L-11

01.12.19

Prime Number (*** আসতে পারে ***)

2 → only one even prime number exist

* There are infinitely many Primes.

__Proof:__ Let, There are numbers of prime number.

$P_1, P_2, P_3 ---- P_n$

$P = P_1 \times P_2 \times P_3 \times ---- \times P_n + 1$

$= 2 \times 3 \times 5 \times 7 + 1 = 211$

211 % 2 = 1

211 % 3 = 1

211 % 5 = 1

211 % 7 = 1

Let, $P \% P_m = 1 \; [1 \le P_m \le P_n]$

$\therefore$ P is a prime number.

So, Thats why there is infinitely prime Number.

L-4

GCD → Greatest Common Divisor

LCM → Lowest Common Multiplayer

24 = 1, 2, 3, 4, 6, 8, 12, 24

18 = 1, 2, 3, 6, 9, 18

CD (24, 18) = 1, 2, 3, 6

GCD (24, 18) = 6

CM (24, 18) = 144, 288, 432

LCM (24, 18) = 144

$$LCM(a, b) = \frac{a \times b}{GCD(a, b)}$$

$$LCM(24, 18) = \frac{24 \times 18}{6}$$

$$= 72$$

Prime Factorization

$24 = 2^3 \times 3^1$

$18 = 2^1 \times 3^2$

$\frac{24}{2} = \frac{12}{2} = \frac{6}{2} = \frac{3}{3} = 1$

$\frac{18}{2} = \frac{9}{3} = \frac{3}{3} = 1$

L-11

Prime Fectorization

GCD (24, 18) = $2^1 \times 3^1$ → minimum power should be taken
= 6

LCM (24, 18) = $2^3 \times 3^2$ → maximum ----
= 72

* 65, 39, 130

$65 = 5^1 \times 13^1 \times 2^0$

$39 = 3^1 \times 13^1$

$130 = 2^1 \times 5^1 \times 13^1$

$\frac{65}{13} = \frac{5}{5} = 1$

$\frac{39}{13} = \frac{3}{3} = 1$

$\frac{130}{13} = \frac{10}{5} = \frac{2}{2} = 1$

GCD (65, 13, 130) = $2^0 \times 3^0 \times 5^0 \times 13^1 = 13$

LCM (65, 13, 130) = $2^1 \times 5^1 \times 3^1 \times 13^1 = 390$

L-11

Euclid - GCD (a, b)

if (b > a) return Euclid - GCD (b, a)

if (b == 0) return a

else return Euclid GCD (b, a % b)

---

*GCD (18, 24)

step-1

a = 18

b = 24

step-2

a = 24

b = 18

step-3

a = 18

b = $\cancel{\frac{24}{18}} = 6$

b = 24 % 18 = 6

step-4

a = 6

b = $\cancel{\frac{18}{6}}$ 18 % 6 = 0

* GCD (733, 855)

step-1

a = 733

b = 855

step-2

a = 855

b = 733

step-3

a = 733

b = $\cancel{\frac{855}{733}}$ 855 % 733

b = 122

step-4

a = 122

b = 733 % 122 = 1

step-5

a = 1

b = 122 % 1 = 0

step-6

GCD = 1

L-11

Relatively Prime

GCD(a, b) = 1

L-13

08.12.19

Heap

1	2	3	-3	6	8	-6

$O(n)$

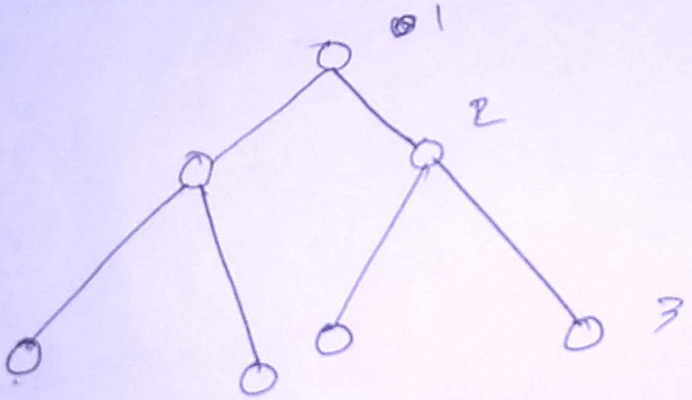
$\boxed{n \to \text{node}}$

$n \to \log_2 n$

(i) Insert $\to O(\log_2 n)$

(ii) Find $\to O(1)$

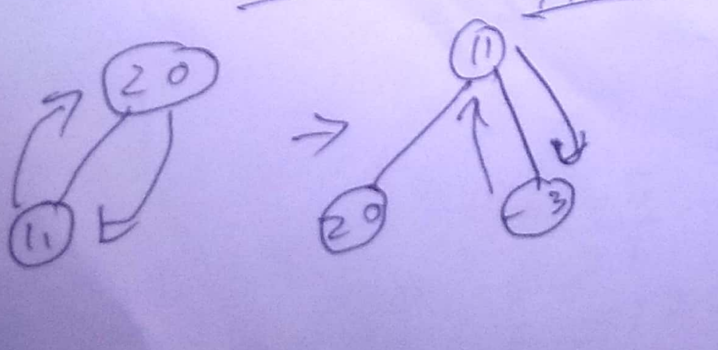
(iii) Delete $\to O(\log_2 n)$



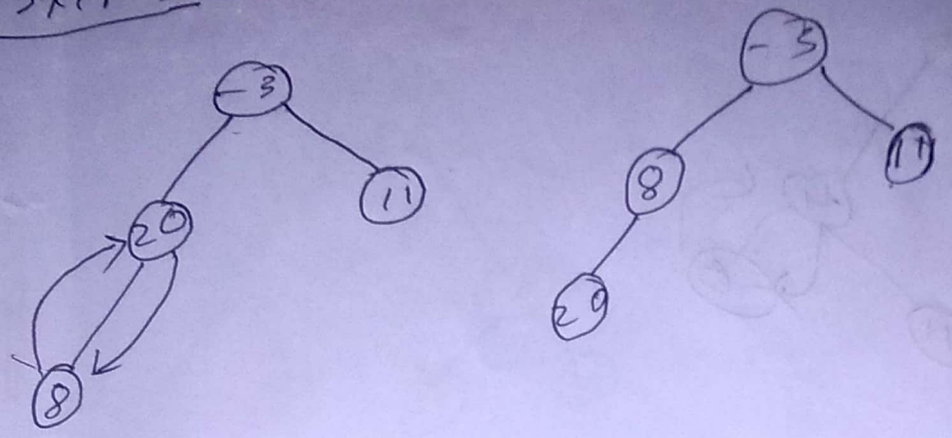
20	11	-3	8	7	4	-6	[illegible]

Step-1

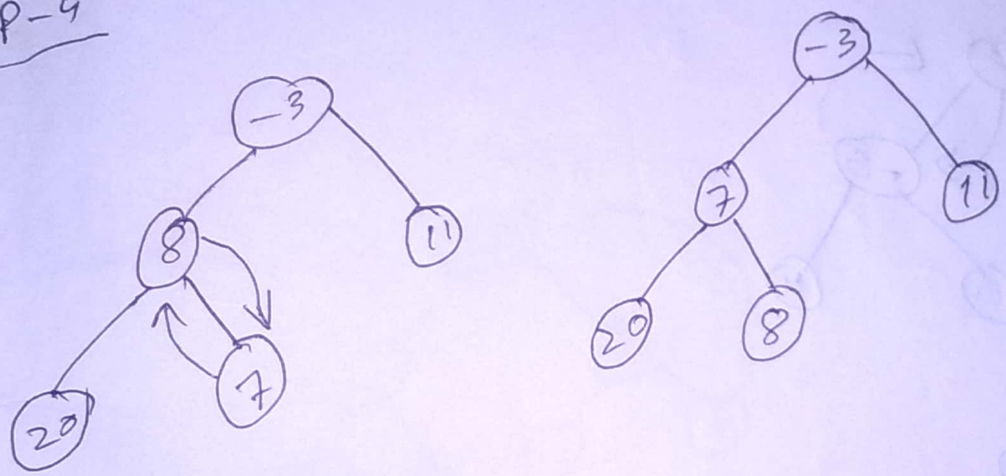
step-2



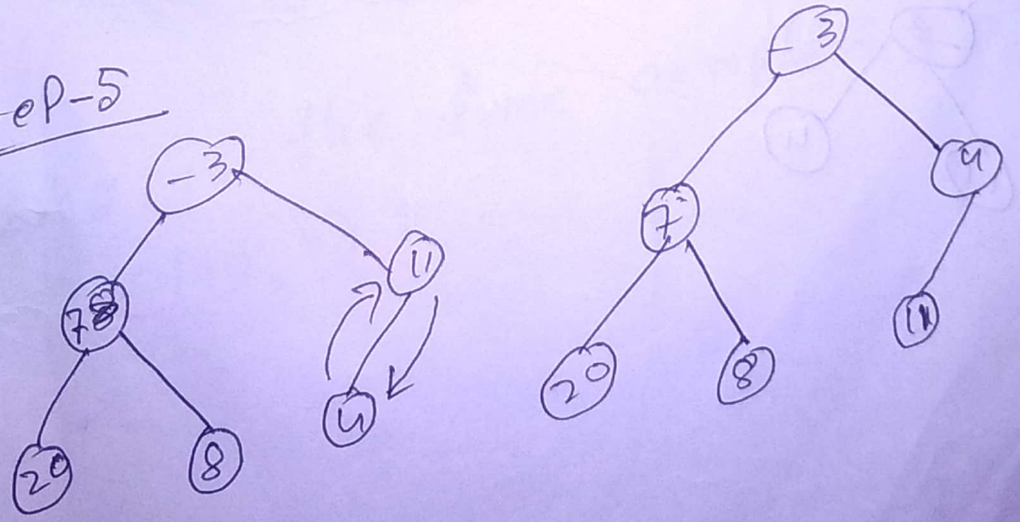
Step-3



Step-4

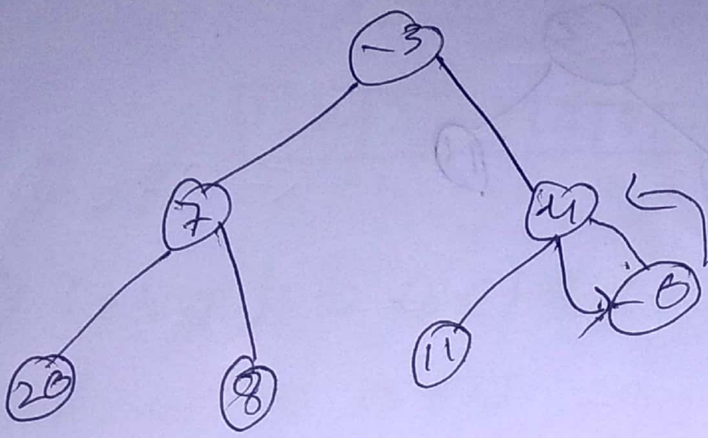


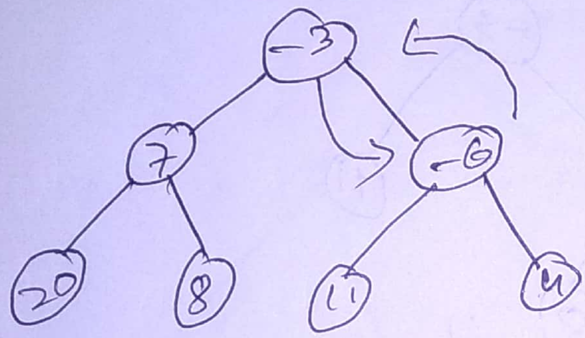
Step-5

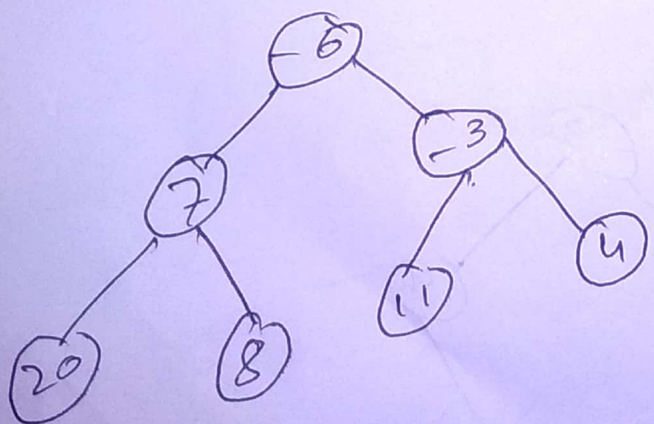


Step - 6

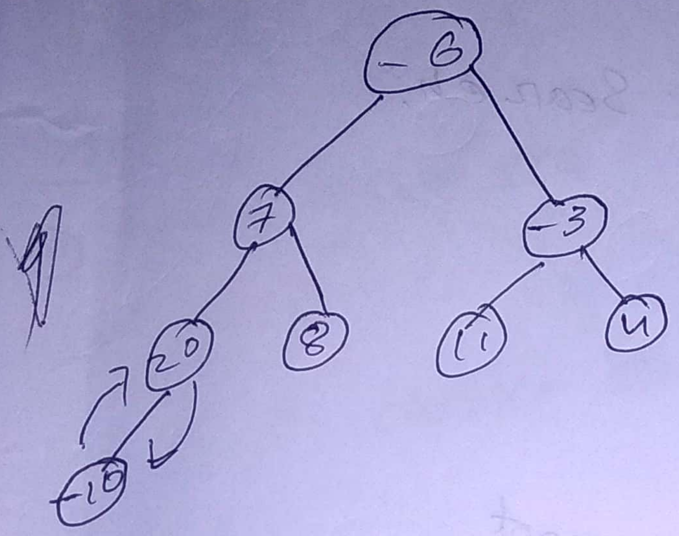
L - 12

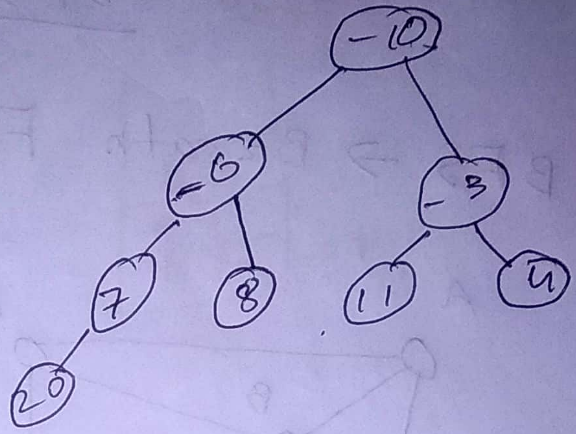




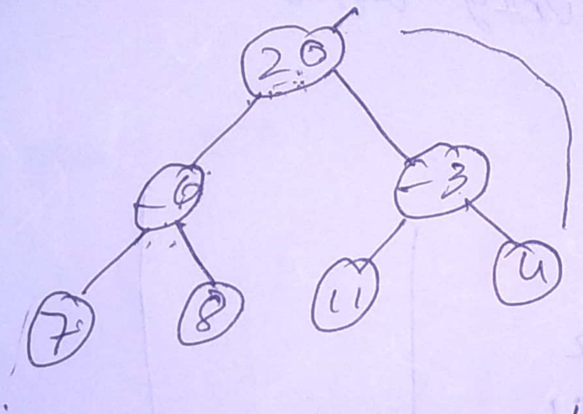


20	11	-3	8	7	4	-6	-10





Delete!



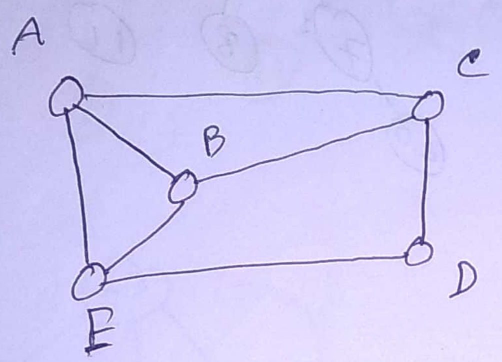
Why is the time complexity of Heap is $O(\log_2 n)$?

10-12-19

L-13

BFS

- BFS → Breath First Search:



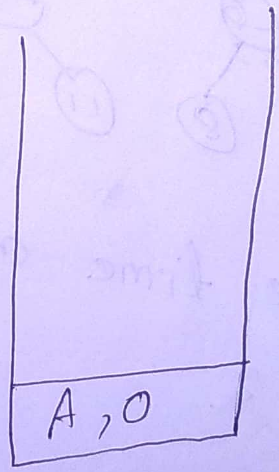
Rule: ① cost of every ages (root) is same.

Ⓘ FIFO

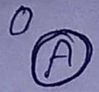
Tree:

① Push করবে color change করতে হবে।

- array
- Black



Step-1



E, 1
C, 1
B, 1

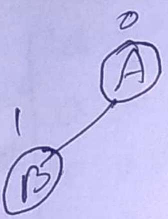
→ pop

Step-2

(ii) Black করতে হবে

A হলো।

E, C, B change হবে।

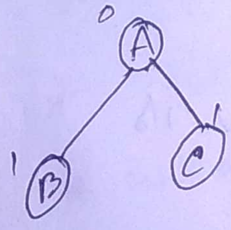


D - 2
E - 1
C - 1

- pop

Step-3

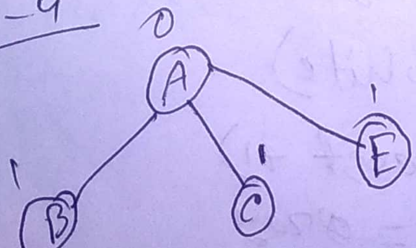
D gray হবে।



D - 2
E - 1

- pop

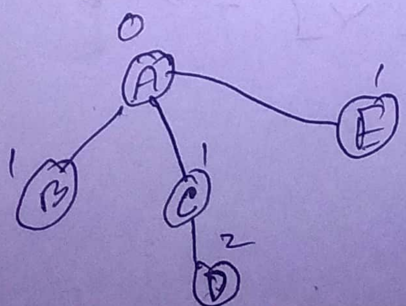
Step-4



D - 2

pop

Step-5



L-13

O(V+E)

```
BFS (G, S)
{
  color [ ] = white
  distance [ ] = 0
  Queue Q
  Q push (S, 0)
  color [S] = gray
  while (Q = empty)
  {
    u = Q. Front;
      Q .pop
    distance [u] = u. cost
    color [u] = Black.
    for (all v ∈ Graph ~~&~~ & & v is
        connected with u)
    if (color [v] = = white)
        Q. push (v, u cost +1)
          color [v] = gray
  }
```

L-13

Haffman coding: $O(\log_2 n)$

→ linked List

* Set / Heap

* STL

BTXE	E
B	EX
BT	EXT
BTX	EXTB
BTXE	

prefix ~~code~~

abc	abc
ax	a
edf	bx
gh	b

00 11

10 0011

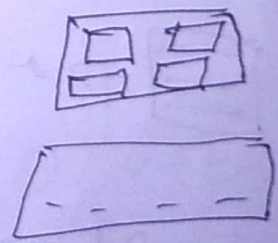
000

prefix code

* what is prefix ?

* what is prefix code ?

* what is the time complixity of Halfman coding ?



a = 0011

b = 0

c = 1

d = 011

A = 12

a = 13

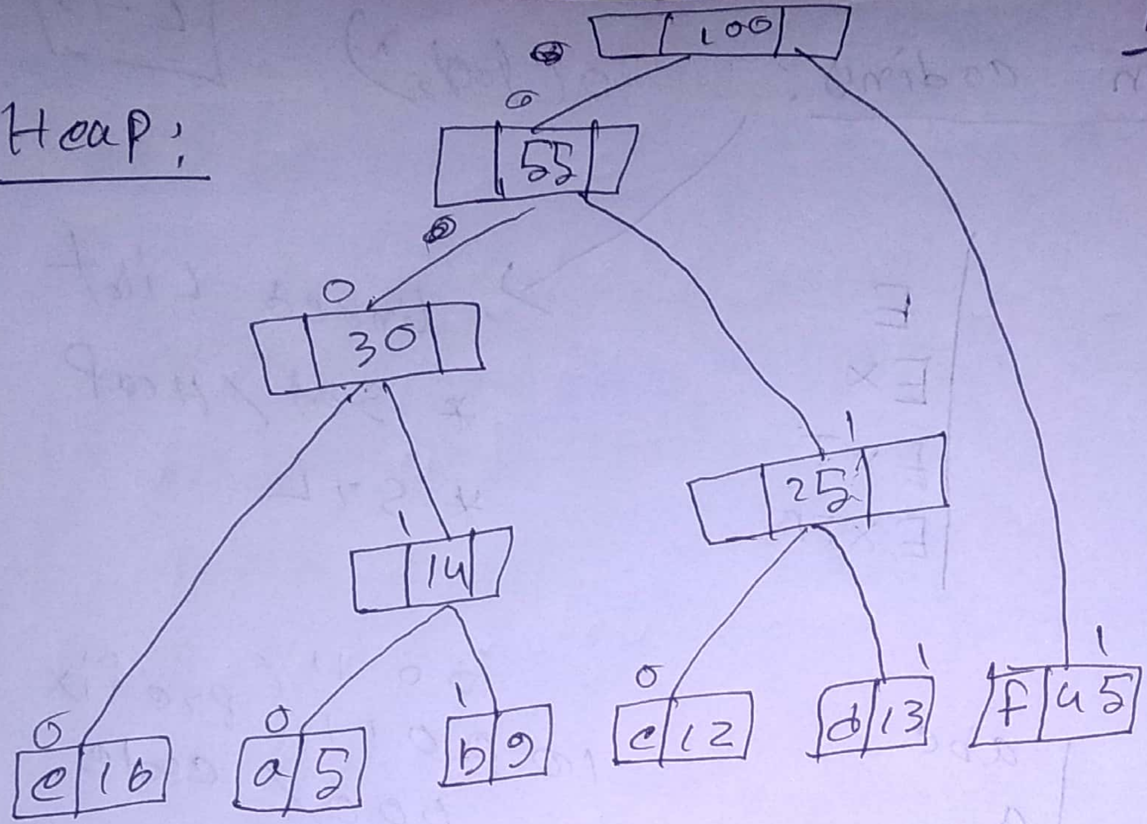
; = 10

= 13

3 = 3

c = 8

Heap:



e = 000   c = 010

a = 0010   d = 011

b = 0011   f = 1

acd = $\underline{0010}$ $\underline{010}$ $\underline{011}$

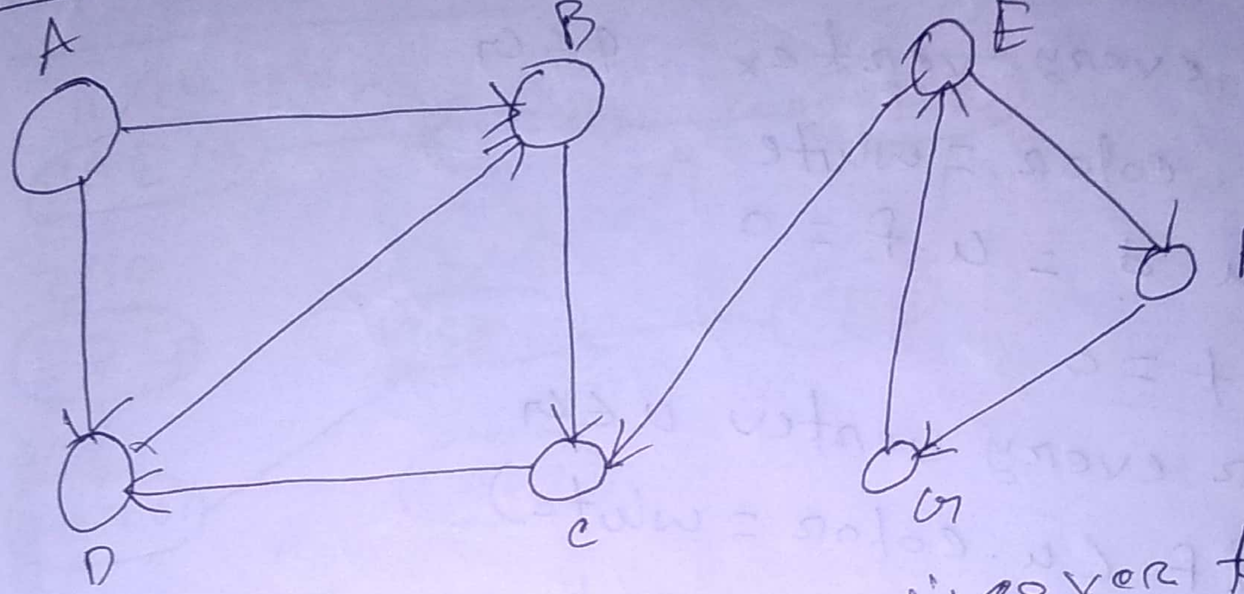
→ a = 12

Haffman coding দিয়ে Binary ~~bet~~ থেকে ছোট হবে।

$\boxed{a}$ zip

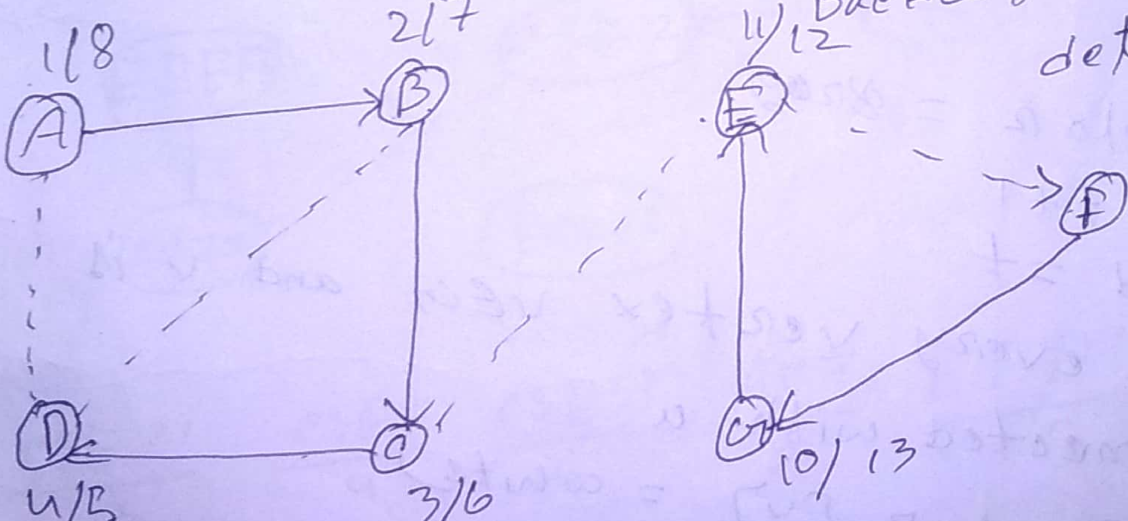
15.12.19

DFS : Depth first Search



start discover time < end discover time = forword edge.

Backedge - loop/circle detect করে?



$O(V+E)$

Algorithm

DFS (G)

for every vertex $q \in G$

u. color = white

u. d = u. f = 0

t = 0

for every vertex $u \in G$

if (u. color = white)

DFS - visit (u)

DFS_ visit (G, u)

{

u. color = gray

t = t+1

u.d = t

for every vertex $v \in G$ and v is connected with u.

if (color [v] = white)

DFS _visit (G, v)

t = t+1

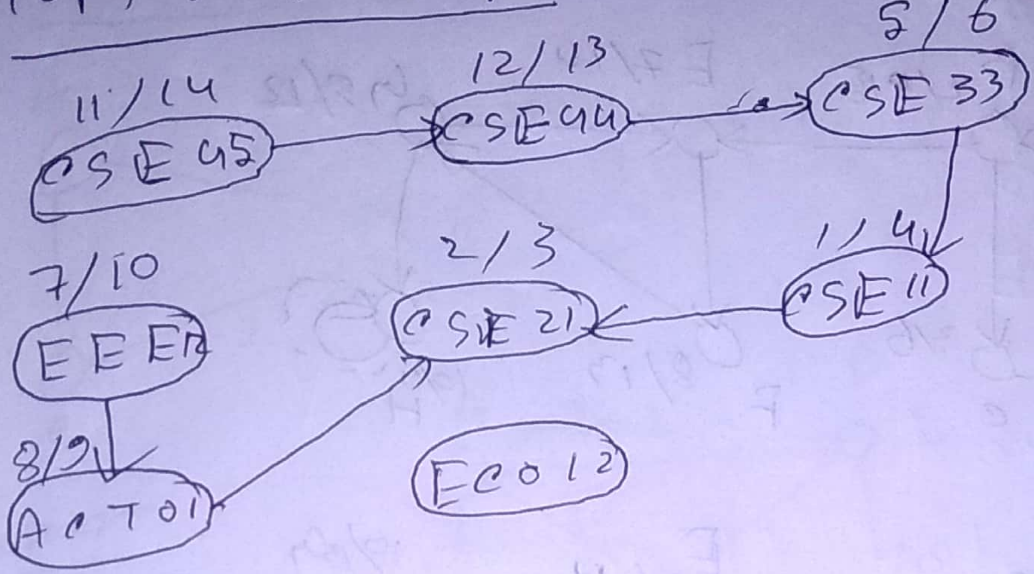
u . f = t

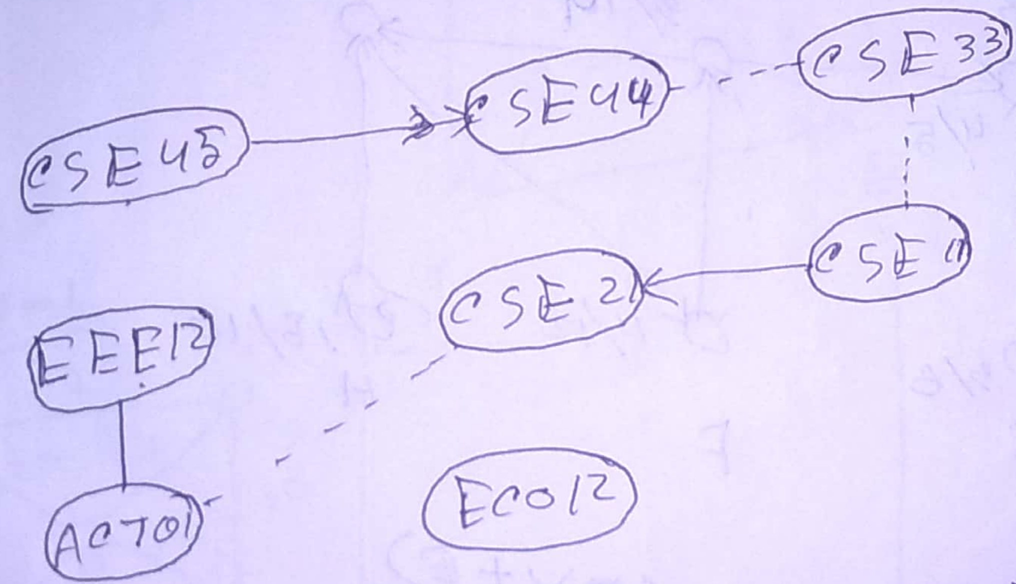
u . color = Black.

}

22.12.19

Topological Graph:





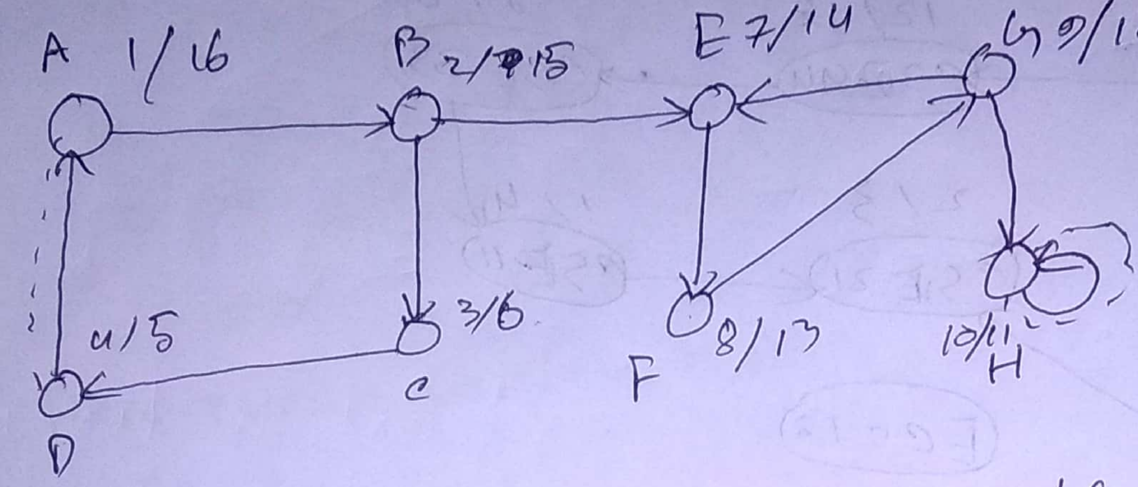
$\frac{\text{CSE 21}}{3}$ $\frac{\text{CSE 11}}{4}$ $\frac{\text{CSE33}}{6}$ $\frac{\text{ACT}}{9}$ $\frac{\text{EEE12}}{10}$

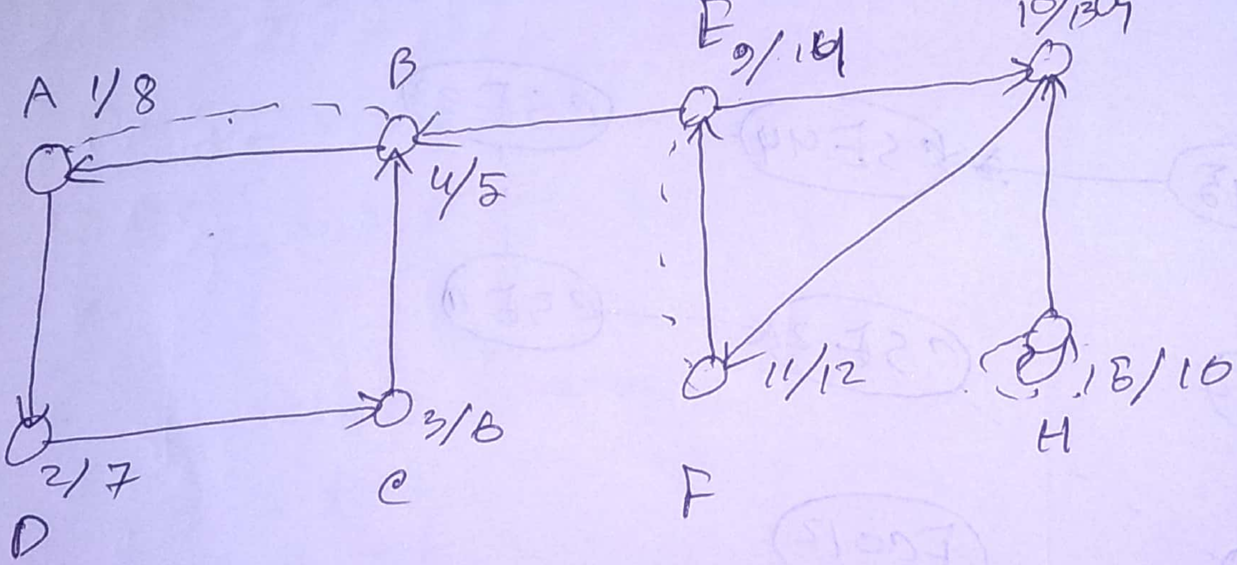
$\frac{\text{CSE 44}}{13}$ $\frac{\text{CSE 45}}{14}$ $\frac{\text{ECO12}}{16}$

If backedge found in DFS. Answer is not valied.

L-15

Strongly Connected Component:



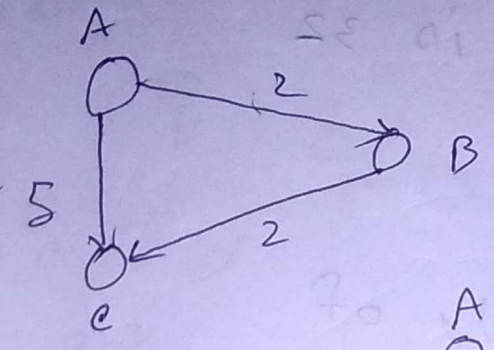


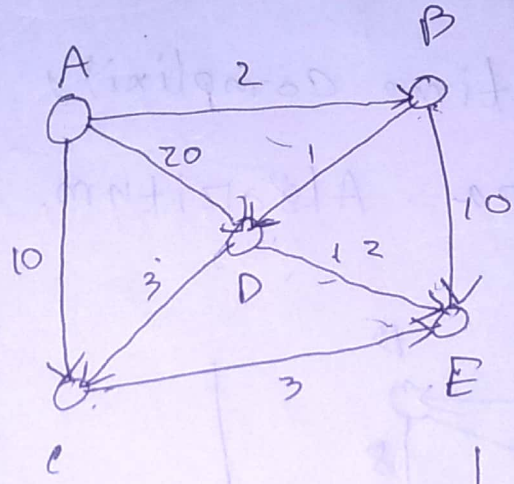
$2 \times (V + E)$

$= V + E$

L-16 24.12.19

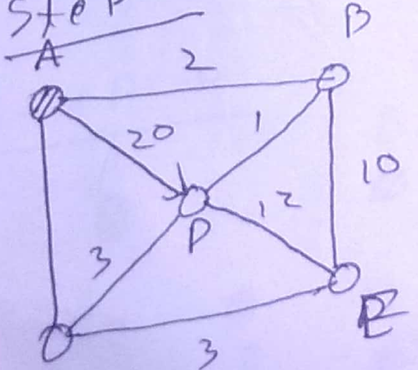
Diykstra Algorithm



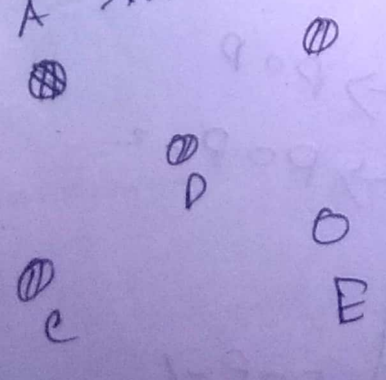


~~Step-1~~

Step-1

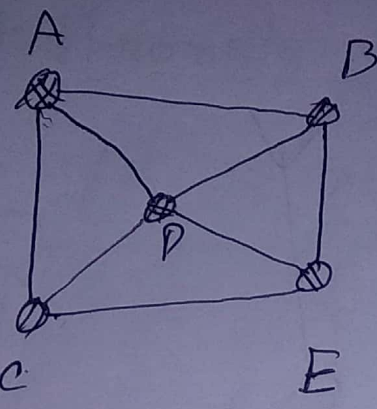


Step-2



D-E, 12	→ pop 3
D-C, 3	
A-C, 10	
A-B, 2	→ pop 2
✗ A-D, 20	
A, 0	→ pop 1

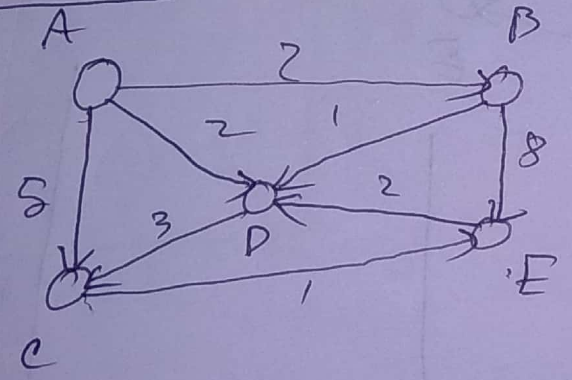
Step-3



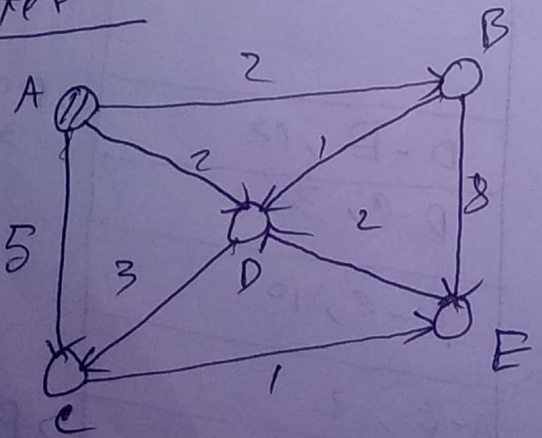
Maximum is 32

what is the time complixity of ~~Dest~~ Diykstra Algorithm.

Practice;

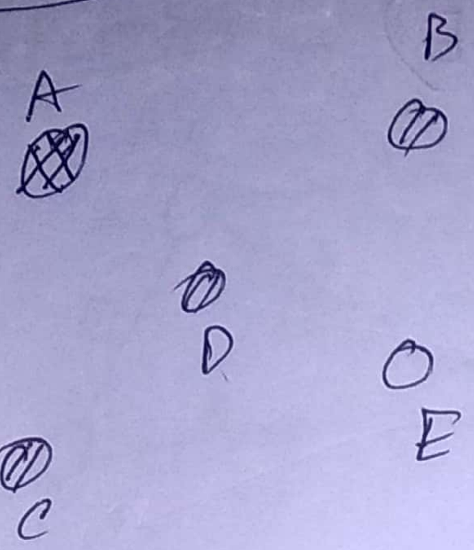


Step-1

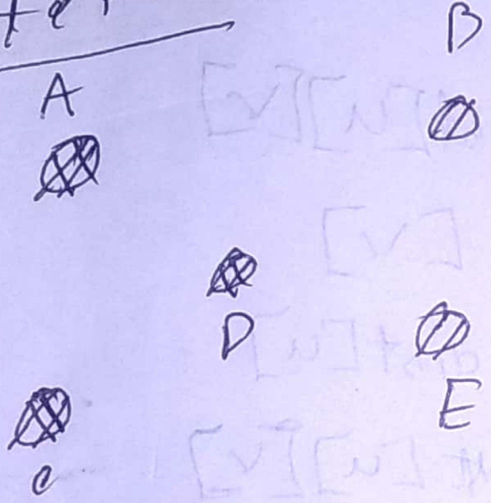


E-B, 8
C-E, 1
D-C, 3 → pop
A-D, 2 → pop-2
A-C, 5
A-B, 2
A, 0 pop-1

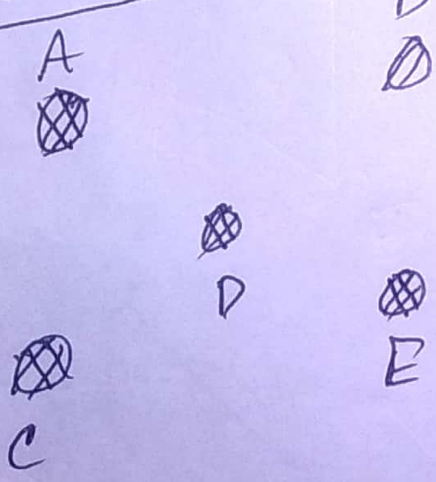
Step-2



Step-3



Step-4

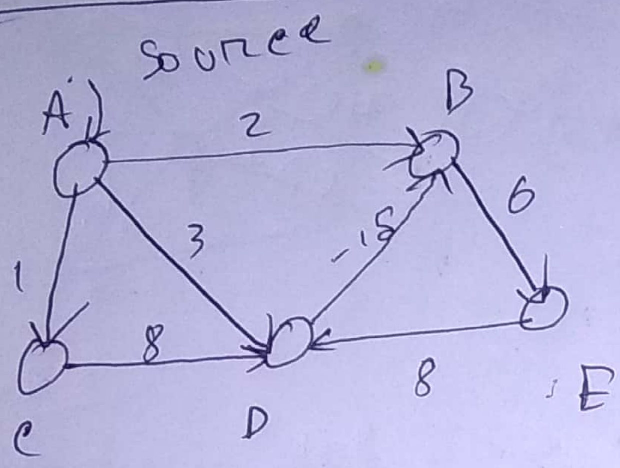


Minimum is 6

L-17

Bellman-Ford

$O(VE)$

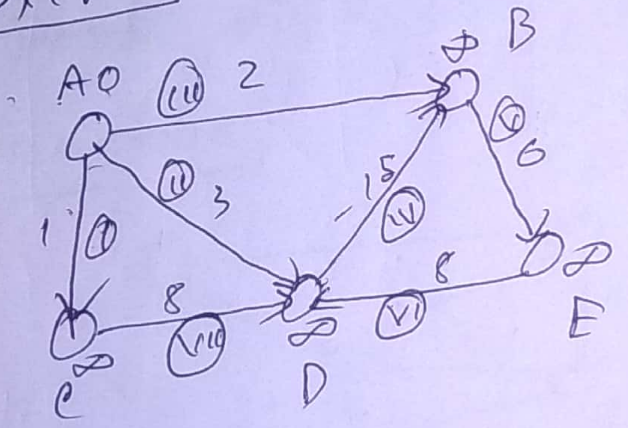


Relax

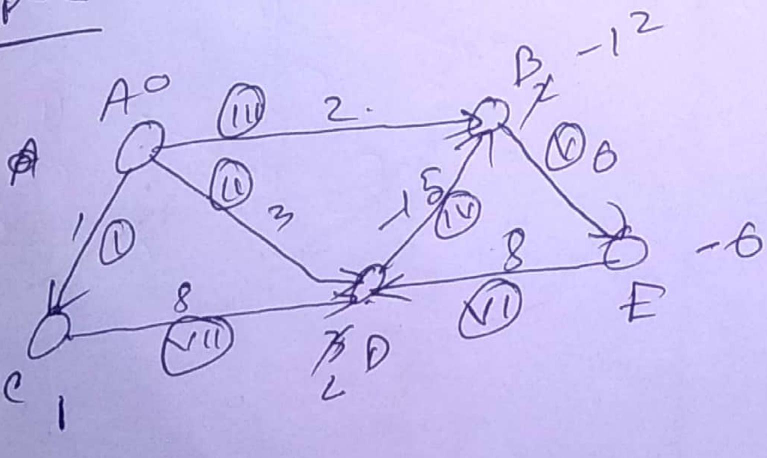
$dist[u] + cost[u][v] < dist[v]$

$dist[v] \Leftarrow dist[u] + cost[u][v]$

step-1

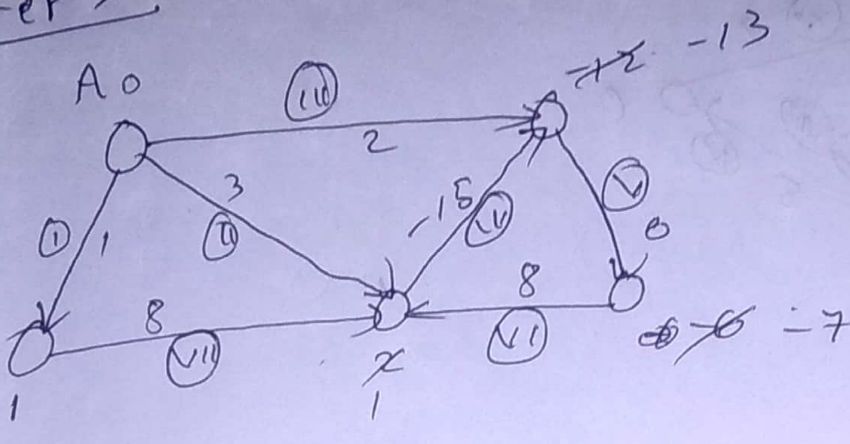


step-2

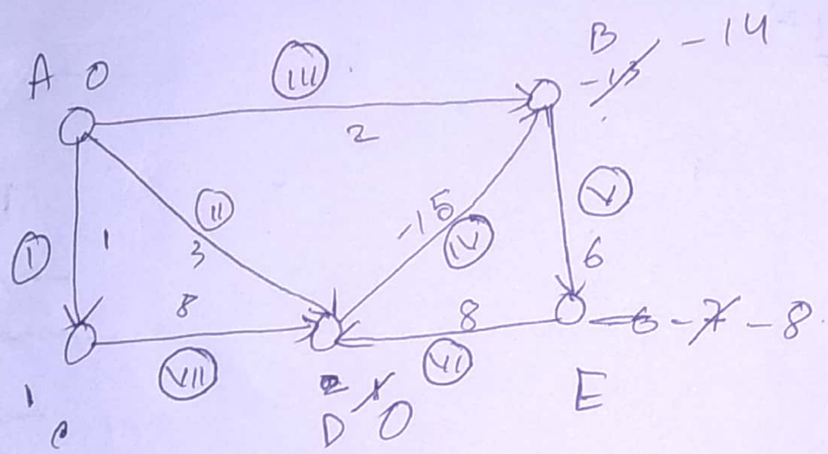


L-17

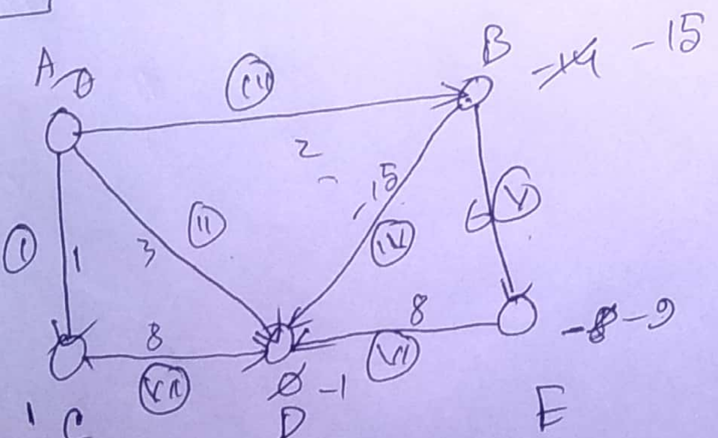
Step-3



Step-4

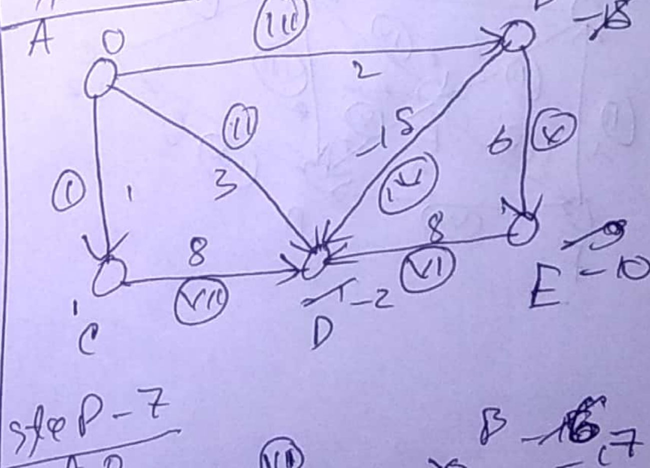


Step-5

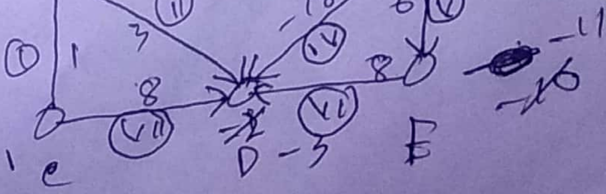


There is an infinity loop.

Step-6

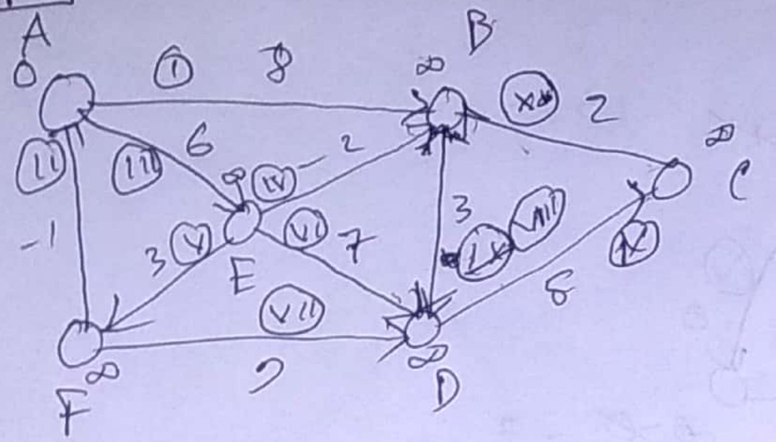


Step-7

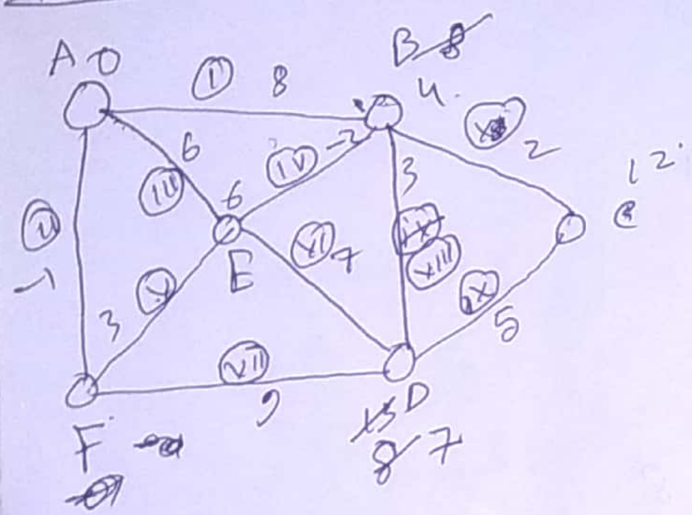


L-17

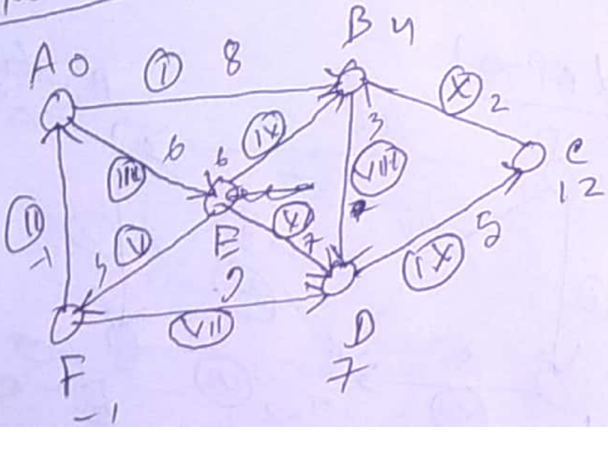
Step-1



Step-2



Step-3



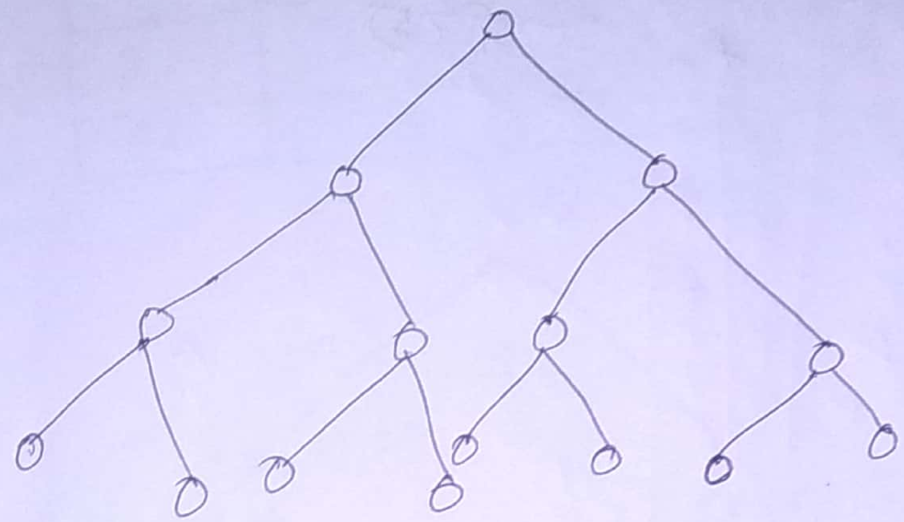
L-18

05.01.2020

Dynamic Programming:

(1+2) +(5+4) +(2+1)

(i) overlapping subproblem



(ii) optimal Substructure:

0 1 1 2 3 5 8

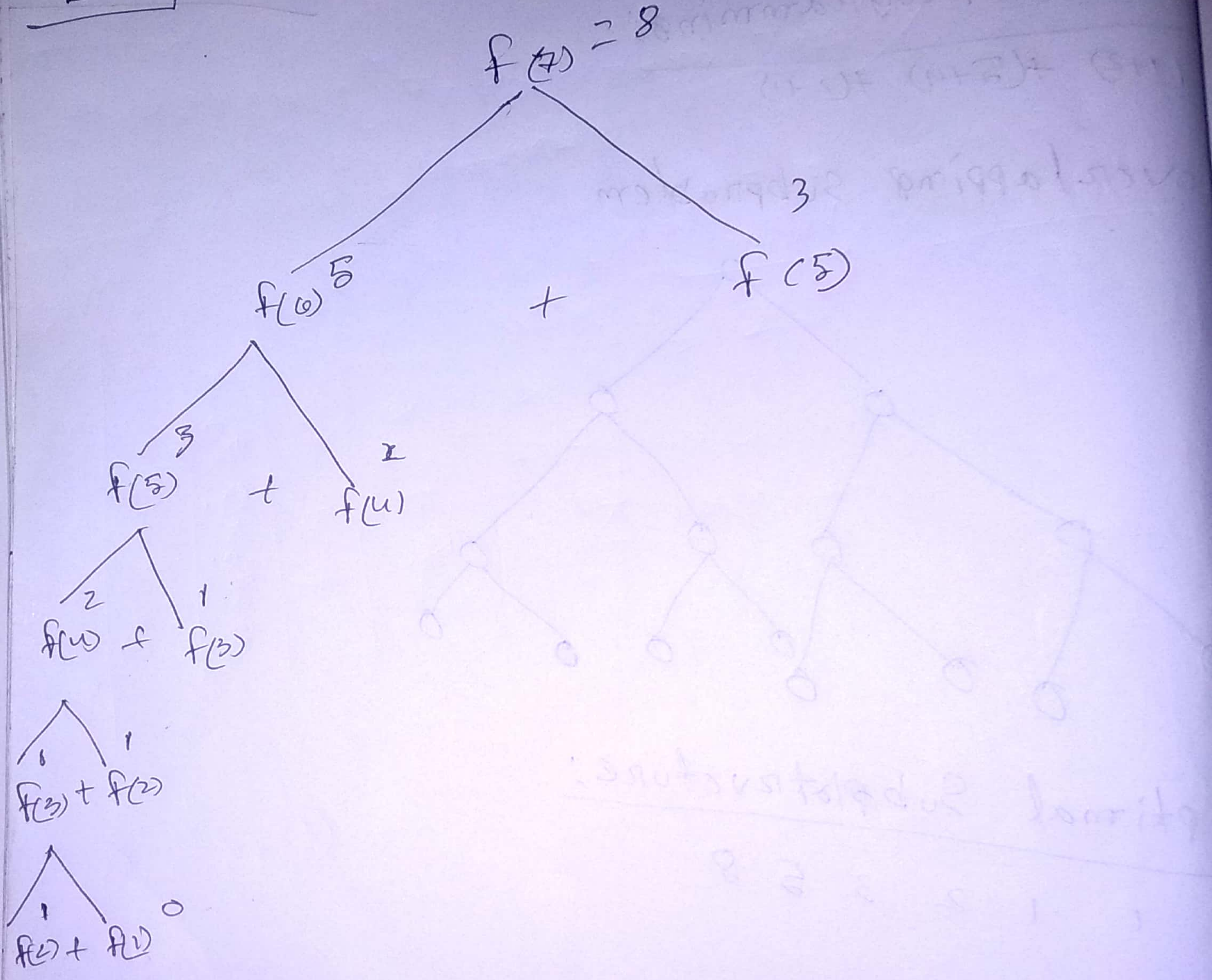
$f(1) = 0$

$f(2) = 1$

$f(n) = f(n-1) + f(n-2) \quad n \geq 3$

Scanned by CamScanner

L-13



L-18

0/1 Knapsack

max weight = 7

Item	weight	cost
A	5	20
B	3	10
C	3	11

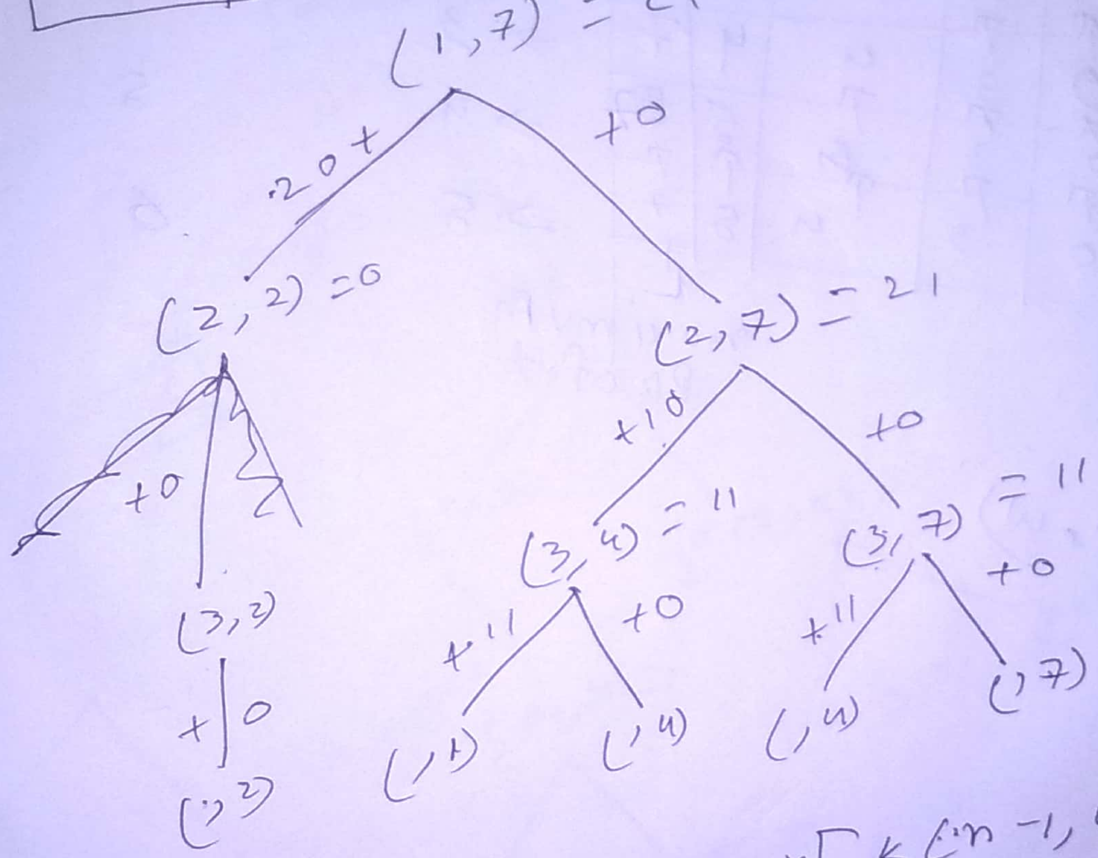
5/20 3/10 3/11 8/30 8/31

{A}, {B}, {C} {A, B} {A, C}

{B, C} {A, B, C} {Ø}

6/21 11/41 0/0

✓ × ×



max[K(n − 1, w − w[n]) + P[i],

K(n+1, w)]

$K(n, 0) = 0$

$K(0, w) = 0$

$K(n, w) =$

K

L-19

07.01.2020

Dynamic Program Table

0/1 knapsack

$n = 4$ $\quad$ $W = 5$

	weight	profit
→	2	3
→	3	4
→	4	5
→	5	6

n \ w	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	3	3	3	3
2	0	0	3	4	4	7
3	0	0	3	4	5	7
4	0	0	3	4	5	7

maximum profit

O (nw)

L-20

09.01.2020

~~Lo~~ Longest Common Subsequence

acx

{a}, {c}, {x} {a, c}

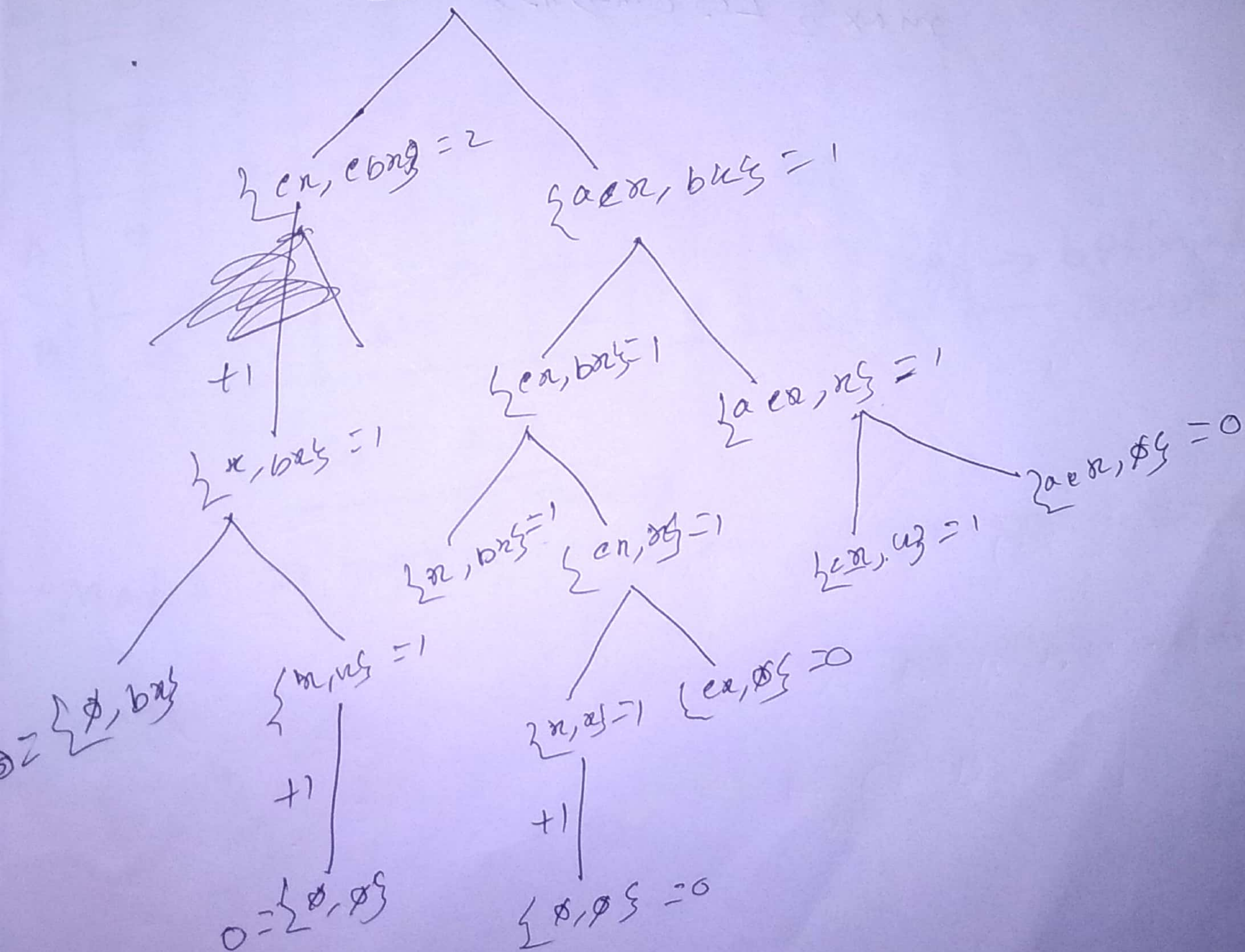
{a, x} {c, x} {a, c, x} {∅}

cbx

{c} {b} {x} {c, b} {c, x}

{b, x} {c, b, x} {∅}

{acx, cbx} = 2



$LCS(m, \phi) = 0$

$LCS(\phi, n) = 0$

$LCS(m, n) =$

if $(m[0] == n[0])$

$1 + LCS(m-1, n-1)$

else

$\max\{LCS(m-1, n), LCS(m, n-1)\}$

Scanned by CamScanner

Strt 1 = AGGTAB

Start 2 = GXTXAYB

	0	G	X	T	X	A	Y	B
0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0+1 = 1	1	1
G	0	0+1 = 1	1	1	1	1	1	1
G	0	0+1 = 1	1	1	1	1	1	1
T	0	1	1	1+1 = 2	2	2	2	2
A	0	1	1	2	2	2+1 = 3	3	3
B	0	1	1	2	2	3	3	3+1 = 4

→ optimal solution

GTAB

উত্তর

Match না করলে Row, col থেকে বড়টা নিবে।

Match করলে কর্ণ বরাবর উপরের ঘরের মান আসলেটা নিয়ে +1.