Ahsan Habib Sir

TOPIC NAME: Algorithm analysis & Design

Lecture-1

DAY: Sunday

TIME:

DATE: 6 / 8 / 23

Algorithm:

Sequence of steps to perform a specific tasks.

Property:

① Input ② Output ③ Completeness ④ Finiteness

Complexity:

→ Nested loop = $O(n^2)$

→ constant time = $O(1)$

→ N element = $O(n)$

Graph Representation:

① Adj matrix

② Adj list

Graph Algorithm:

Graph $\xrightarrow{\text{Bfs / Dfs}}$ Spanning Tree

Edge Types:

1. Tree edge: New explore
2. Back edge:
3. Forward edge:
4. Cross edge:

Alwan Habib Sir

Lecture-2

TOPIC NAME : Complexity   DAY : Wednesday   TIME :   DATE : 9/8/23

Computer Program:

- Time → main concern
- Space
- Bandwidth

* Assign করলেই time লাগবে ।

```
# int main () {
    int a=0;   → 1
    int b=0;   → 1
    int c= a+b; → 2
                 ____________
                 4 unit of time
```

```
for (int i=0; i<n; i++) { }  → 2n+2
     1,4B     n+1   n
n=5;   Print(" ");
4B
    a=10;  → n
}   4B
                ____
                3n+2
```

$T(n) = 3n+2$

$g(n) = 12B$

TOPIC NAME : DAY : TIME : DATE : / /

```
int main() {
    int n=5;
    for(int i=0; i<n; i++) { }        → 2n+2
        1        n+1   n
        for(int j=0; j<n; j++) { }    → n(2n+2)
            1        n+1   n
```

$\Rightarrow f(n) = 2n^2 + 2n + 2n + 2 = 2n^2 + 4n + 2$

n=10

```
for(int i=0; i<n; i+=2)
```

$\frac{n}{2} + 1 \cong n$

```
for(int i=0; i<n; i=i*i
    int i=1; i<n; i=i*i
    int i=2; i<n; i=i*i, i*2=i
```

$i = 2 = 2^1 = 2^{2^0}$

$i = 4 = 2^2 = 2^{2^1}$

$i = 16 = 2^4 = 2^{2^2}$

$i = 256 = 2^8 = 2^{2^3}$

$\Rightarrow \log_2(\log_2 n)$

$i = 2^0$

$i = 2^1$

$i = 4 = 2^2$

$i = 8 = 2^3$

$i = 16 = 2^4$

$\Rightarrow \log_2$

$i^3 = i * i * i$ } এত complex দরকার নাই।

2^3

2^9

2^{27}

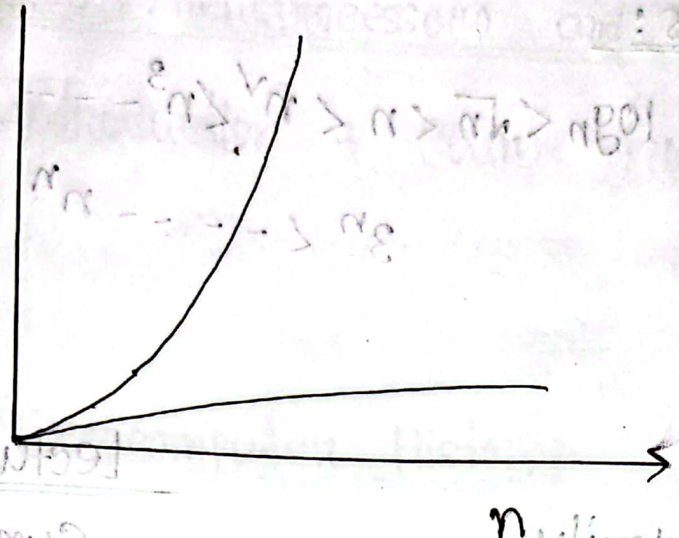
$\Rightarrow \log_3(\ \ )$

$\boxed{}$ $4n+5$ → constant গুলো বাদ যাবে।

n=1	4n	5	4n+5
1	4	5	9
2	8	5	13
3	12	5	17
4	16	5	21
100	400	5	405

$\boxed{}$ $n^2 + \underbrace{4n + 5}_{\times}$ ⟶ n^2 consider করা লাগবে।

n^2	n	4n	5
1	1	4	5
4	2	8	



Cases:

1. Best   2. Average   3. worst → always worst

$\Rightarrow O(1)$ $\Rightarrow O(\frac{n}{2})$ $\Rightarrow O(n)$ case নিয়ে করি।

Asymptotic Notation:

1. Best (Ω)
2. Worst (O)
3. Avg (θ)

$c_1 \cdot g_1(n) < f(n) < c_2 \cdot g_2(n)$

Avg

Ex: $f(n) = 4n+5$ $g(n) = n$ $n = n_0$

$c \cdot g(n) > f(n)$

TOPIC NAME :

DAY :

TIME : DATE : / /

Complexity class:

$1 < \text{-----} < \log n < \sqrt{n} < n < n^2 < n^3 \text{---} < n^{100} < 2^n <$

$3^n < \text{----} n^n$

Lecture-3

Sunday

13/8/23

Ahsan Habib Sir

Complexity

Book: Sartaj Shani

Time complexity:

Algorithm 1.8 (Page-20)

Asymptotic Notation:

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

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

Sum( ) { }

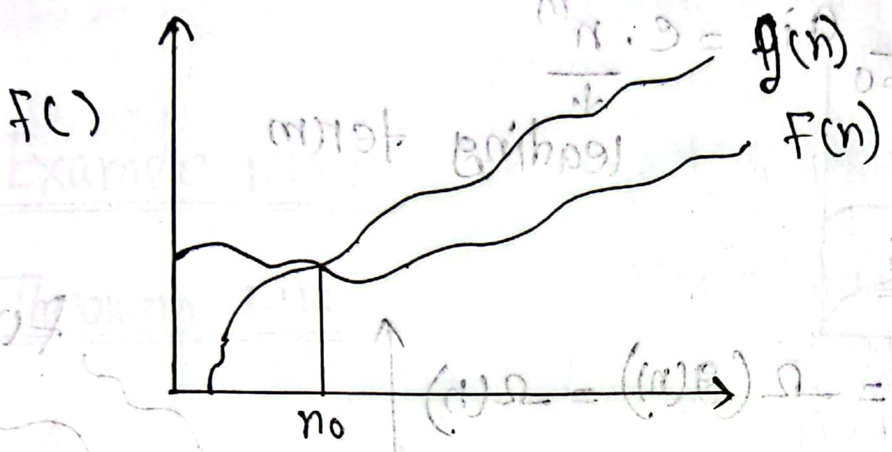
$$f(n) = 3n + 2$$

$c \cdot n \geq 3n + 2 \Rightarrow 4n \geq 3n + 2$

$n \rightarrow$ check করা লাগবে।

$c = 4, n_0 = 1$

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



<u>Example 1.11 :</u> $f(n) = 100n + 6 \rightarrow O(n)$ হবে কী না, হলে ও কার মত) ?

$100n + 6 \le c \cdot n$

$\Rightarrow c = 101$

$100n + 6 \le 101 \cdot n$

$n \ge 6$, $c = 101$ হয়

$\therefore f(n) = O(n)$

* g এর value minimum নিব।
→ leading term.

<u>Theorem 1.2 :</u> $f(n) = a_m n^m + \cdots + a_1 n + a_0$

$$= \sum_{i=0}^{m} a_i n^i$$

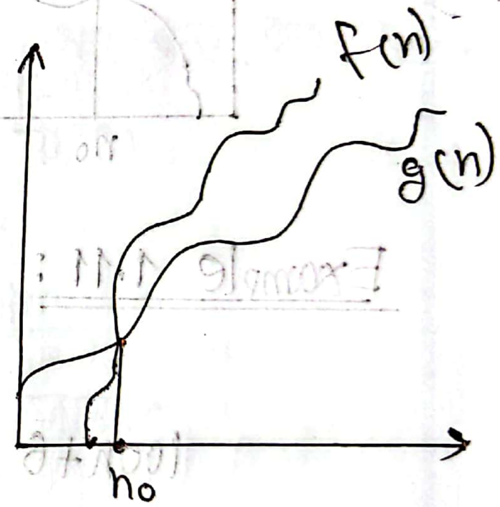
TOPIC NAME : DAY : TIME : DATE : / /

$$= n^m \sum_{i=0}^{m} a_i n^{i-m}$$ → বড়টা consider করো।

$$= n^m \sum_{i=0}^{m} a_i = c \cdot \underbrace{n^m}_{\text{leading term}}$$

Omega Ω :

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



$c * g(n) \le f(n)$

$$3n \le 3n+2$$

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

* অনেকগুলো থেকে বড়টা নিও।

Example-1.12 :

Theorem 1.3 :

Theta Θ :

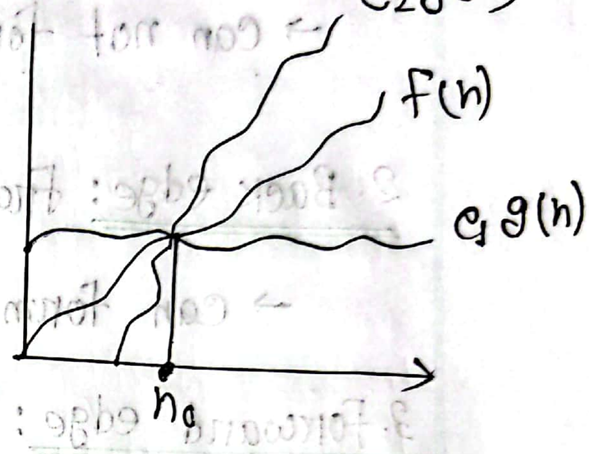
→ Precise value provide করে।

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

TOPIC NAME : DAY : TIME : DATE : / /

⊛ $c_1 g(n) \le f(n) \le c_2 g(n)$ $f(n) = 3n+2$

$2n \le 3n+2 \le 4n$



Example 1.13 :

Theorem 1.4 :

Little O : $\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0$

Example 1.14

Little Ω : $\lim_{n \to \infty} \frac{g(n)}{f(n)} = 0$

Ahsan Habib sir

Lecture-4

TOPIC NAME : DFS Edges

DAY : Sunday

TIME :

DATE : 20/8/23

DFS kind of edges :

1. Tree edge : encounter new vertex.
   → can not form cycle.

2. Back edge : from descendent to ancestor
   → can form cycle.

3. Forward edge : from ancestor to descendent.

4. Cross edge : between a tree or subtree.

BFS Traversal application :

1. shortest path
2. Connected components.
   → কয়বার bfs call হচ্ছে, তাই।

→ Two color Problem.

DFS application :

1. Finding cycle.
2. Topological sort
   → Node এর linear order.
3. SCC

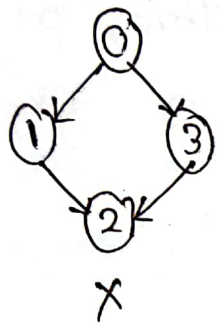
TOPIC NAME :

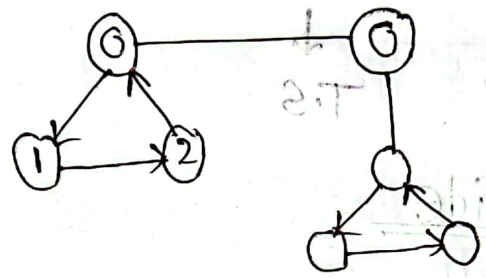
DAY :
TIME : DATE : / /

Topological sort : (slide)

→ Indegree algorithm → Time descending algorithm

Strongly connected components :





Ahsan Habib Sir

Critical Path

Lecture-5
Tuesday
22/8/23

Book : Algorithms unlocked

Critical path in a PERT chart : Page-81

⇒ Maximum time = critical path

⇒ যার indeg 0 ও যার outdeg 0, তাদের মধ্যবর্তী Summation of time = critical path

TOPIC NAME : DAY : TIME : DATE : / /

⇒ maximum path = minimum time

Shortest Path in a directed acyclic graph :

DAG
↓
T.S

SCC → slide

Lecture-6
Sunday
27/8/23

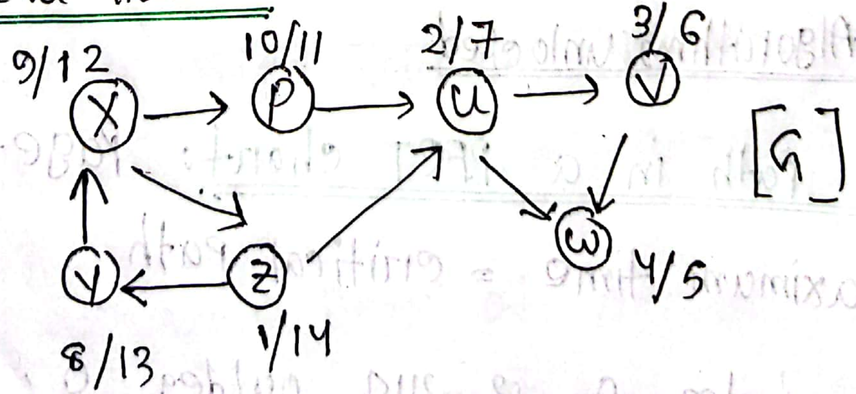
Ahsan Habib Sir

Network Flow

→ Algorithm Design

Book : Eva Trados

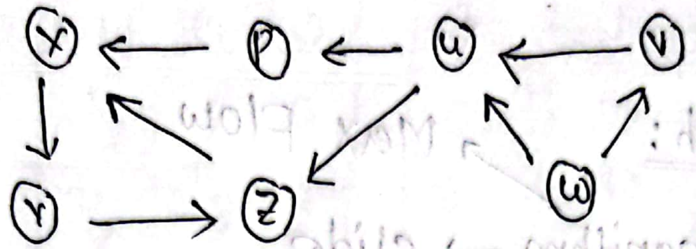
SCC :
↓
slide



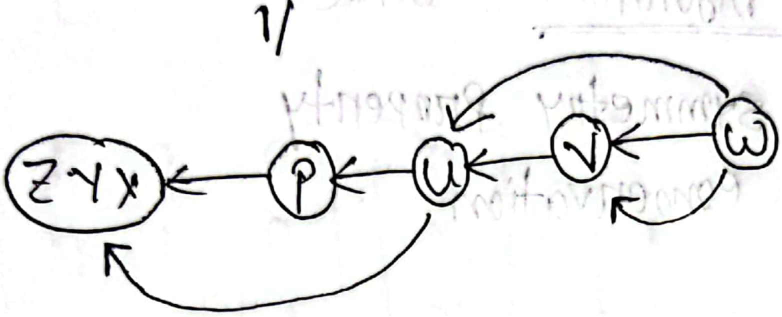
Z Y X P u V ω

G^T =

TOPIC NAME : DAY : TIME : DATE : / /

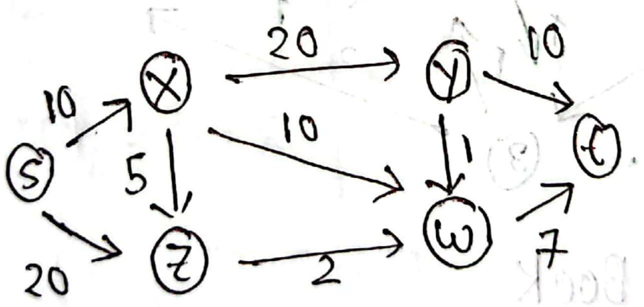


[G^T]



[Acyclic component]

Network Flow



* Problem : Maximum flow source to Destination.

Flow Network : Traffic flow + given capacity

Definition - Page 338 + 339