DAA

* Sum of 'n' natural numbers

I/p : n = 3
O/p : 6 // 1+2+3

I/p : n = 5
O/p : 15 // 1+2+3+4+5

Write down fun^n yourself & compare these solution. Discuss which is more efficient.

Sol^n

① n(n+1)/2

② sum = 0
for (i = 1; i < n+1)
sum = sum + i
return sum

③ sum = 0
for (i = 1; i < n+1)
for (j = 1; j+1)
sum = sum + 1
return sum

→ Time taken by program is dependent on machine you running

→ programing Language 'c' takes less time because in python interpretion happen relatably slow

→ load on machine

To Overcome this problem we have Asymptotic Analysis (Theoritical Analysis)

Scanned with OKEN Scanner

Asymptotic Analysis (Theoritical Analysis)

- → No depending on machine, programming Language
- → we do not have to implement all ideas Algorithm
- → It's about measuring order of growth in terms of input size

Order of Growth

(1) fun1() → C_1 → constants of all operation (* + /)

(2) fun2() → $C_2 n + C_3$ → n times in for loop & constants initialization

(3) fun3() → $C_4 n^2 + C_5 n + C_6$ → n^2 execution time, n times & constants
(factor)

Algorithm (ex: Tea making)

↳ Set of steps to complete task

or

Sequence of instructions to solve computational problem

Worst, Average and Best Case Analysis of Algorithm

1) Worst Case Analysis (Mostly used)

- we calculate upper bound on running time of Algo. (max no. of operation to be executed)
- for linear search, worst case is when element to be searched (x) is not present in the array. (compares one by one)
- most commonly used analysis of Algorithm.

2) Best case Analysis (Rarely used)

- we calculate lower bound on running time of Algo (min no. of operation to be executed)
- for linear search, best case occurs when (x) is present at first location. No. of operations in best case is constant.

3) Average case Analysis (Rarely used)

- we take all possible i/p & calculate the computing for all i/p.
- for linear search 'x' is not present in array

Properties of Algorithm

1) Input = 0 or more inputs → Home work

2) Output = atleast one output

3) Unambiguous =

4) Finiteness

How to Write Algorithm

Step 1 : Start

Step 2 : Read a, b or input a, b

Step 3 : Sum = a+b

Step 4 : Print Sum

Step 5 : Stop

Scanned with OKEN Scanner

Asymptotic

→ mathematical way of supposing time complexity

i) Big O (upper bound (at most) worst case)

$$\boxed{f(n) = O\,g(n)}$$

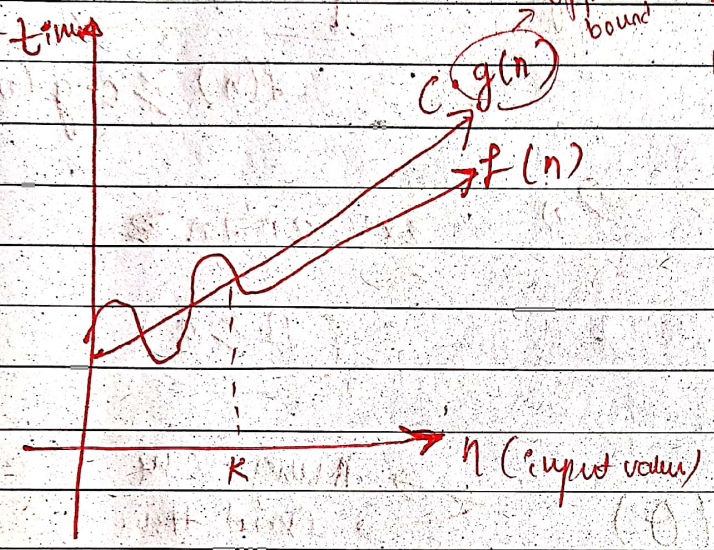
given → Algorithm; upper bound of f(n)

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

$c > 0$ constant

$n \ge K$

$K \ge 0$



Ex: $f(n) = 2n^2 + n$ → time taken to execute

$f(n) = O(?)$

$2n^2 + n \le c \cdot g(n^2)$

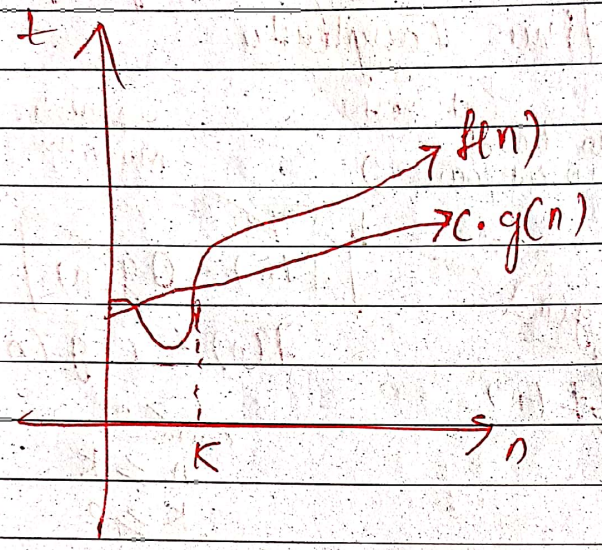
↳ upper bound

find out at most value

for $n \ge 1$ always hold

$2n^2 + n$ can be written as $O(n^2)$

2) Big - Omega (Ω) → Best case → Lower bound. (at least)



our Algorithm

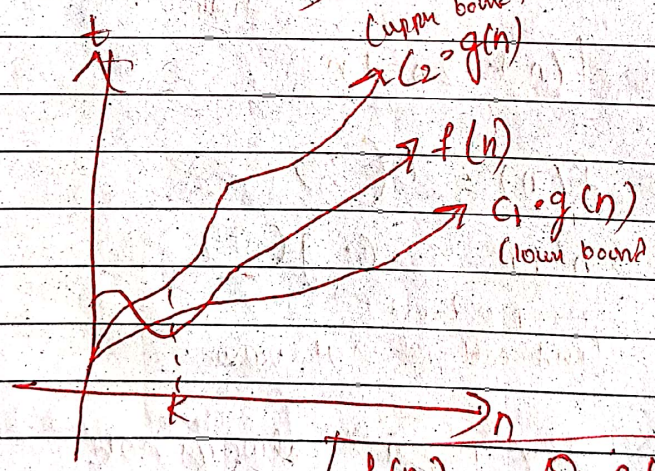
$$\boxed{f(n) = \Omega\, g(n)}$$

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

Ex: $2n^2 + n$

$n \geq 0$

3) Theta (Θ) → Average case → Exact time



$$\boxed{f(n) = \Theta\, g(n)}$$

$$c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$$

Ex: $\underset{\text{constant}}{2} \cdot n^2 \leq 2n^2 + n \leq \underset{\text{const}}{3} n^2$

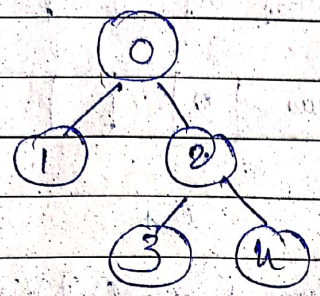
Text Book Example for Searching Topic

Scanned with OKEN Scanner

UNIT - II

BFS : Given an Undirected graph & Source (Queue DS) vertex 'S' print BFS from the Given Source

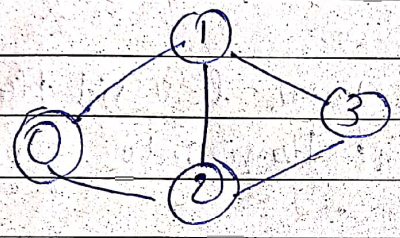
I/p :



S = 0

O/p : 0 1 2 3 4

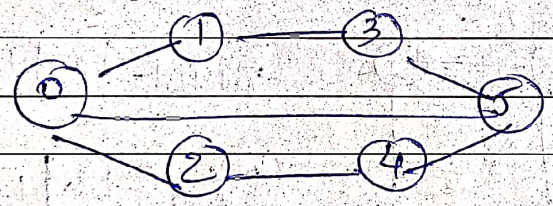
I/p :



S = 0

O/p : 0 1 2 3

I/p :



S = 0

O/p : 0 1 2 5 3 4
(any order)

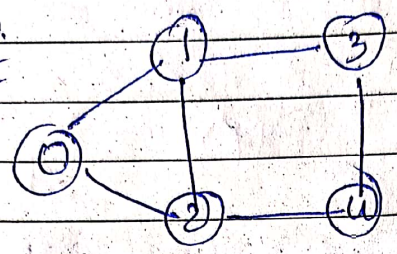
Visit and print

1. Source vertex
2. immediate neighbour vertex not visited
3. this vertex.

Note : Make sure no vertex visited more than Once.

1) Add source to Queue & mark as visited

Ex:



Visited | 0 | | | | |

Queue | 0 | | | | |

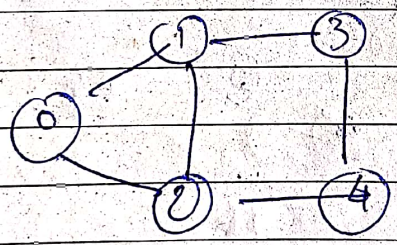
↑ front

push '0' into Queue & mark it visited

2) Remove front of queue & visit the unvisited neighbours & push them into queue

Enque()

Ex:

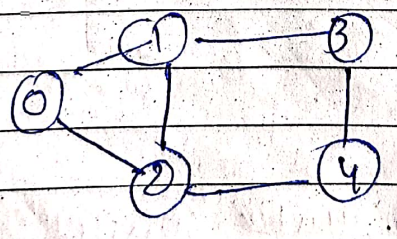


Visited | 0 | 1 | 2 |

Queue | ~~0~~ | 1 | 2 |

↑ front

Remove '0' & visit next node

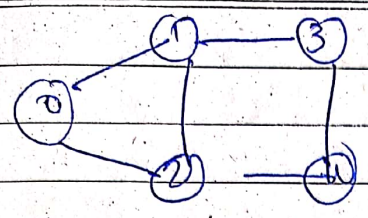


Visited | 0 | 1 | 2 | 3 |

Queue | ~~1~~ | 2 | 3 | |

↑ front

Remove '1' & visit next node

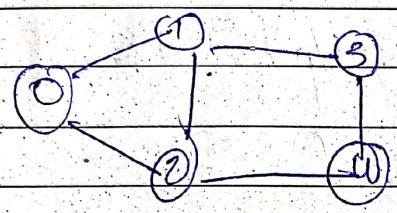


Visited	0	1	2	3	4

Queue	~~2~~	3	4	

↑ front

Remove '2' & visit next node

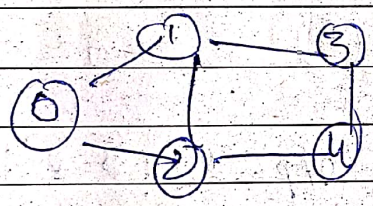


Visited	0	1	2	3	4

Queue	~~3~~	4		

↑ front

Remove '3' & visit next node



Visited	0	1	2	3	4

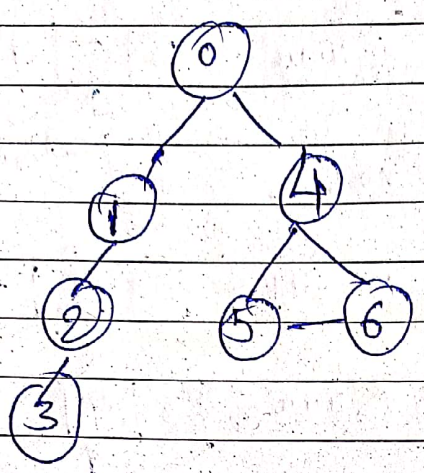
Queue	~~4~~			

↑ front

→ recursive in nature

DFS (Depth first search)

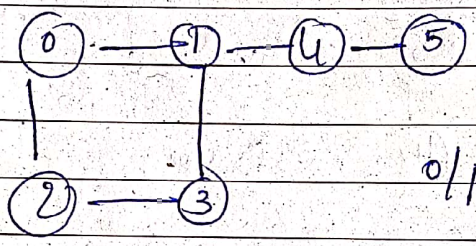
I/p



s = 0

o/p : 0 1 2 3 4 5 6

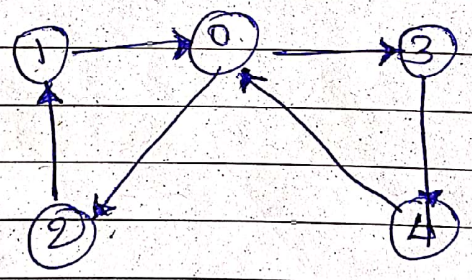
I/p :



s = 0

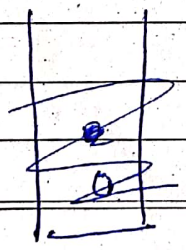
o/p : 0 1 3 2 4 5

Ex :



Adjacency list

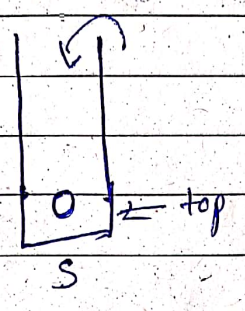
0 → 2, 3
1 → 0
2 → 1
3 → 4
4 → 0



Scanned with OKEN Scanner

1) Initially all nodes are Unvisited in Graph

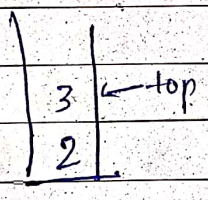
2) Create Stack S & push 0 in it



3) We loop while the Stack is not empty.

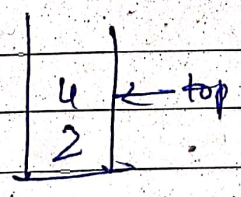
4) Pop "0", print it & mark as visited.

5) Traverse through entire adjacency list of 0 & push not are visited



6) Pop '3', print it & mark it as visited

7) Traverse adjcency list of 3 & push it not visited

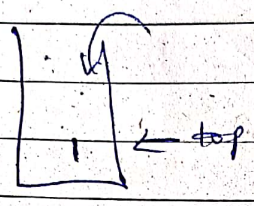


8) pop '4' print it & mark as visited

9) Traverse adjcency list of 4 & push if not visited

10) pop '2' print & mark it as visited

11) Traverse adjacency list of 2 & push if not visited



12) pop '1' print & mark it as visited

13) Traverse adjacency list of 1 & push if not visited

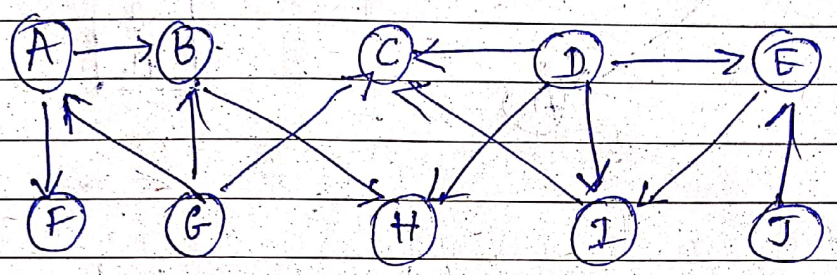
14) Empty

15) End Algorithm

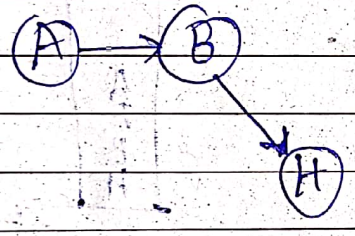
O/P : 0 3 4 2 1

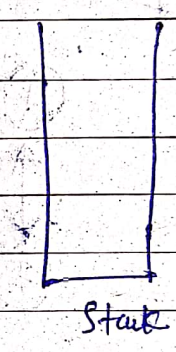
Topological Sorting → DFS approach for DAG Graphs

Topological Ordering : Arrange the vertices of DAG in a sequence or list.

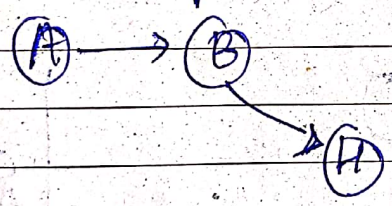


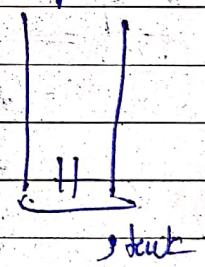
Step 1 : Visit vertices A → B → H



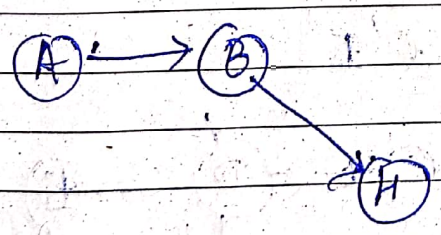


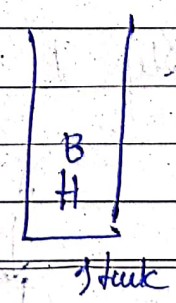
Step 2 : no path for 'H' to visit then push 'H'





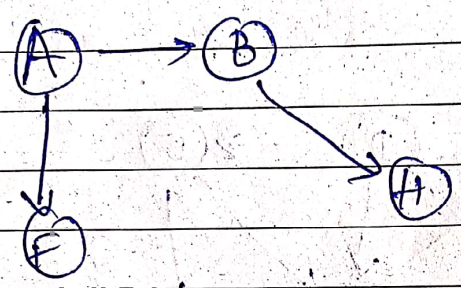
Step 3 : Reverse back to 'B' & push 'B'

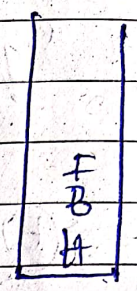




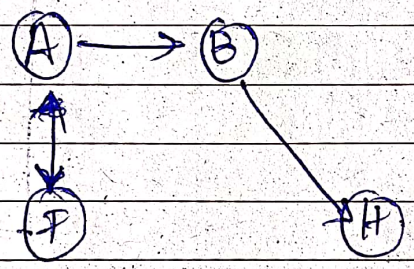
Scanned with OKEN Scanner

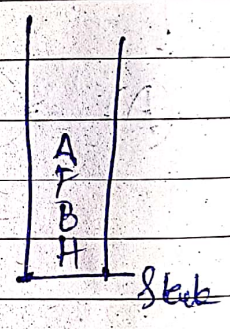
Step 4 : visit another path 'F' vertex & then push to stack



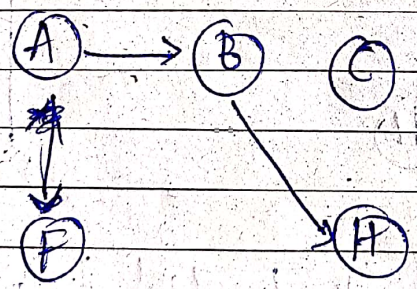


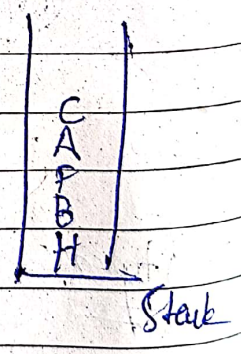
Step 5 : Queue back to 'A' & push 'A' to stack



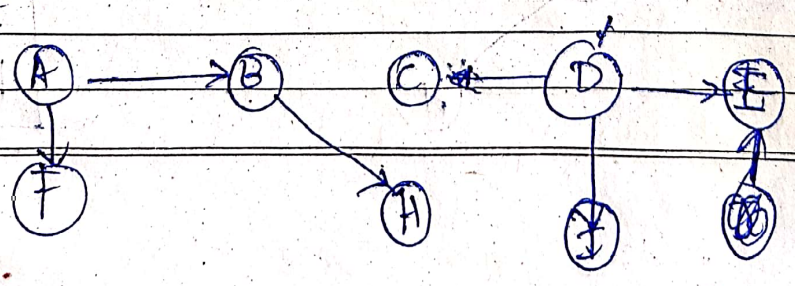


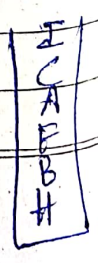
Step 6 : Visit 'C' & push to stack





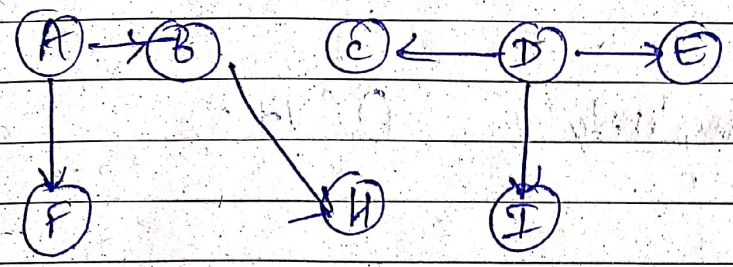
Step 7 : visit 'D' & 'E'





Scanned with OKEN Scanner

Step 8: Visit D → E & push



E
I
C
A
F
B
H

Step 9: push D to stack

D
E
I
C
A
F
B
H

Step 10: Visit 'G' & push

G
D
E
I
C
A
F
B
H

Step 11: Visit 'J' & push

J
G
D
E
I
C
A
F
B
H

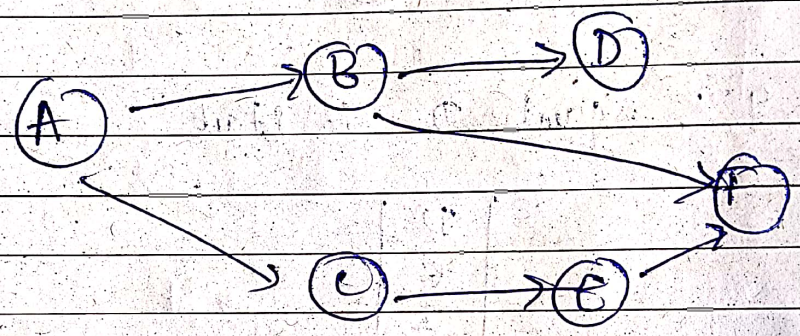
Scanned with OKEN Scanner

Final order is

J G D E I C A F B H

Time complexity : $O(V+E)$

Ex:



Master Method

Step 1 : write down & unify with General Form

$$T(n) = aT(n/b) + f(n)$$

$$a \geq 1 \quad b > 1 \quad f(n)$$

find out : a, b & $f(n)$ value

Step 2 : find out $n^{\log_b a}$

Step 3 : check which case is suitable to solve

→ Case 1 : If $n^{\log_b a} > f(n)$
then $T(n) = O(n^{\log_b a})$

→ Case 2 : If $n^{\log_b a} = f(n)$
then $T(n) = O(n^{\log_b a} \cdot \log n)$

→ Case 3 : If $n^{\log_b a} < f(n)$ $\quad af(n/b) <= c \cdot f(n)$
when $c < 1$
then $T(n) = O(f(n))$

Scanned with OKEN Scanner

Ex: (1) $T(n) = 4T(n/2) + n$

Step 1 : $a = 4$ $b = 2$ $f(n) = n$

Step 2 : $n^{\log_b a} = n^{\log_2 4} = n^{\log_2 2^2} = n^{2(\log_2 2)} = 1$

$\therefore n^{\log_b a} = n^2$
$f(n) = n$ } compare

Step 3 : $n^2 > n$

$\boxed{\text{Case 1}}$ is applicable
with the help of formula find out Time Complexity

$T(n) = O(n^{\log_b a}) = O(n^2)$

(2) $T(n) = T(2n/3) + 1$
Can be written as

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

Step 1 : $a = 1$, $b = 2/3$, $f(n) = 1$

Step 2 : $n^{\log_b a} = n^{\log_{2/3} 1} = n^0 = 1$ ($\log_{2/3} 1 \to 0$)

$\therefore n^{\log_b a} = 1$
$f(n) = 1$ } compare

Step 3 : $1 == 1$

$\boxed{\text{Case 2}}$ is applicable

with the help of formula find out Time Complexity

$T(n) = O(n^{\log_b a} \cdot \log n) = O(1 \cdot \log n)$

$= O(\log n)$

(3) $T(n) = 4T(n/2) + n^3$

Step 1 : $a = 4$, $b = 2$, $f(n) = n^3$

Step 2 : $n^{\log_b a} = n^{\log_2 4} = n^{\log_2 2^2} = n^{2 \log_2 2} = n^2$

$\therefore n^{\log_b a} = n^2$
$f(n) = n^3$ } compare

Step 3 : $n^2 < n^3$

$\boxed{\text{Case 3}}$ is applicable

$a \cdot f(n/b) = 4\left(\frac{n^3}{8}\right) = \frac{n^3}{2}$

$c \cdot f(n) = c \cdot n^3$

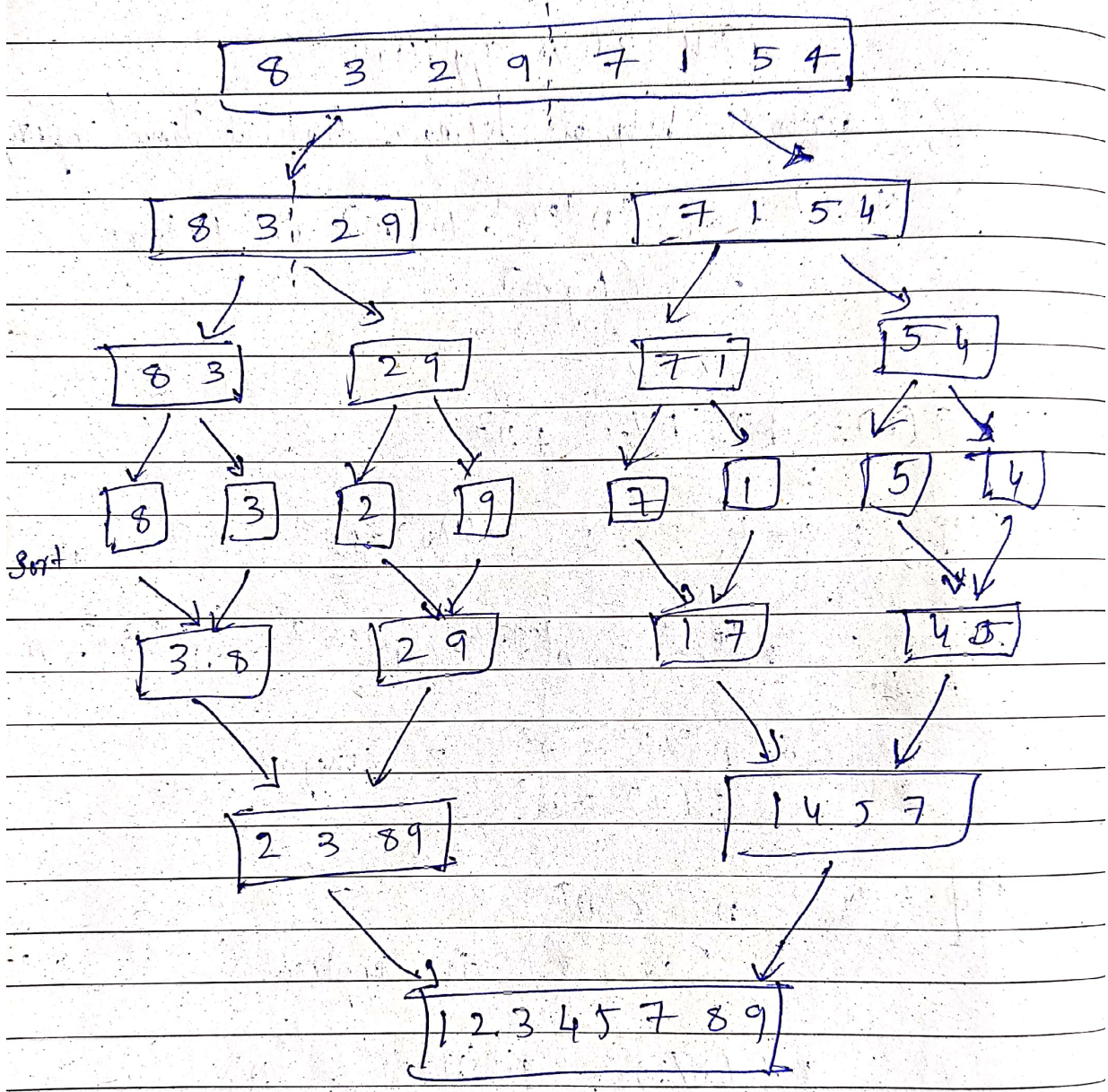
$\frac{n^3}{2} \leq c \cdot n^3$ (ignore)

$\frac{n^3}{2} = c \cdot n^3$

$\boxed{c = 1/2}$ $\therefore c < 1$

$T(n) = O(f(n)) = O(n^3)$

Merge Sort



Scanned with OKEN Scanner

Quick Sort → partition array

Steps →

1. choose pivot (to divide array)
2. partition array
3. Recursively sort sub arrays

0	1	2	3	4	5	6
10	15	1	2	9	16	11

↑

General form → [partition | pivot | partition 2]

value < pivot → array 1

value > pivot → array 2

Step 1 :

0	1	2	3	4	5	6
2	1	9	10	15	11	16

pivot

Pivot = 7

Ex: 1

0	1	2	3	4	5	6	7	8
7	6	10	5	9	2	1	15	17

~~a[0] pivot~~ start ↑ ↑ ↑ end

≤ pivot → increment ~~pivot~~ ~~start pivot~~ start

≥ pivot → decrement end

Step 1 :

0	1	2	3	4	5	6	7	8
7	6	10	5	9	2	1	15	17

↑ ↑ ↑ start ↑ end

Swap a[2] = a[8]

after Swap

Step 2 :

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

↑ start ↑ end

Step 3 :

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

↑ start ↑ end

Swap

After Swap

Step 4 :

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

↑ start ↑ end

Scanned with OKEN Scanner

Step 5 → | 7 | 6 | 7 | 5 | 1 | 2 | 9 | 15 | 10 |

pivot, end, start

'no swap' start crossed end

swap with pivot & end

Step 6 → | 2 | 6 | 7 | 5 | 1 | 7 | 9 | 15 | 10 |

0 1 2 3 4 5 6 7 8

start, pivot, end, pivot

Same procedure

Step 7 →

Ex: 2 | 20 | 2 | 7 | 12 | 15 | 1 | 6 | 8 |

Ex: 3 | 35 | 50 | 15 | 25 | 80 | 20 | 90 | 45 |

pivot

p, q, q

≥ pivot → stop

< pivot → stop

swap

Step 1 → | 35 | 20 | 15 | 25 | 80 | 50 | 90 | 45 |

p, q, p, q

in same position or crossed } swap q with pivot } 1st pass

Step 2 → | 25 | 20 | 15 | 35 | 80 | 50 | 90 | 45 |

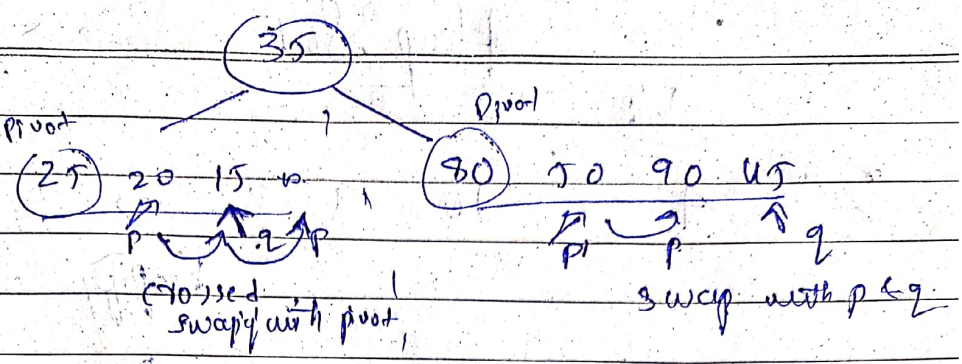
pivot

right position

25, 20, 15

80, 50, 90, 45

Step 3 :



15 20 25 35 80 50 45 90

p q p q

45 50 80 90

15	20	25	35	45	50	80	90

Scanned with OKEN Scanner

Heap Sort

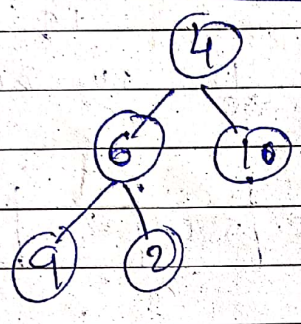
I/p O/p

Ex: 4 6 10 9 2 ⇒ 2 4 6 9 10

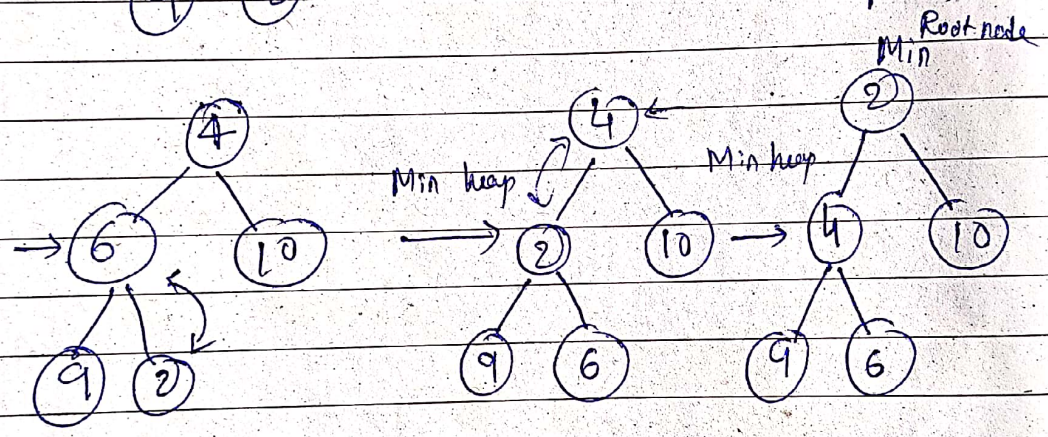
heapify

1) Build Heap Method = O(n) time

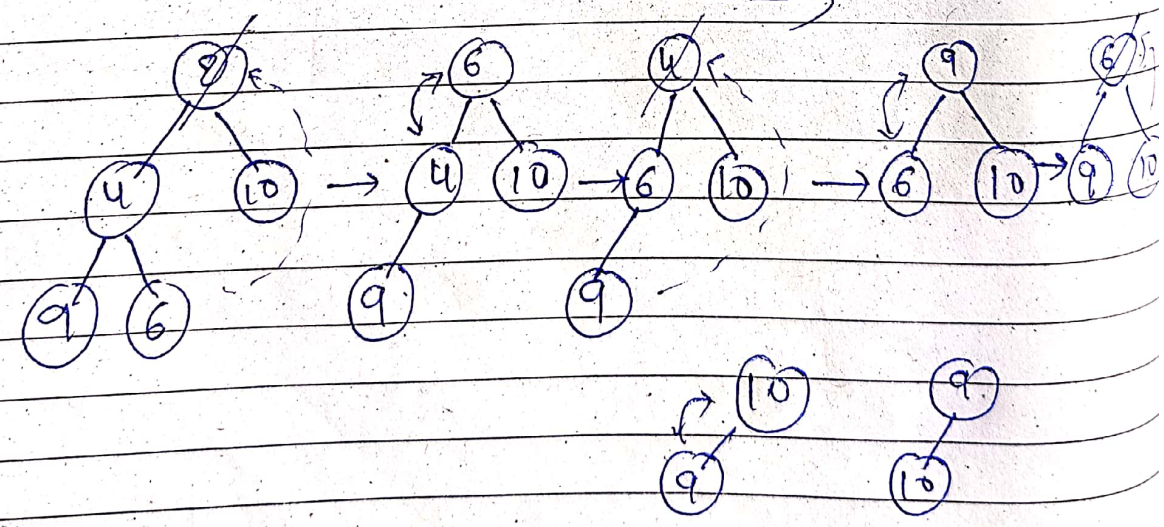
Complete binary tree funda



make either max heap or min heap.



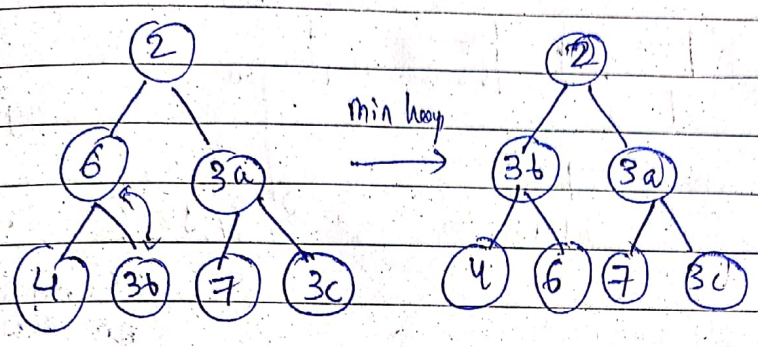
2) Sort (delete from Root)



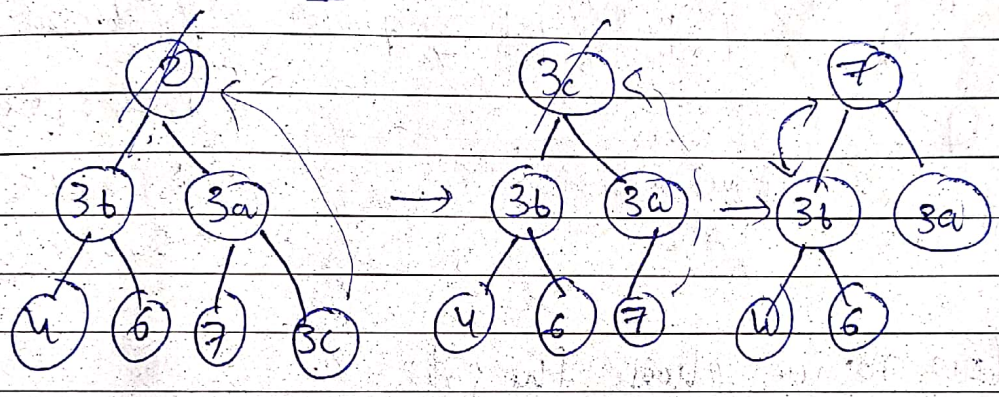
O/p: 2 4 6 9 10 → Sorted

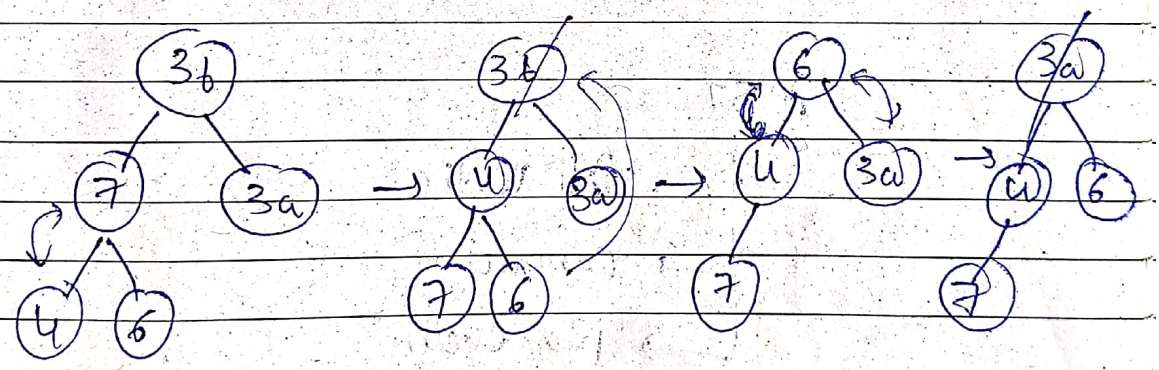
Heap sort is Unstable

Ex: 2 6 3a 4 3b 7 3c



Sort (delete)

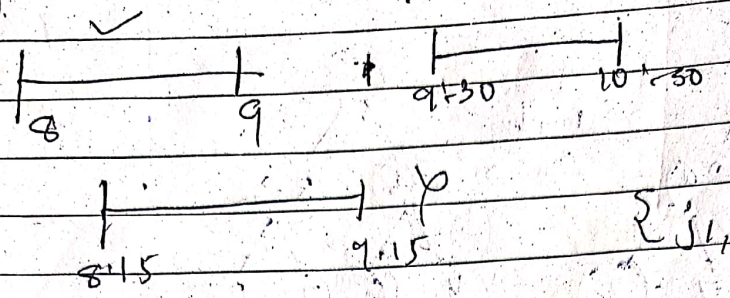




2 3c 3b 3a 4 6 7

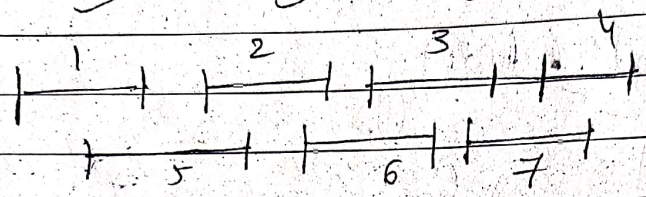
Interval Scheduling problem

Ex: ①



3 jobs

$\{j_1, j_2\}$ → Compatible no overlap possible

② 

timing →

feasible sol^n

$\{1, 2, 3, 4\}$

$\{5, 6, 4\}$

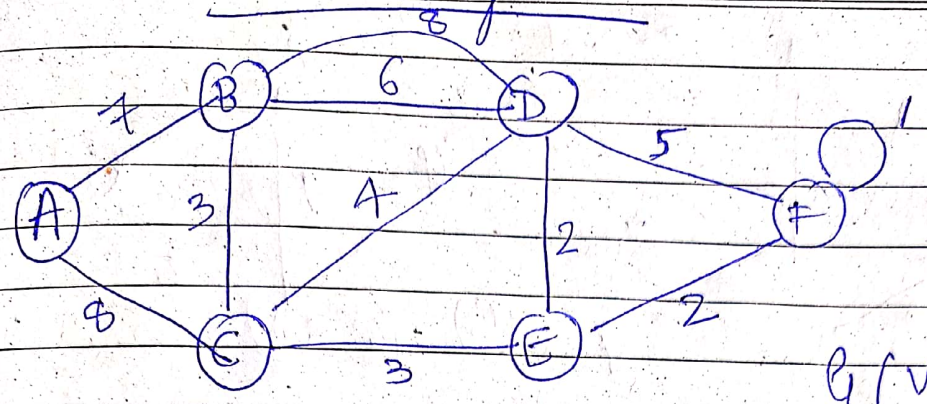
$\{5, 6, 7\}$

How do we choose the jobs ?

MST

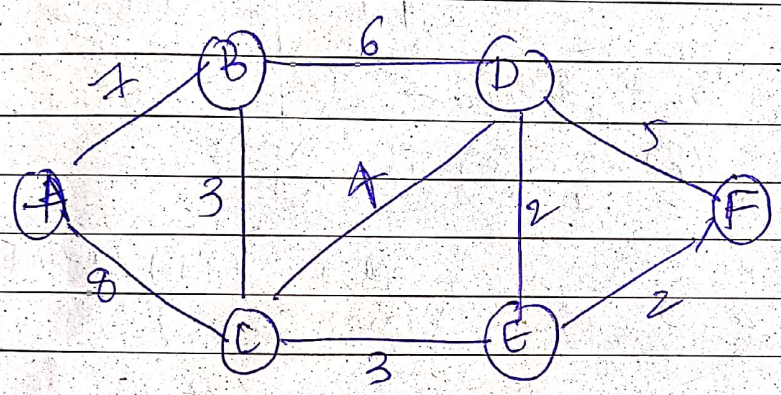
- → All vertices
- → connected
- → No cycles

Prims Algorithm

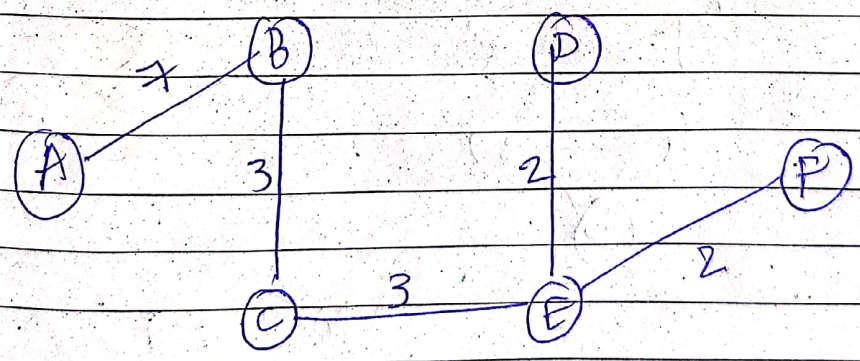


G(V, E)

Remove all loops & parallel Edges from the given Graph



Traverse the nodes or vertices from any node and calculate MST.



$E = V - 1$

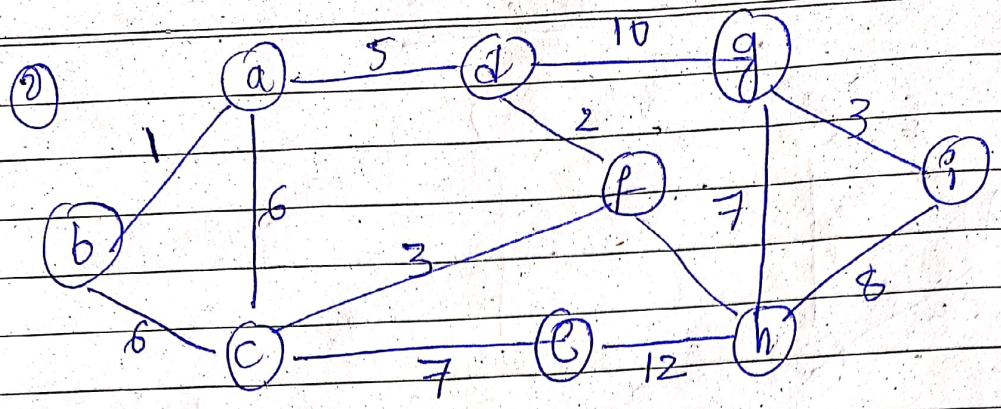
$= 6 - 1$

$\boxed{E = 5}$

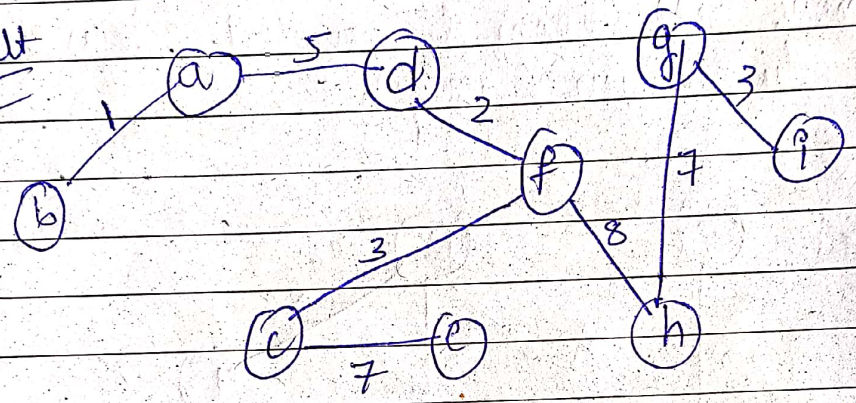
Result $= 7 + 3 + 3 + 2 + 2$

$\boxed{MST = 17}$

(2)



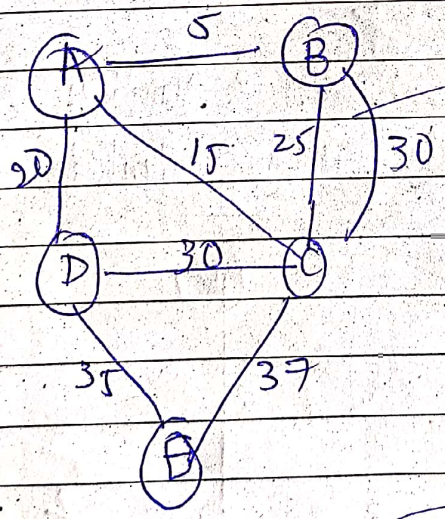
Result



MST = 1+5+2+3+7+8+7+3

$\boxed{MST = 36}$

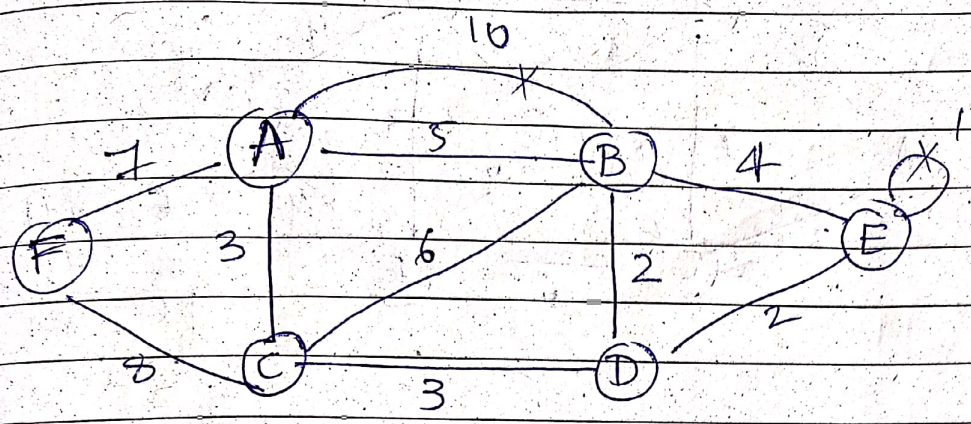
H.W



→ remove parallel edge.

$\boxed{MST = 75}$

Kruskal's Algorithm



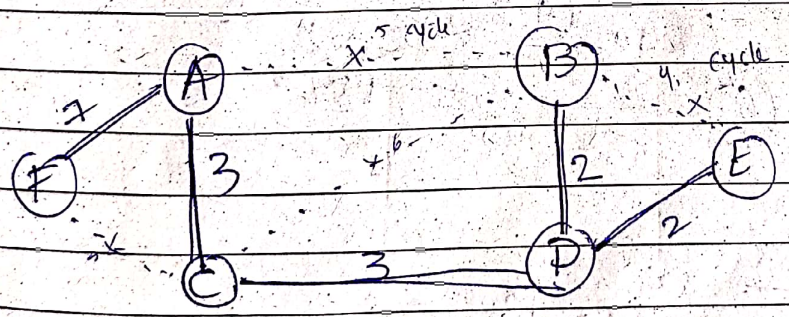
Arrange all Edges in increasing order of Edge weight

(1) Arrange edge weight in increasing order.

(2) Choose minimum edge weight from order

// should contain cycle if it is discard & proceed

BD = 2
DE = 2
AC = 3
CD = 3
× BE = 4
× AB = 5
× CB = 6
AF = 7
× CF = 8
order ~~Sorted~~ Edges



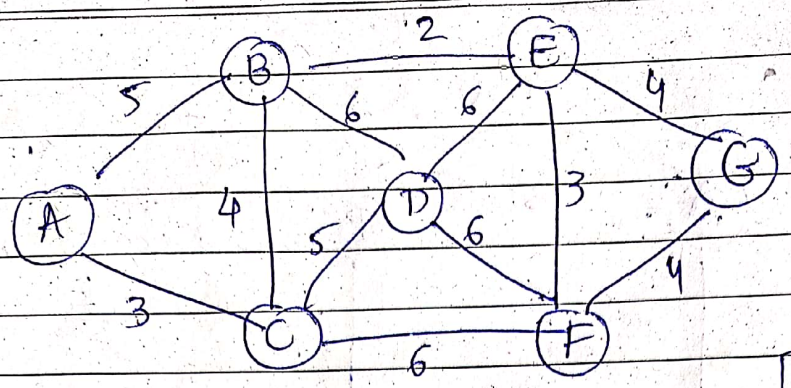
$E = V - 1$

$= 6 - 1$

$\boxed{E = 5}$

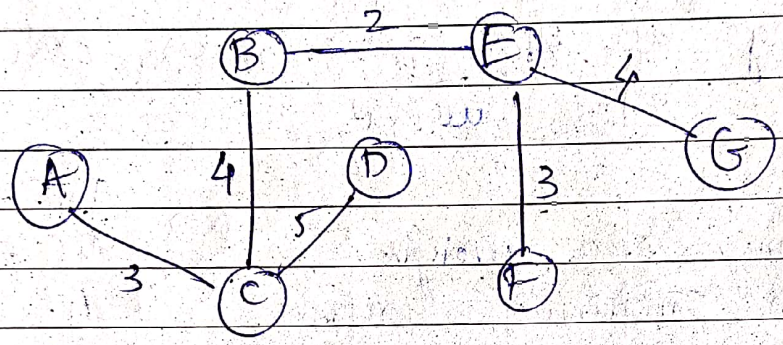
Total Cost $= 7 + 3 + 3 + 2 + 2 = 17$ //

(2)



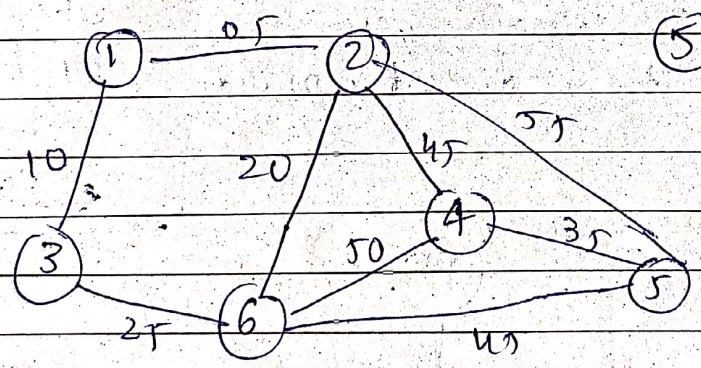
$E = V - 1$
$= 7 - 1$
$\boxed{E = 6}$

Result

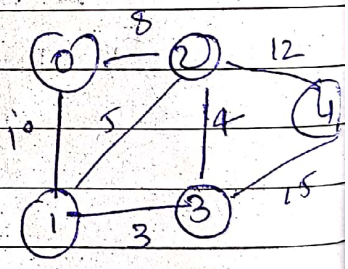


Total cost = 3+4+5+2+4+3 = 21

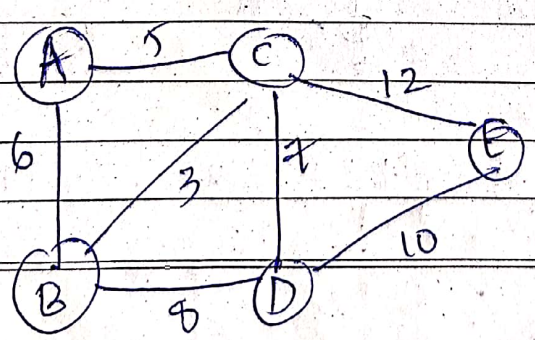
(3)



(5)



(4)



Total cost = 25

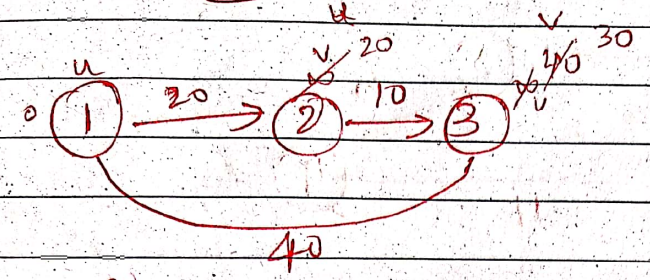
Ex: Gmap

solve for DAG & Undirected

Single Source Shortest Path

Dijikstra's Algorithm

(1)



Minimum Cost

if $d(u) + c(u, v) < d(v)$:-

update $d(v) = d(u) + c(u, v)$

pass 1 : $0 + 20 < \infty$

$\longrightarrow$ $20 < \infty$ then update $d(v)$

pass 2 : $0 + 40 < \infty$

$\longrightarrow$ $40 < \infty$ then update $d(v)$

pass 3 : $20 + 10 < 40$

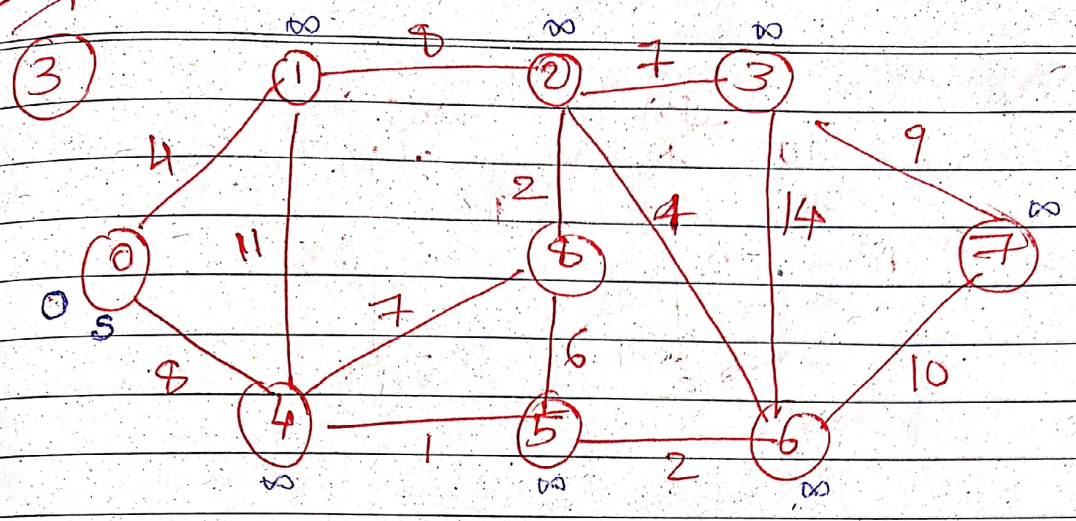
$\longrightarrow$ $30 < 40$ then update $d(v)$

Scanned with OKEN Scanner

Jenny's

Work on Directed & Undirected Graph

Single Source Shortest Path

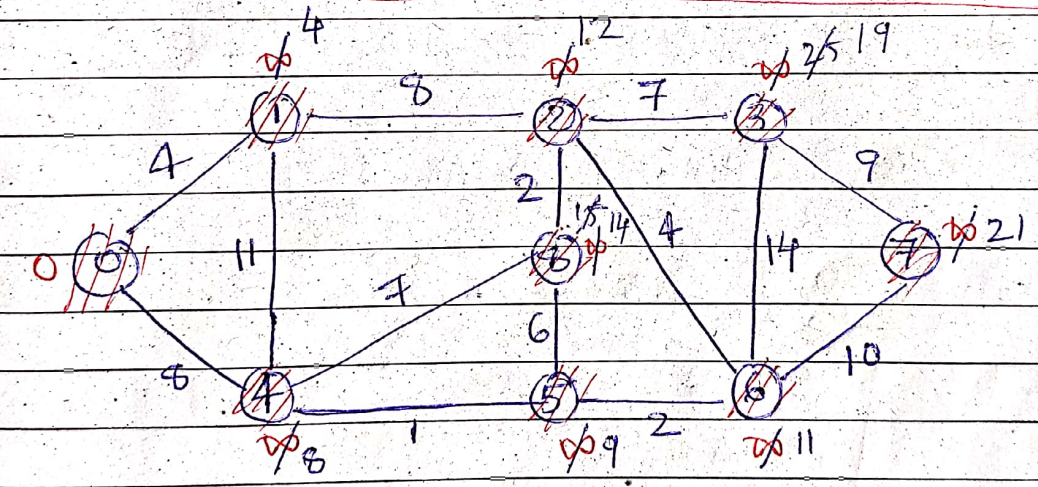


if $d(u) + c(u,v) < d(v)$

$d(v) = d(u) + c(u,v)$

To calculate Shortest distance

Shortest distance, 0 - 7 = 21 , 0 - 2 = 18
0 - 1 = 4 , 0 - 4 = 8



pass 1 : $0 + 4 < \infty$, 4 < ∞ (min) then update

pass 2 : $0 + 8 < \infty$, 8 < ∞ (min) then update

pass 3 : $4 + 8 < \infty$, 12 < ∞ then update

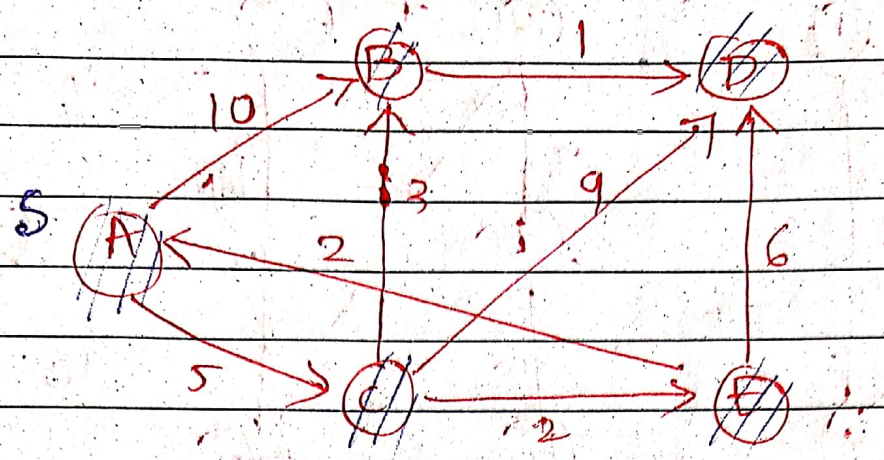
pass 4 : $4 + 11 < 8$, $15 < 8$ not update ×

pass 5 : $8 + 1 < \infty$, $9 < \infty$ then update

pass 6 : $8 + 7 < \infty$, $15 < \infty$ then update

Scanned with OKEN Scanner

5



	Source				
	A min	B	C	D	E
A	0	∞	∞	∞	∞
C		10	5 min	∞	∞
E		8		14	7 min
B		8 min		13	
D				9 min	

5+3 < 10
8 < 10

Short test distance : A-B-(8), A-E-(7), A-D-(9), A-C-(5)

Short test path A-D : DBCA → ACBD = 9
5+3+1

path A-B : BCA → ACB = 8
5+3

6.

Huffman coding

→ data compression technique [or]

→ Variable length coding Algo

Ex: computer (data compression)

A B B C D B C C D A A B B E E E B E A B = 20 char

A – 65 – 01000001 = 8 bits

ASCII

20 × 8 = 160 bits

fixed length coding

Ex: frequency

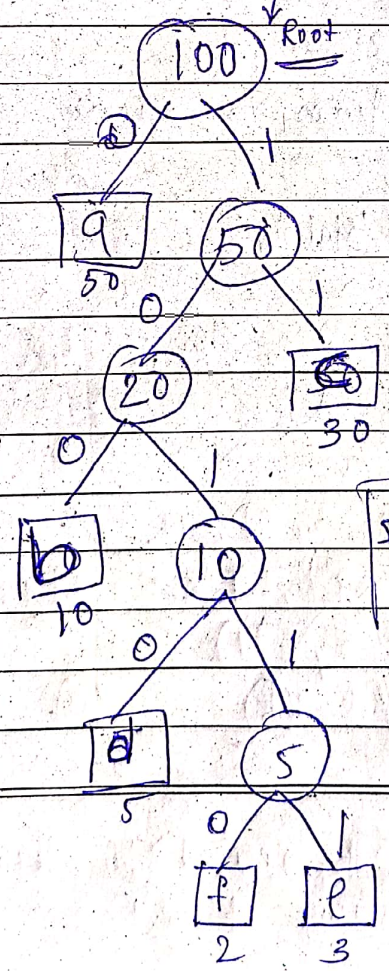
a = 50 d = 5
b = 10 e = 3
c = 30 f = 2

100 × 8 = 800 bits

original ~~char~~ bits

1) Consider min. frequencies & add to root node
(keep smaller element to left and bigger right)

2) add '0' to left & '1' to right



huffman code

a = 0
b = 100
c = 11
d = 1010
e = 1011
f = 10110

Compressed bits

50×1 + 10×3 + 30×2 + 5×4 + 3×5 + 2×5 = 185 bits

Sender (To) Receiver →

01111101 - - - - - - - -

(1) M = ABBCDBCCDAABBEEEBEAB

Char	frequency/count	Huffman code
A	4	01
B	7	11
C	3	101
D	2	100
E	4	00
~~E~~		

Construct huffman tree and find huffman code.

2
3 5
4
4
7

1) Choose minimum frequency and keep on adding it to root node.

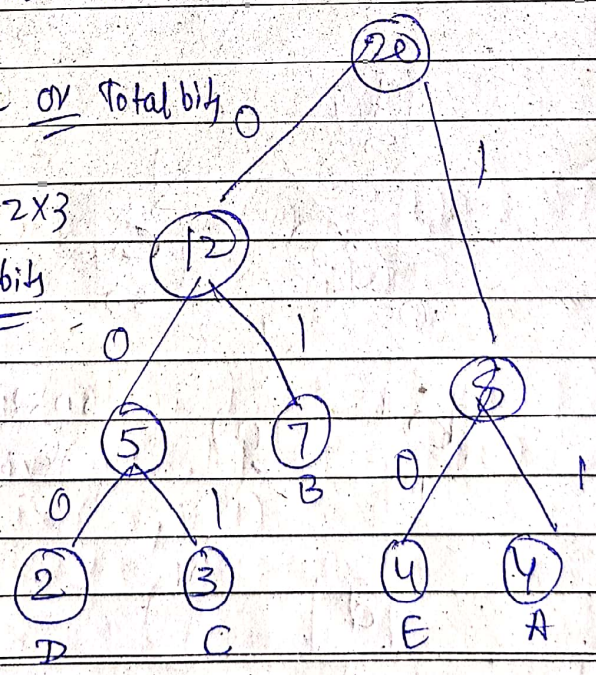
2) Assign the values '0' & '1' to get huffman code

3) Traverse from root node

length of message or Total bits

$4\times2+7\times2+3\times3+2\times3+4\times2 = 45$ bits

Compound bits



Share tree or table to receiver to decode message or text

Knapsack Problem

pic item completely

2 kg, 3 kg

1 kg ✗ 2 kg ✓

Total combination $2^4 = 16$ combination

∴ Check with each combination so that you will get max. profit.

Total no. of weights selected from items $\leq W$

or

Dynamic programming method.

(divided into subproblems) with sequence of steps.

not picked, picked

0/1 Knapsack pblm → with DPM $\sum w_i x_i \leq W$

(1) weights $= \{3, 4, 6, 5\}$

profits $= \{2, 3, 1, 4\}$

$W = 8$ → max capacity of Bag

$n = 4$

$W \longrightarrow$ max, max, max ✓

P_i	W_i	i or n ↓	0	1	2	3	4	5	6	7	8
		0	0	0	0	0	0	0	0	0	0
2	3	1	0	0	0	2 (conv value)	2	2	2	2	2
3	4	2	0	0	0	2	3	3	3	5	5
4	5	3	0	0	0	2	3	4	4	5	6
1	6	4	0	0	0	2	3	4	4	5	6

ascending order

→ max profit

max $= (3+2, 2) = (5, 2)$

(a) w_1 w_2 8

3, 4

(4-3) (just above cell)

max $= (3+0, 2)$
$= (3, 2)$
$= (3)$

it selected ✓ 4 (5-4) not selected ✗ 4 → 3

max $= (3+0, 2) \rightarrow (5)$
$= (3, 2)$
$= 3$

(6-4)

max $= (3+0, 2)$
$= 3, 2$
$= 3$

(7-3) upper cell

max $= (3+2, 2)$
$= (5, 2)$
$= 5$

For

(B) w_1, w_2, w_3
3, 4, 5

* max = (4+0, 5)
= 4, 5
= 5

* max = (4+0, 3)
= 4, 3
= 4

* max = (4+2, 5)
= (6, 5)
= 6

* max = (4+0, 3)
= 4, 3
= 4

c) For $w_4 = 6$

* max = (1+0, 4)
= (1, 4)
= 4

* max = (1+0, 5)
= (1, 5)
= 5

* max = (1+0, 6)
= (1, 6) = 6

To unify table entered values

unify or use formula

$$m[i, w] = \max(m[i-1, w], m[i-1, w-w[i]] + p[i])$$

$$m[4, 7] = \max(m[3, 7], m[3, 7-6] + 1)$$
$$= \max(m[3, 7], m[3, 1] + 1)$$
$$= \max(5, 0+1)$$
$$= \max(5, 1)$$
$$= 5$$

remainder profit
6 − 4 = 2
2 − 2 = 0

approach 1 →

$$x_i = \{1, 0, 0, 1\}$$

Scanned with OKEN Scanner

0/1 Knapsack pblm

(1) items $n = 4$, weight $W = 5$

$w_1, w_2, w_3, w_4 = \{2, 3, 4, 5\}$

profit $= \{3, 4, 5, 6\}$

item	weight	profit
1	2	3
2	3	4
3	4	5
4	5	6

Calculate the max. profit value

Sol^n:

rows $\boxed{W}$ →

column $\boxed{i}$ ↓

	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0					
2	0					
3	0					
4	0					
5	0					

Step 1:

approach 2

or $x_i = \{1, 0, 0, 1\}$

$8 - 5 = 3$

$3 - 3 = 0$

Fractional Knapsack

$N = 3$ $\quad W = 50 \rightarrow$ Total Capacity

~~40~~ 20

Val = [60, 100, 120]

Wt = [10, 20, 30]

per unit wt $\Rightarrow$ 6, 5, 4

Decreasing order of per unit value

fractional

Ans $= 60 + 100 + \boxed{4 \times 20}$

$= 60 + 100 + 80$

$= 240$

Greedy Algorithm

Ex:

I/p:

Val = T

Bellman Ford

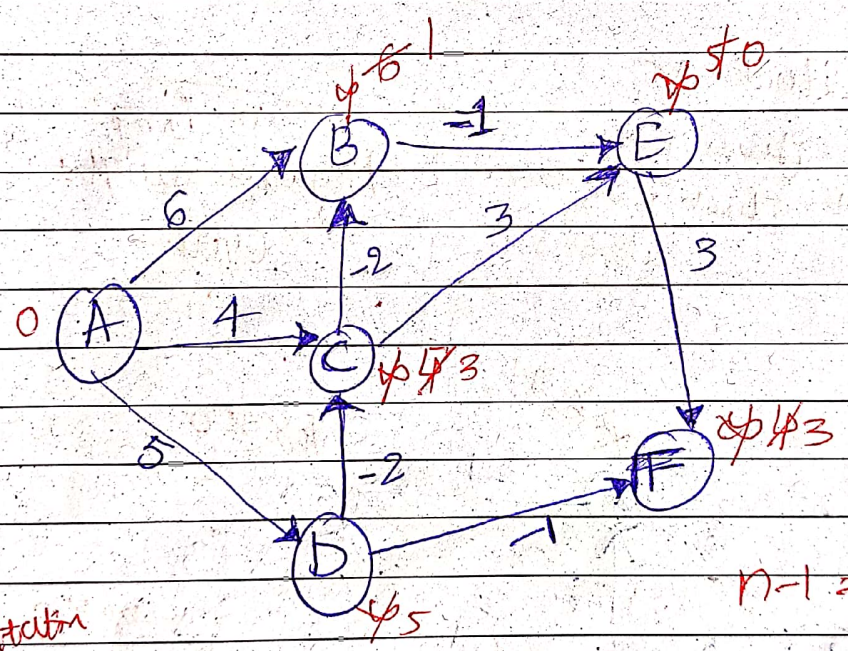
- Slower than Dijkstra's
- Single Source Shortest path
- Helps you to detect negative cycle
- Directed Graph
- Dijkstra's Algorithm does not work on graph having $\boxed{-ve}$ weights but Bellman works

$\boxed{n-1 \text{ times}}$

n = no. of vertices

$$\boxed{\begin{array}{l} \text{if } (d[u] + c(u,v) < d[v]) \\ \quad d[v] = d[u] + c(u,v) \end{array}}$$

Ex:



Shortest distance

A	0
B	1
C	3
D	5
E	0
F	3

$n-1 = 6-1$
$= 5$ iteration

1st iteration

Pass 1 : $d[u] + c[u,v] < d[v]$

$0 + 6 < \infty \rightarrow$ update $d[v]$ as 6

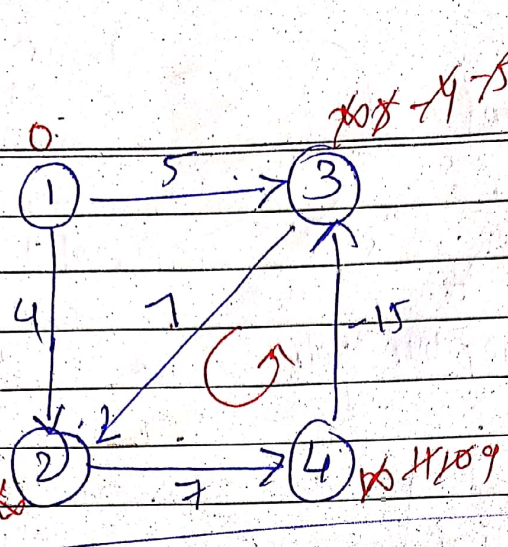
2nd iteration

3rd iteration - - - - 5th iteration.

Edges

(A,B) (A,C) (A,D), (B,E) (C,E) (D,C) (D,F) (E,F) (C,B)

Ex:



$7+7-15=-1$

will not give correct ans

!!! B-F Algo will not work if Graph Contains any negative weight cycle

list out all Edges

(1,3) (1,2) (3,2) (2,4) (4,3)

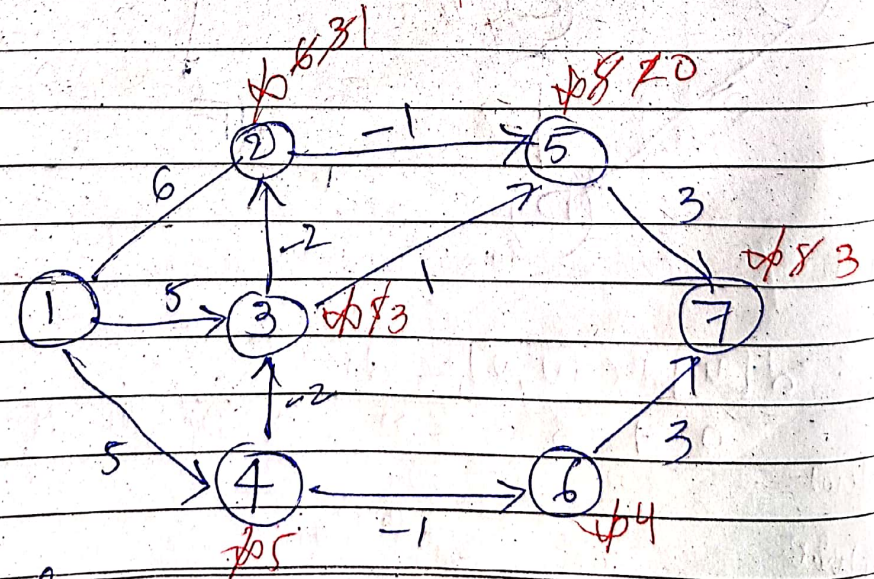
1st iteration ✓

2nd iterat ✓

3rd iterat ✓

This Graph Negative weight cycle

Ex:



Group of Edges

$(7-1) = 6$ edges

(1,2) (1,3) (1,4) (2,5) (3,2) (3,5) (4,3)
(4,6) (5,7) (6,7)

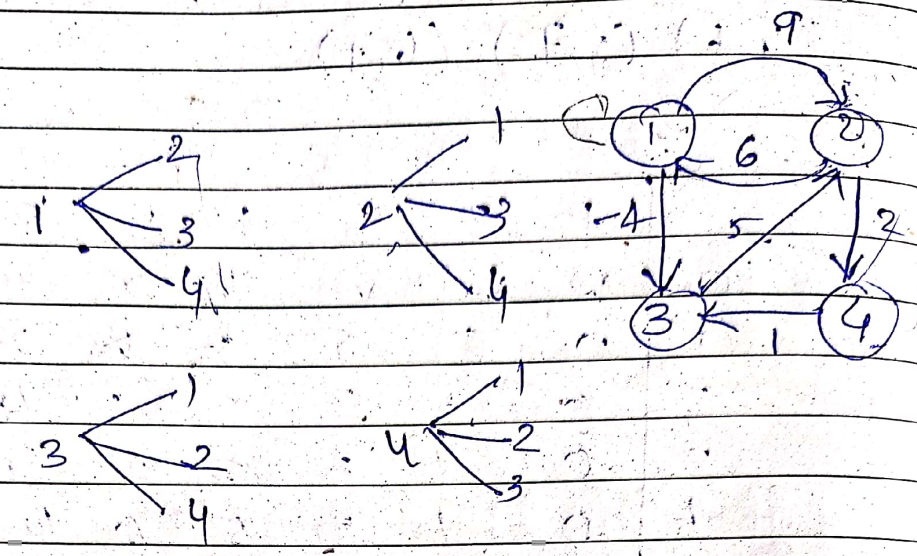
	1	2	3	4	5	6	7
0	∞	∞	∞	∞	∞	∞	∞
1	0	6	5	5	∞	∞	∞
2	0	3	3	5	5	4	∞
3	0	1	3	5	2	4	5
4	0	1	3	5	0	4	3
5	0	1	3	5	0	4	3
6	0	1	3	5	0	4	3

Scanned with OKEN Scanner

→ All pairs shortest path

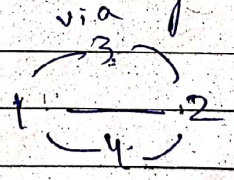
WARSHALL - FLOYD Algo

Ex:



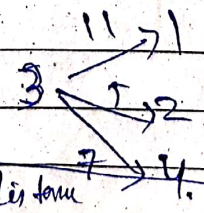
Dynamic program approach

- → Simplifying complicated pblm by breaking down into simple sub problem
- → Some sequence decision is taken out at each step of pblm
- → works on negative weight Edge



9 ①

Check with shortest path from 3



$\boxed{\begin{array}{l} K=4 \\ i=1 \\ j=2 \end{array}}$

To verify (intermediate) middle node distance

$D^k[i,j] = \min\{D^{k-1}[i,j], D^{k-1}[i,k] + D^{k-1}[k,j]\}$ → previous matrix

$\min D^3[1,2] = D^3[1,4] + D^3[4,2]$

$1 < 3 + 6 = 9$

$$D^0 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 9 & -4 & \infty \\ 2 & 6 & 0 & \infty & 2 \\ 3 & \infty & 5 & 0 & \infty \\ 4 & \infty & \infty & 1 & 0 \end{array}$$

Distance matrix

→ Base matrix

via 1

$$D^1 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 9 & -4 & \infty \\ 2 & 6 & 0 & 2 & 2 \\ 3 & \infty & 5 & 0 & \infty \\ 4 & \infty & \infty & 1 & 0 \end{array}$$

Distance matrix via 1

* $D^0[2,3] = D^0[2,1] + D^0[1,3]$

  $\infty \rightarrow 6 + (-4)$ (2)

* $D^0[2,4] = D^0[2,1] + D^0[1,4]$

  $2 < 6 + \infty = (\infty)$

* $D^0[3,2] = D^0[3,1] + D^0[1,2]$

  $5 < \infty + 9 = \infty$

$$D^2 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 9 & -4 & 11 \\ 2 & 6 & 0 & 2 & 2 \\ 3 & 11 & 5 & 0 & 7 \\ 4 & \infty & \infty & 1 & 0 \end{array}$$

Distance matrix via 2

* $D^1[1,3] = D^1[1,2] + D^1[2,3]$

  $-4 < 9 + 2 = 11$

* $D^1[1,4] = D^1[1,2] + D^1[2,4]$

  $\infty \geq 9 + 2 = 11$

* $D^1[3,1]$ $D^1[3,2] + D^1[2,1]$

  $\infty < 5 + 6 = 11$

$$D^3 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 1 & -4 & 3 \\ 2 & 6 & 0 & 2 & 2 \\ 3 & 11 & 5 & 0 & 7 \\ 4 & 12 & 6 & 1 & 0 \end{array}$$

* $D^2[1,2] = D^2[1,3] + D^2[3,2]$

  $9 > -4 + 5 = (1)$

* $D^2[1,4] = D^2[1,3] + D^2[3,4]$

  $11 > -4 + 7 = (3)$

* $D^2[2,1] = D^2[2,3] + D^2[3,1]$

  $6 < 2 + 11 = (13)$

$$D^4 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 1 & -4 & 3 \\ 2 & 6 & 0 & 2 & 2 \\ 3 & 11 & 5 & 0 & 7 \\ 4 & 12 & 6 & 1 & 0 \end{array}$$

* $D^3[1,2] = D^3[1,4] + D^3[4,2]$

  $1 < 3 + 6 = (9)$

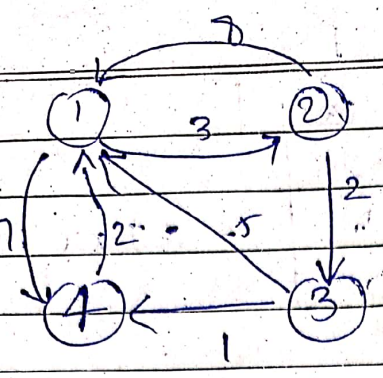
* $D^3[1,3] = D^3[1,4] + D^3[4,3]$

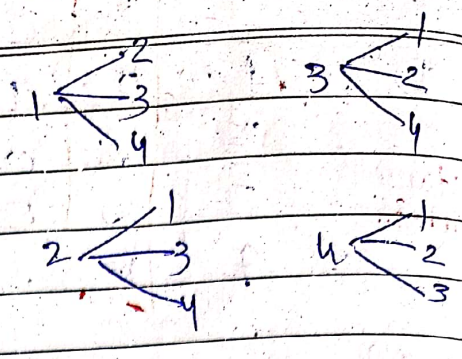
  $-4 < 3 + 1 = (4)$

* $D^3[2,1] = D^3[2,4] + D^3[4,1]$

Scanned with OKEN Scanner

H.W





$$A^0 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 3 & \infty & 7 \\ 2 & 8 & 0 & 2 & \infty \\ 3 & 5 & \infty & 0 & 1 \\ 4 & 2 & \infty & \infty & 0 \end{array}$$

Base matrix

$$A^1 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 3 & \infty & 7 \\ 2 & 8 & 0 & 2 & 15 \\ 3 & 5 & 8 & 0 & 1 \\ 4 & 2 & 5 & \infty & 0 \end{array}$$

$$A^2 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 3 & 5 & 7 \\ 2 & 8 & 0 & 2 & 15 \\ 3 & 5 & 8 & 0 & 1 \\ 4 & 2 & 5 & 7 & 0 \end{array}$$

$$A^3 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 3 & 5 & 6 \\ 2 & 7 & 0 & 2 & 3 \\ 3 & 5 & 8 & 0 & 1 \\ 4 & 2 & 5 & 7 & 0 \end{array}$$

$$A^4 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 3 & 5 & 6 \\ 2 & 5 & 0 & 2 & 3 \\ 3 & 3 & 6 & 0 & 1 \\ 4 & 2 & 5 & 7 & 0 \end{array}$$

Scanned with OKEN Scanner

Algo

for generating matrix

no. of vertices $n \times n$

∴ So use 2 for loops

```
for (k=1; k≤n; k++)
{
  for (i=1; i≤n; i++)
  {
    for (j=1; j≤n; j++)
    {
      D[i,j] = min(D[i,j], D[i,k] + D[k,j])
    }
  }
}
```

Scanned with OKEN Scanner

LPP

(1) Maximize $z = 12x_1 + 16x_2$

Subject to

$10x_1 + 20x_2 \le 120$ —— (1)

$8x_1 + 8x_2 \le 80$ —— (2)

$x_1 \& x_2 \ge 0$

Sol^n

$10x_1 + 20x_2 = 120$ —— (1)

x_1	0	12
x_2	6	0

$x_1 = 0$ $x_2 = 6$

$10(0) + 20x_2 = 120$

$20x_2 = 120$

$x_2 = \frac{120}{20}$

$\boxed{x_2 = 6}$

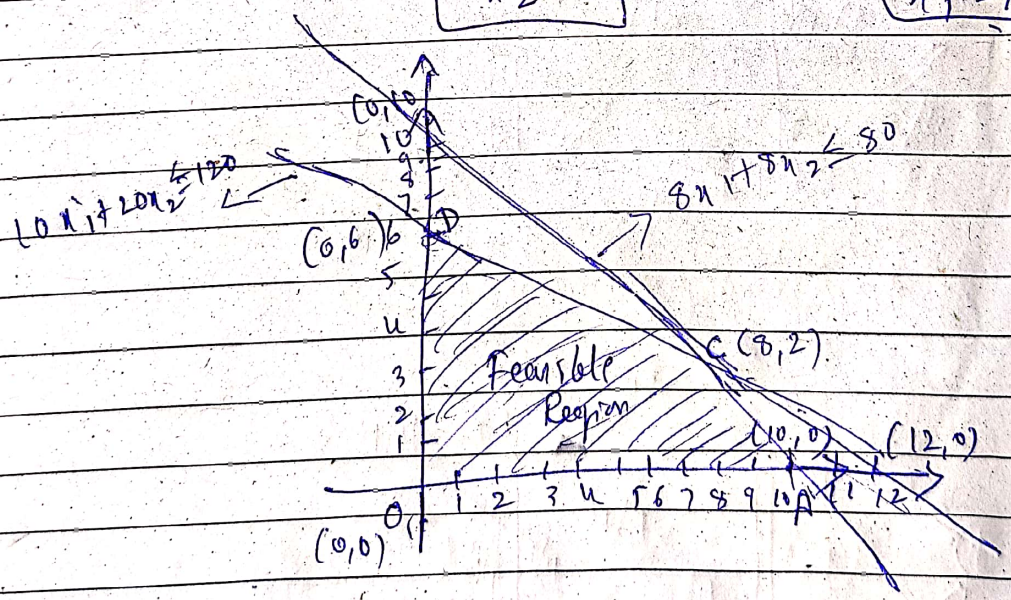
$x_2 = 0$ $x_1 = 12$

$10x_1 + 20(0) = 120$

$10x_1 = 120$

$x_1 = \frac{120}{10}$

$\boxed{x_1 = 12}$



$8x_1 + 8x_2 = 80$ — (2)

x_1	0	10
x_2	10	0

when $x_1 = 0$ $x_2 = 10$

$8x_1 + 8x_2 = 80$

$8(0) + 8x_2 = 80$

$8x_2 = 80$

$x_2 = \frac{80}{8}$ 10

$\boxed{x_2 = 10}$

when $x_1 = ?$ $x_2 = 0$

$8x_1 + 8x_2 = 80$

$8x_1 + 8(0) = 80$

$8x_1 = 80$

$x_1 = \frac{80}{8}$ 10

$\boxed{x_2 = 10}$

Max $Z = 12x_1 + 16x_2$

$Z(A) = 12(0) + 16(0) = 0$

$Z(B) = 12(10) + 16(0) = 120$

$Z(C) = 12(8) + 16(2) = 96 + 32 = 128$ ✓

$Z(D) = 12(0) + 16(6) = 0 + 96 = 96$

∴ The Z value is maximum for corner point 'C'

$x_1 = 8$, $x_2 = 2$

$\boxed{Z(\text{optimum}) = 128}$

→ max. flow in flow n/w

Ford - Fulkerson

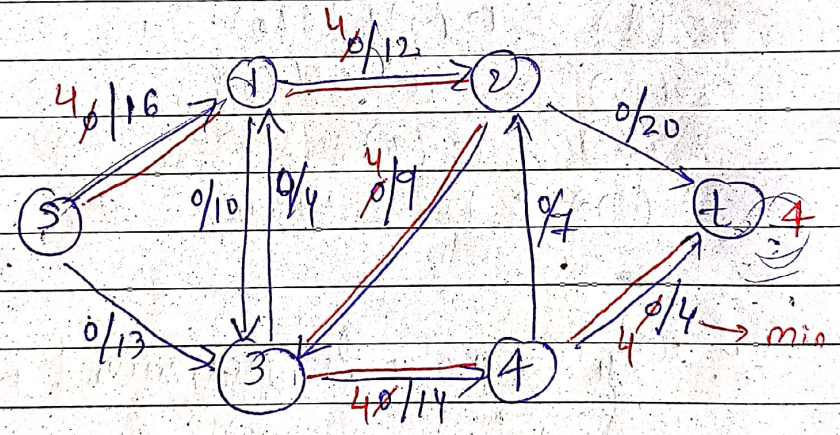
→ minimum cut
→ Residual Graph

Maximum flow
Source → Sink

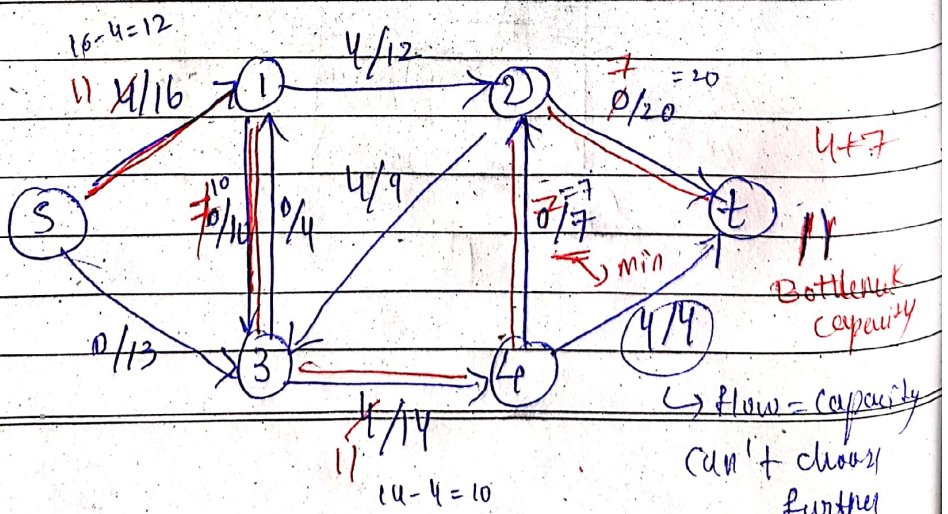
Steps

1) find an augmenting path
2) Compute the bottleneck capacity
3) augment each edge & total flow

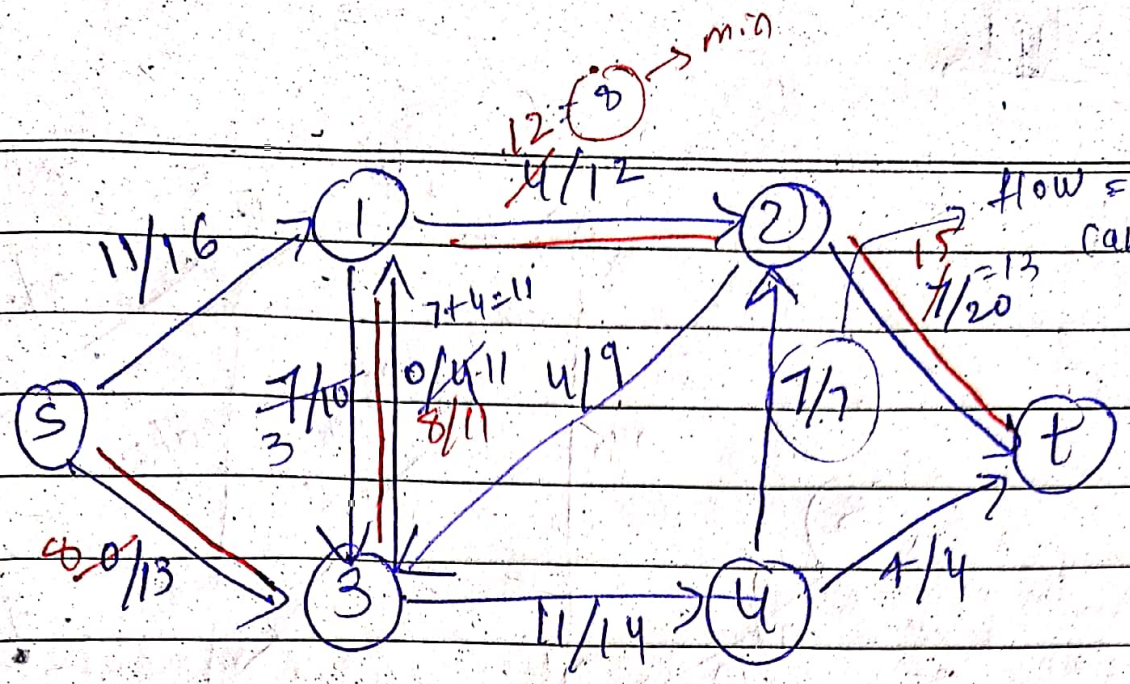
Step 1



Choose augmented path from S-T

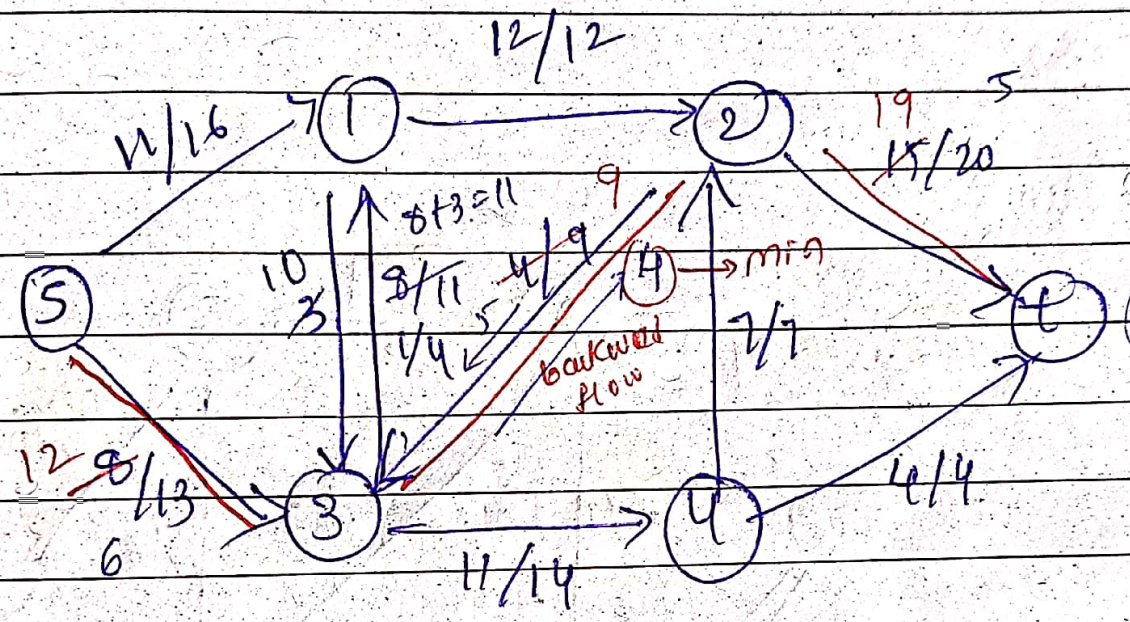


↳ Flow = capacity
can't choose further



flow = capacity
can't choose further

11+8
=19



19+4

23

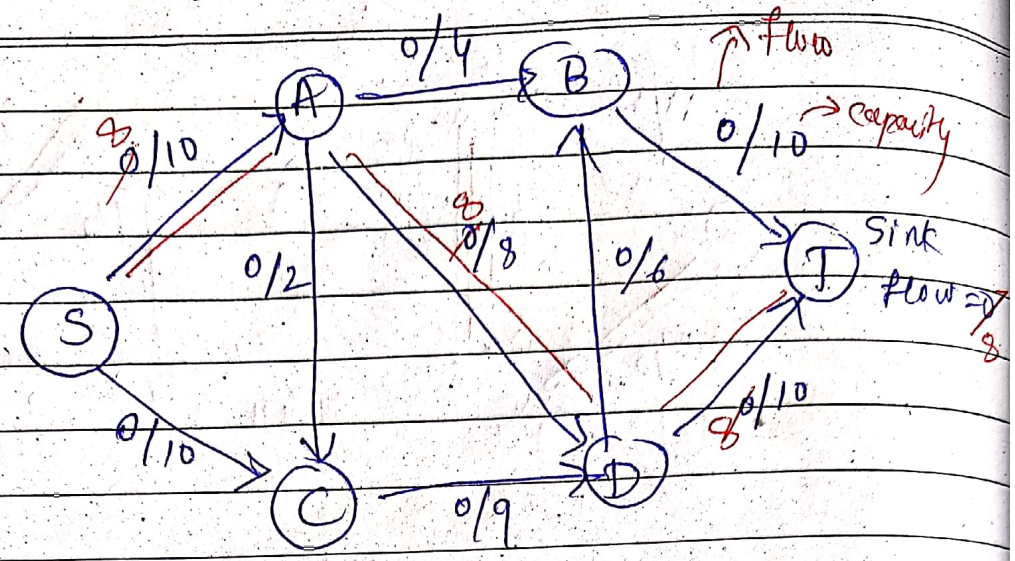
outgoing flow = incoming flow

11+12 = 19+4

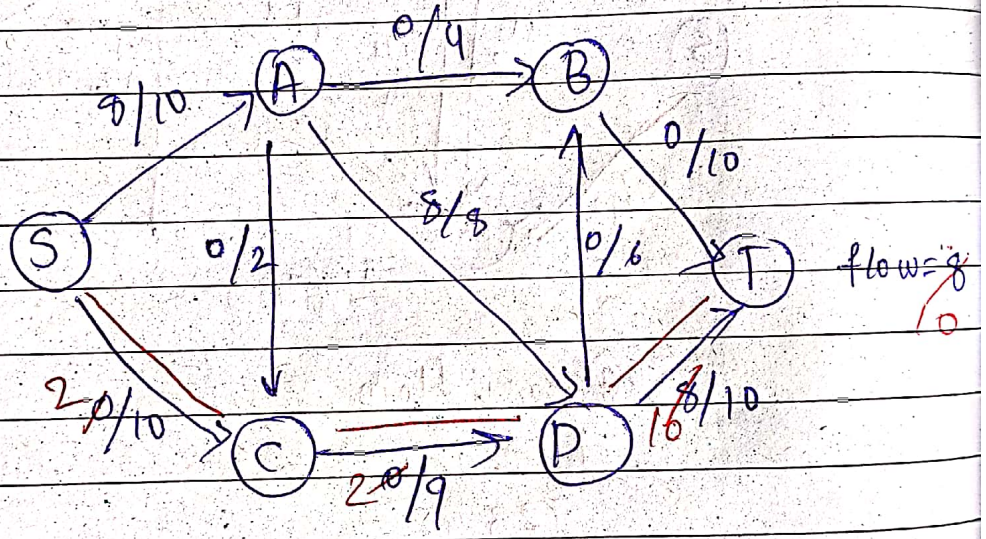
23 = 23

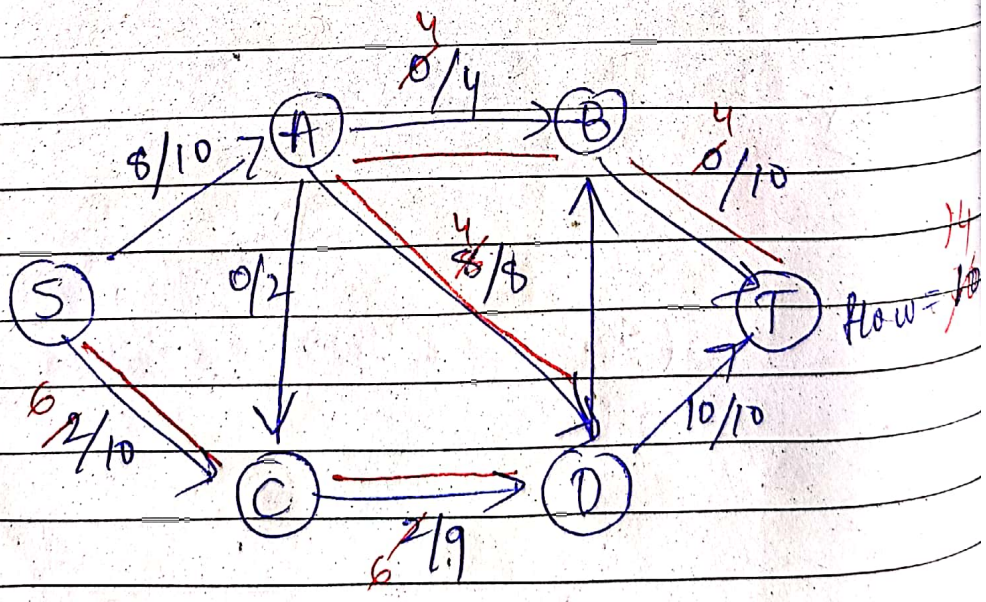
Step 1

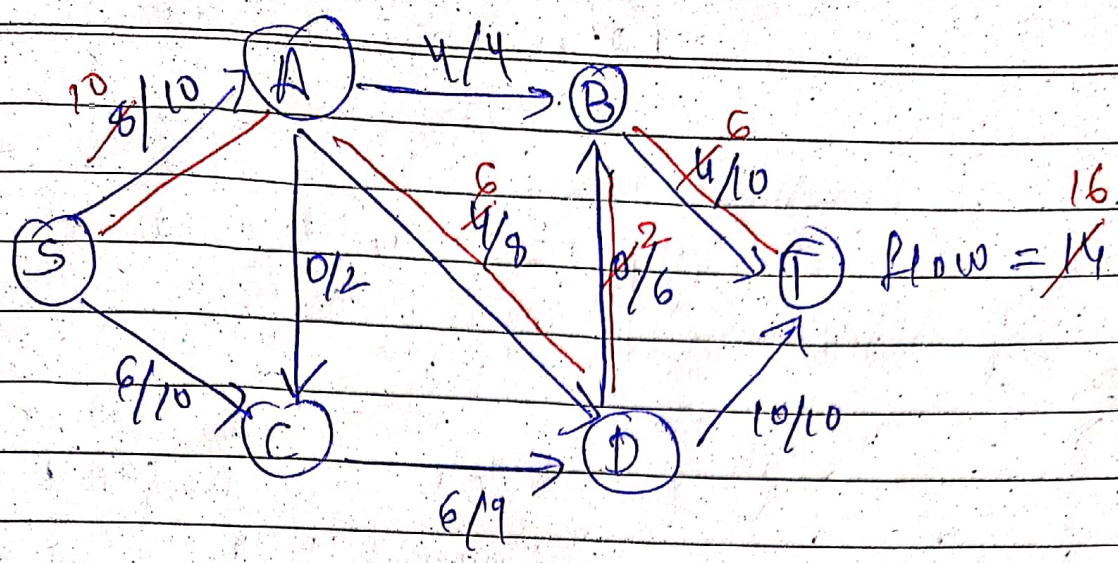
Ex:

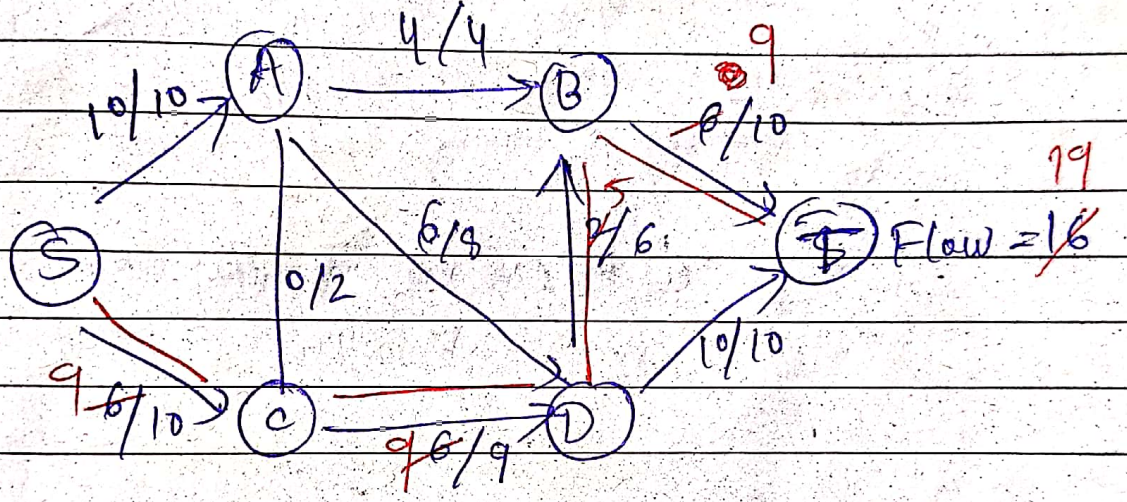


Step 2:









Augmenting path	Bottle neck Capacity
① S → A → D → T	8
② S → C → D → T	2
③ S → C → D → A → B → T	4
④ S → A → D → B → T	2
⑤ S → C → D → B → T	3

Scanned with OKEN Scanner

Polynomial Time Reduction

P → problem which can be solved in polynomial time

polynomial time

NP → problem which can't be solved in polynomial time.

Non-deterministic polynomial time

Ex: a problem within 1 hr time

depending on computing time Algorithm

polynomial time

takes less time

Exponential time

takes more time even if is small pblm

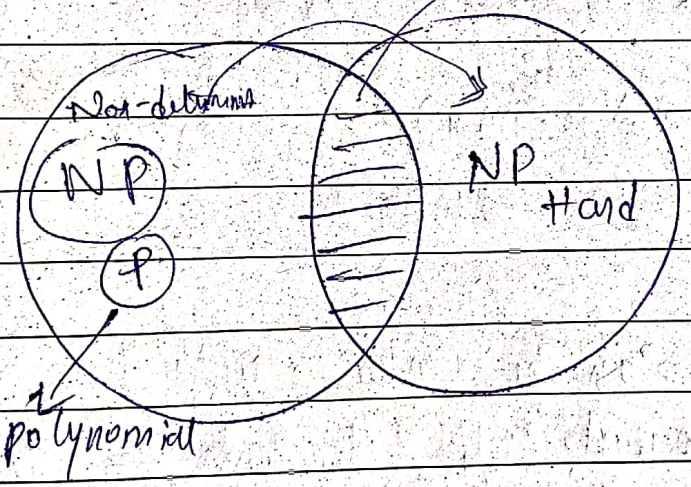
Ex: Linear Search - n
binary Search - log n

Ex: 0/1 Knapsack - 2^n
TSP - 2^n

Scanned with OKEN Scanner

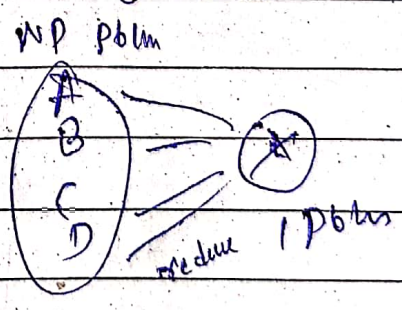
NP problems

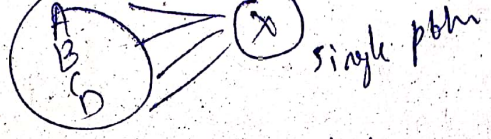
NP complete	NP Hard problem
1) A problem that is NP Complete has property that it can be solved in polynomial time if & only if all other NP complete problems can be solved in polynomial time	If an NP hard Problem can be solved in polynomial time, then all NP Complete problems Can be solved in polynomial time



or

A problem is called NP Hard if Every problem in NP can be polynomially reduced.

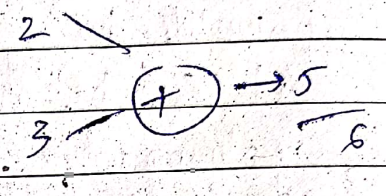




NP Hard & NP Complete

Non deterministic Algorithm (NP)

→ O/p can't be predicted properly even we know the i/p.



Reduction

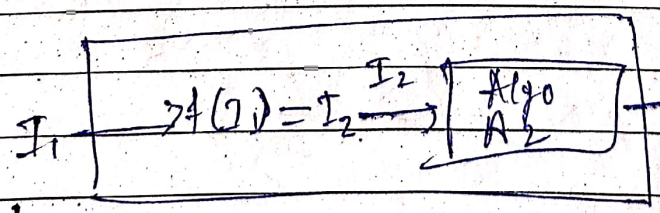
Consider we have 2 decision Pblm P_1 & P_2 → yes/No

P_1 → Input (I_1), Algo (A1) – Unknown

P_2 → Input (I_2), Algo (A2) – known

If P_1 can be solved with help of A2, then we convert i/p I_1 into I_2 and find Sol^n for P_2

→ P_1 is reducible to P_2



i/p for pblm P_1

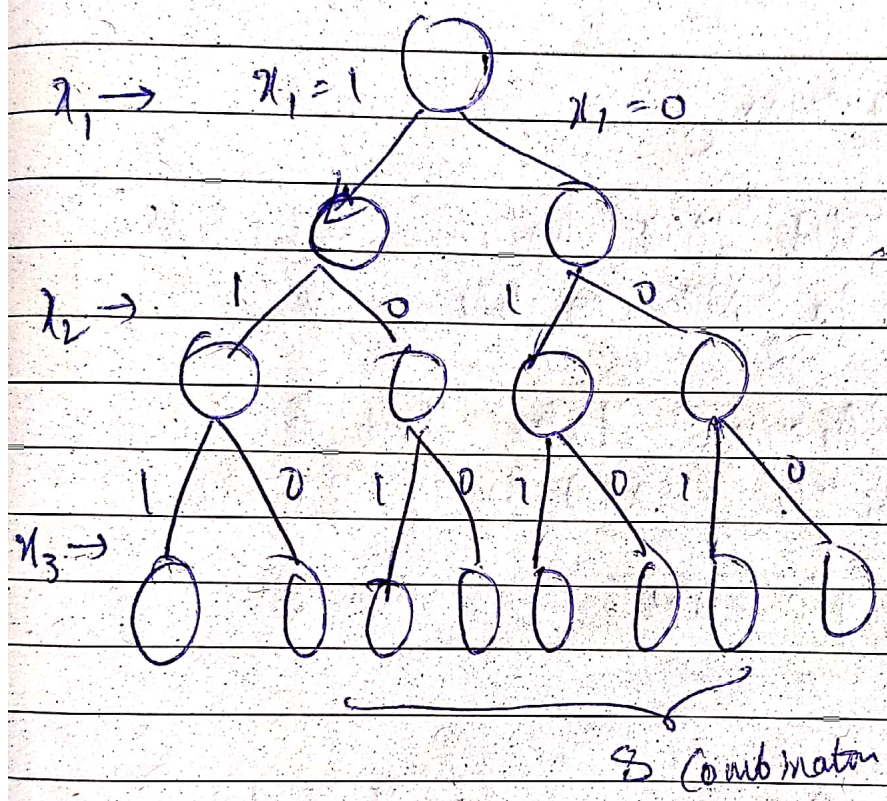
Sol^n for P_1

Satisfiability (SAT)

Ex: $(x_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee \bar{x}_3)$

$x_1 x_2 x_3 = 2^3 = 8$ combination

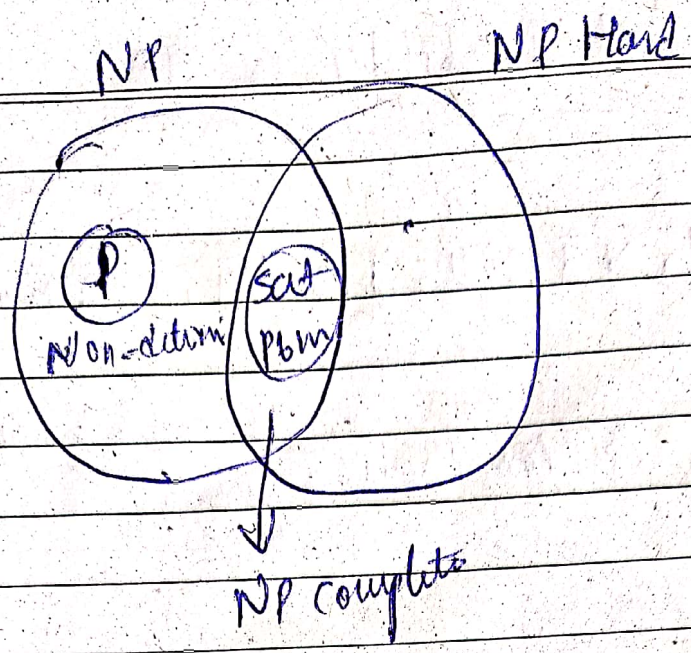
Boolean $\leftrightarrow$ State space tree



x_1	x_2	x_3
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	0

:- If the satisfiability problem is solved in polynomial time using same concept we ~~use~~ can solve other related problems too.

Reduction: deriving from 1 to another ~~Satisfiability~~ α of:



Cook's theorem

* 'Cook' Stated that boolean satisfiablity pblm is in NP complete pblm.
* 'Cook' theorem states that the Satisfiability (SAT) is in P,

if, $\boxed{P = NP}$

Ex: $(x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_3)$

let $x_1 = 1$ $x_2 = 0$ $x_3 = 0$

$\Rightarrow (1 \vee 0 \vee 1) \wedge (0 \vee 1 \vee 1)$

$1 \wedge 1 = 1$ (True)

Scanned with OKEN Scanner

Sequencing Problem : TSP

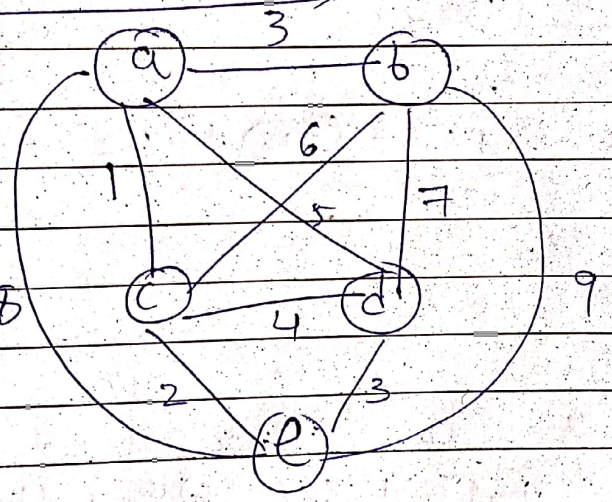
* The objective of TSP is to find a route such that a salesperson visits all the cities & returns back to the same city from where he started the tour with the minimum cost.

* TSP is a min. pblm, hence we are supposed to find the lower bound.

* The lower bound (lb) is denoted as $\left[\frac{S}{2}\right]$

S - is ~~sum~~ sum of 2 ~~closest cities of a city for all cities~~ smallest distance edges.

Compute lower bound



$$lb = \frac{\overset{a}{(1+3)}+\overset{b}{(3+6)}+\overset{c}{(1+2)}+\overset{d}{(3+4)}+\overset{e}{(2+3)}}{2}$$

$$= \left[\frac{28}{2}\right] = 14 \longrightarrow \text{initial } (Lb)$$

$$\boxed{lb = 14}$$

Initial State space Tree

Start
lb = 14

level 0

Compute lower bound from vertex 'a'

* $a \rightarrow b = \frac{\overset{A}{(3+1)} + \overset{B}{(3+6)} + \overset{C}{(1+2)} + \overset{D}{(4+3)} + \overset{E}{(2+3)}}{2}$

('3' compulsory)

$\boxed{lb = 14}$

* $a \rightarrow c = \frac{\overset{A}{(1+3)} + \overset{B}{(3+6)} + \overset{C}{(1+2)} + \overset{D}{(4+3)} + \overset{E}{(2+3)}}{2}$

('1' compulsory)

$\boxed{lb = 14}$

* $a \rightarrow d = \frac{\overset{A}{(5+1)} + \overset{B}{(3+6)} + \overset{C}{(1+2)} + \overset{D}{(5+3)} + \overset{E}{(2+3)}}{2}$

('5' compulsory)

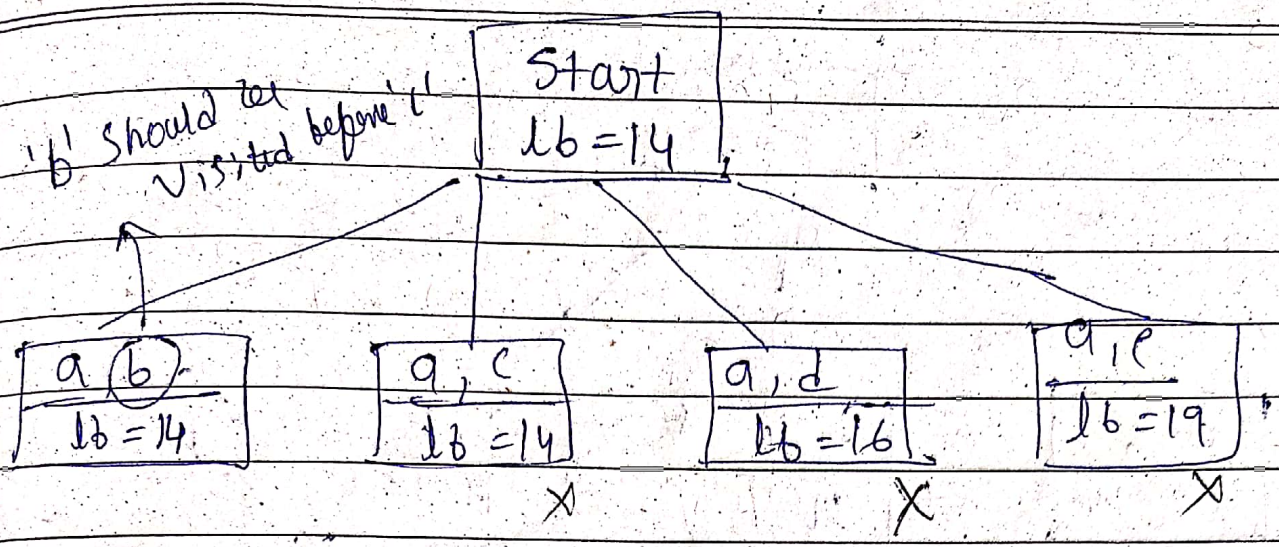
$\boxed{lb = 16}$

* $a \rightarrow e = \frac{\overset{A}{(8+1)} + \overset{B}{(3+6)} + \overset{C}{(1+2)} + \overset{D}{(4+3)} + \overset{E}{(8+2)}}{2}$

('8' compulsory)

$\boxed{lb = 19}$

Scanned with OKEN Scanner



Compute lower bound from vertex b

* $a \rightarrow b$: we already found $lb = 14$ //

* $b \rightarrow c = \dfrac{\overset{A}{(3+1)} + \overset{B}{(6+3)} + \overset{C}{(6+1)} + \overset{D}{(4+3)} + \overset{E}{(2+3)}}{2}$

$\boxed{lb = 16}$

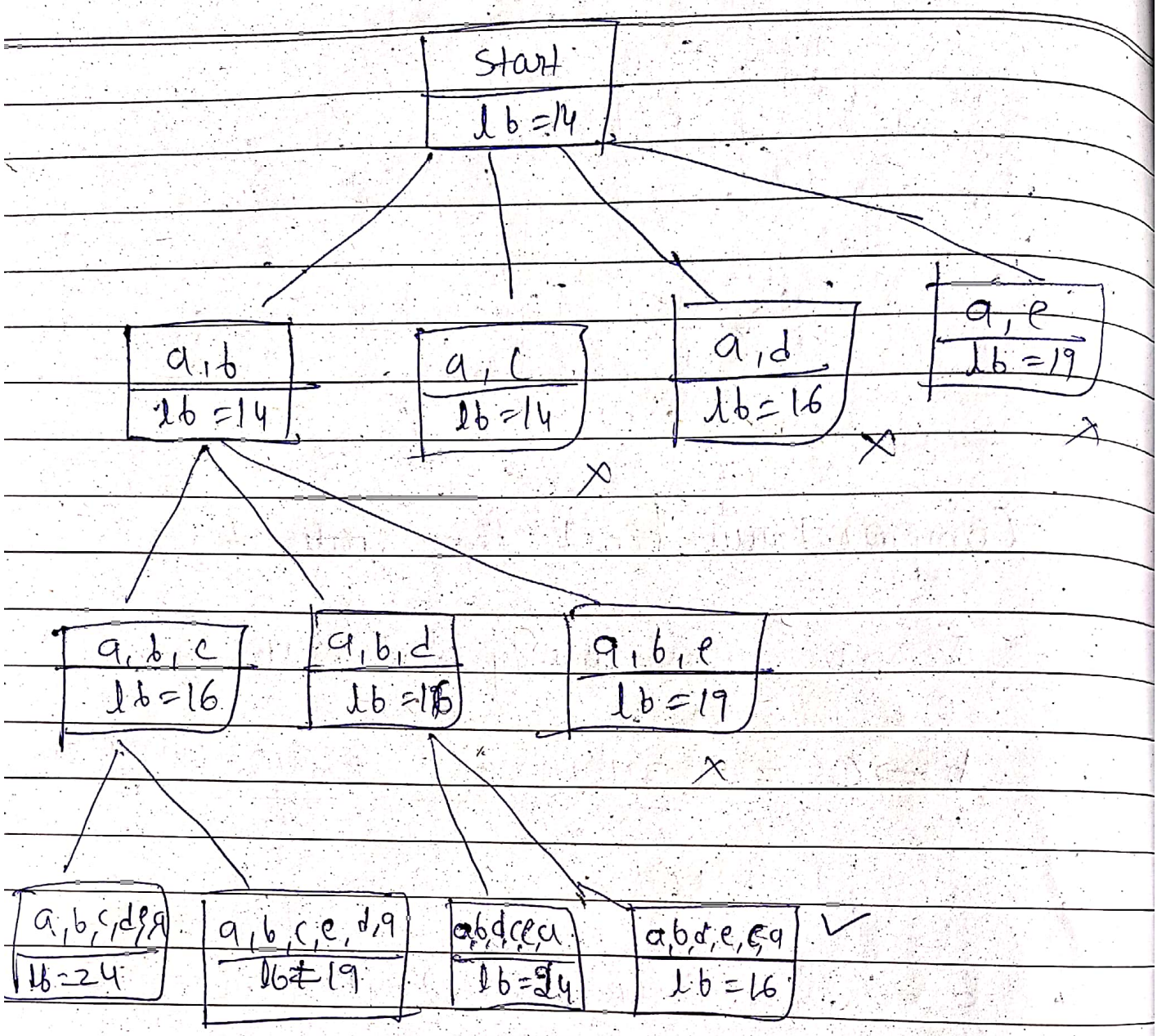
* $b \rightarrow d = \dfrac{(3+1) + (7+3) + (1+2) + (7+3) + (2+3)}{2}$

$\boxed{lb = 16}$

* $b \rightarrow e = \dfrac{(3+1) + (3+9) + (1+2) + (3+4) + (9+2)}{2}$

$\boxed{lb = 19}$

Scanned with OKEN Scanner



TSP path ⇒

$a \to b \to d \to e \to c \to a = 16$ (or) +

∴ lowest cost visit all nodes from the given graph

Compute lower bound from vertex d & e

$$a \xrightarrow{3} b \xrightarrow{6} c \xrightarrow{4} d \xrightarrow{3} e \xrightarrow{8} a$$

$$= \frac{(3+8) + (6+3) + (6+4) + (4+3) + (3+8)}{2}$$

$\boxed{lb = 24}$

Compute lower bound from vertex e & d

$$a \to b \to c \to e \to d \to a$$

$$= \frac{(3+5) + (3+6) + (6+2) + (3+5) + (2+3)}{2}$$

$\boxed{lb = 19}$

Compute lower bound from vertex c & e

$$a \to b \to d \to c \to e \to a$$

$$= \frac{(3+8) + (3+7) + (4+2) + (7+4) + (2+8)}{2}$$

$\boxed{lb = 24}$

Compute lower bound from vertex e & c

$$a \to b \to d \to e \to c \to a$$

$$= (3+1) + (3+7) + (2+1) + (7+3) + (3+2)$$

$\boxed{lb = 16}$

Scanned with OKEN Scanner

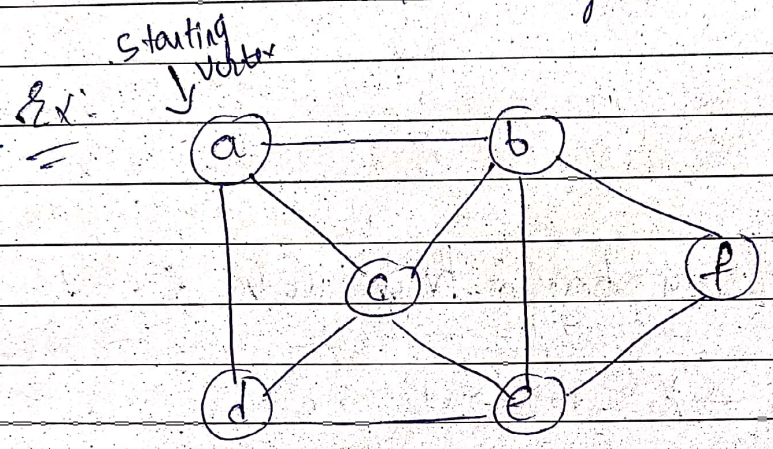
→ Undirected Graph

Backtracking - Hamiltonian Cycle Pblm

↳ Undirected graph visits each vertex exactly once returns to starting vertex

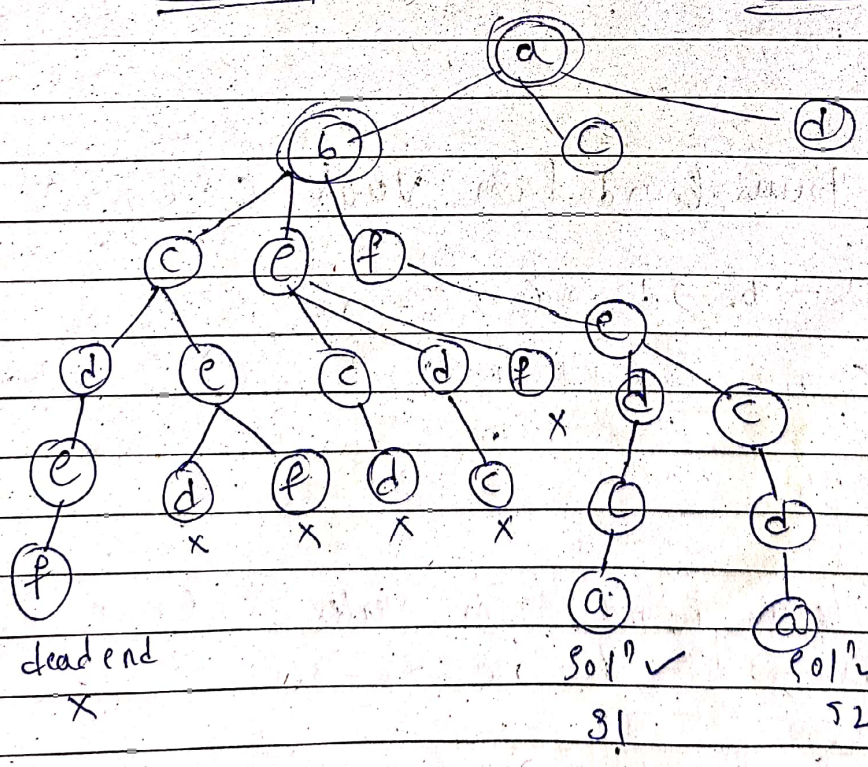
↳ similar to TSP, but here not given cost of edges.

Ex:



State space tree

a → b

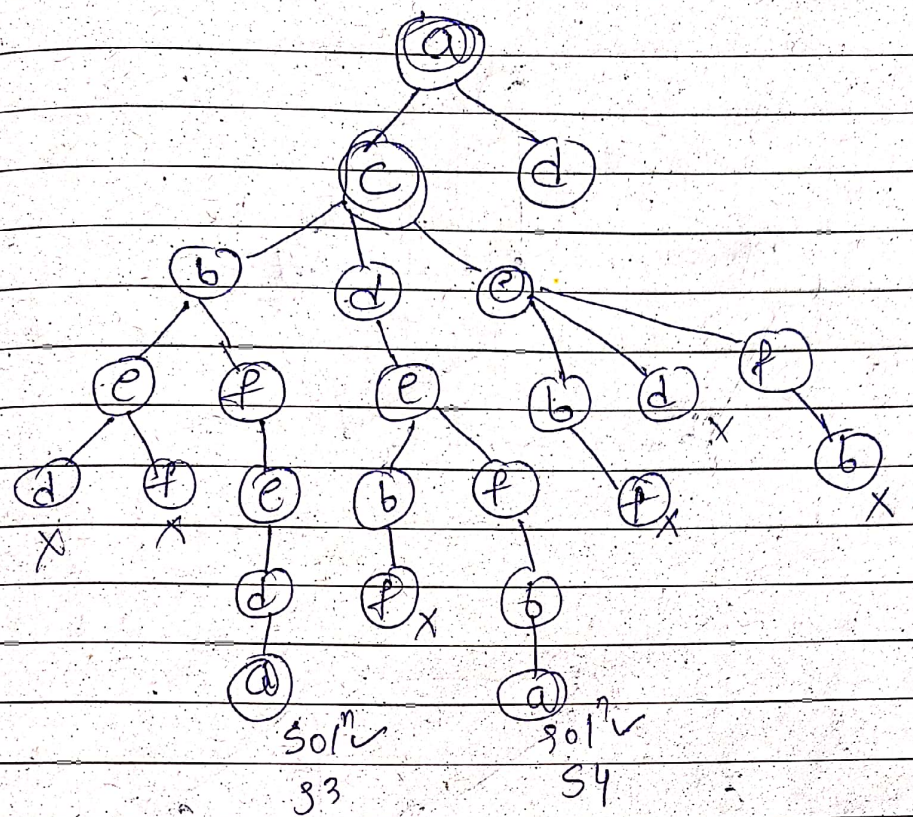


$S_1 \rightarrow a\ b\ f\ e\ d\ c\ a$
$S_2 \rightarrow a\ b\ f\ e\ c\ d\ a$ } Hamiltonian cycle

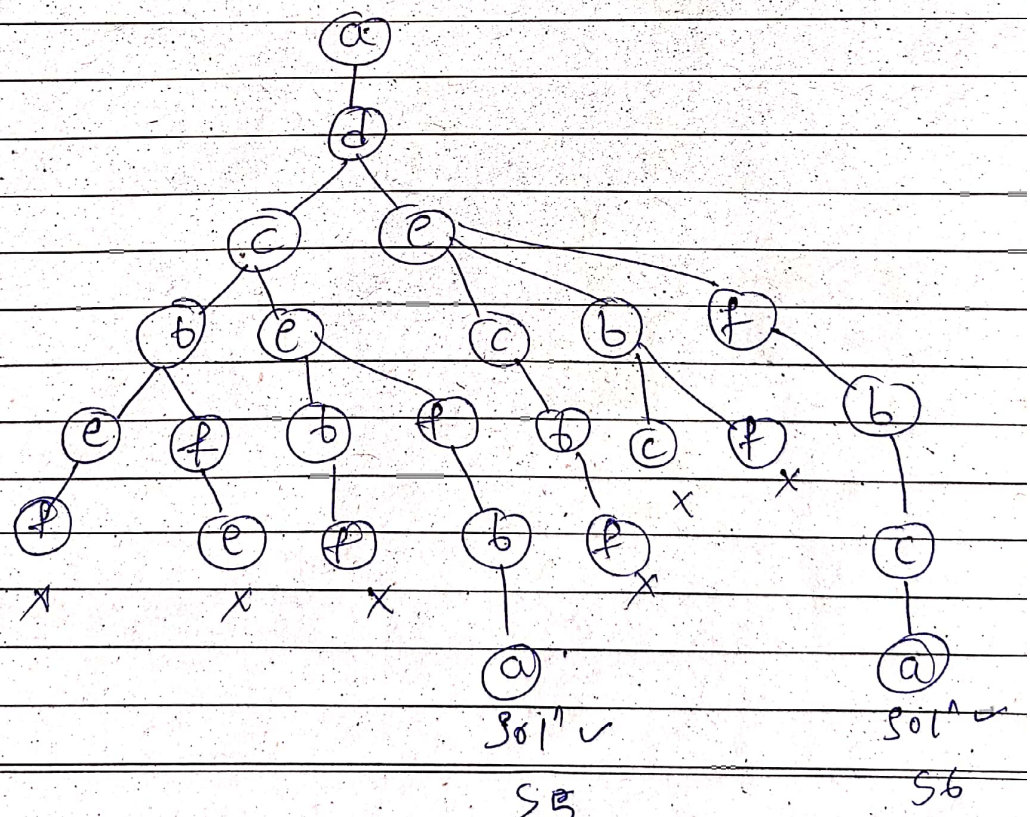
a → ..... a

$S_3 \rightarrow a\ c\ b\ f\ e\ d\ a$
$S_4 \rightarrow a\ c\ d\ e\ f\ b\ a$ } HC

$a \rightarrow c$



$a \rightarrow d$



$S_5 \rightarrow a\ d\ e\ f\ b\ c\ a$
$S_6 \rightarrow a\ d\ c\ e\ f\ b\ a$ } HC

Scanned with OKEN Scanner