constant value

$\lim_{n\to\infty} \frac{2}{2^n \cdot \ln 2 \cdot \ln 2}$

$= \frac{2}{2^\infty}$

$= \frac{2}{\infty}$

$= 0$

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

$n^2 \neq \Omega(n^2)$

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

$\lg(n) + n^2 + n = \Theta(n^2)$

$= \lim_{n\to\infty} \frac{\lg(n)}{n^2} + \frac{n^2}{n^2} + \frac{n}{n^2}$

$= \lim_{n\to\infty} \frac{\lg(n)}{n^2} + \lim_{n\to\infty} \frac{n^2}{n^2} + \lim_{n\to\infty} \frac{n}{n^2}$

$= \frac{\infty}{\infty} + 1 + \frac{1}{\infty}$

$= 1 + 0$

$= 1$

$= \lim_{n\to\infty} \frac{\lg(n)}{n^2} \rightarrow \lim_{n\to\infty} \frac{\ln n}{\ln 2}$

$= \lim_{n\to\infty} \frac{\frac{d}{dn}(\lg(n))}{\frac{d}{dn}(n^2)}$ $= \lim_{n\to\infty} \frac{\ln n / \ln 2}{n^2/1}$

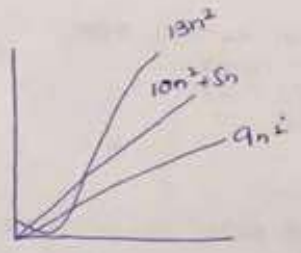
$= \lim_{n\to\infty} \frac{1}{2n}$ $= \lim_{n\to\infty} \frac{\ln n}{\ln 2 \cdot n^2}$

$= \lim_{n\to\infty} \frac{\frac{d}{dn}(\ln n)}{\frac{d}{dn}(\ln 2 \cdot n^2)}$

Ω (Omega Notation): Lecture: 6 20 Mar

Standard function is below your Alg.

We talk about best case.



$0 \le c \le \infty$
$f(n) = O(g(n))$
$0 < c \le \infty$
$f(n) = \Omega(g(n))$
$0 < c < \infty$
$f(n) = \Theta(g(n))$

$10n^2+5n = O(13n^2)$

$10n^2+5n = \Omega(9n^2)$

$\lg(n) = O(n)$

$$\lim_{n\to\infty} \frac{\log(n)}{n} = \frac{\infty}{\infty} \text{ (undefined)}$$

L. Hopital Rule

Since $\lg(n) = \frac{\ln n}{\ln 2}$

By applying L. Hopital Rule

$$\lim_{n\to\infty} \frac{\ln n / \ln 2}{n/1}$$

$$\lim_{n\to\infty} \frac{\ln n \cdot 1}{\ln 2 \cdot n}$$

$$\lim_{n\to\infty} \frac{\frac{d}{dn}(\ln n)}{\frac{d}{dn}(\ln 2 \cdot n)}$$

$$\lim_{n\to\infty} \frac{\frac{1}{n}(\ln 2)}{\frac{\ln 2}{1} \cdot \frac{dn}{dn}}$$

$$\lim_{n\to\infty} \frac{\frac{\ln 2}{n}}{\frac{\ln 2}{1}}$$

$\Rightarrow 2^n$

```
f(n)
{
if (n==1)
  return 2;
else
  return 2*f(n-1);
}
```

2^n
$\Rightarrow 2^4$
$2 \cdot 2^3$
$\downarrow$
$2 \cdot 2^2$
$\downarrow$
$2 \cdot 2^1$
$\downarrow$
2^1

$T(n) = T(n-1) + 1$ → one operation

Merge Sort: When value of start is less than value of End.

```
ms(A,S,E)
{
  if (S<E)
    mid = (S+E)/2;
    ms(A,S,m)
    ms(A,mid+1,E)
    Merge(A,S,m,E)
}
```


```
Merge(A,S,m,E)
{
  B[S..E]
  i=K=S; j=mid+1;
  while (i<=mid && j<=E)
  {
    if (A[i]<A[j])
      B[K++] = A[i++];
    else
      B[K++] = A[j++]
  while (i<=mid)
  B[K++] = A[i++];
  while (j<=E)
    B[K++] = A[i++];
  for (i=S to E)
    A[i] = B[i];
```

Divide & conquer:

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

↓ for 1st half ↓ for 2nd half ↓ merge

$T(n) = 2T(n/2) + Cn$ ← Cost of selection (Div+merg)

$T(1) = C$ — No. of subproblem

Size of subproblem