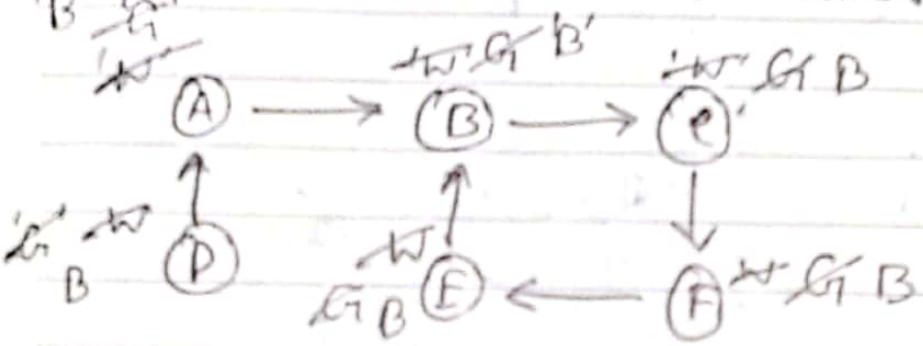
Date:

TECHNO Printing & Packaging

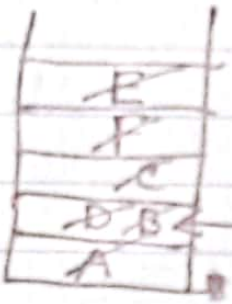
Cycle detection of directed graph →

flag → 'W' → Unvisited
'G' → Visited but adj unvisited / pushed
'B' → Whole visited / pop



Cycle = 0

stack →



visited → A, D, B, C, F, E

Empty

'B' was in stack

from 'E' to 'B' and B = 'G', cycle += 1

cycle = 1

for undirect graph →

All same

Condition → Par = {'A': 0, 'B': 'A', 'C': 'B', 'F': 'C', 'E': 'F', 'D': 'A'}

Next node = 'Gray' and par[next_node] = cur_node

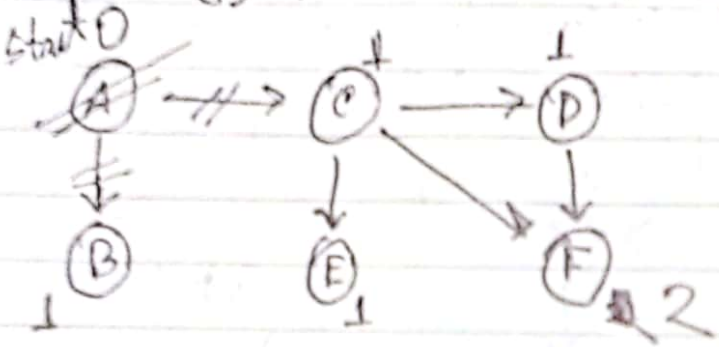
Date:

condition → (i) directed graph
(ii) No cycle

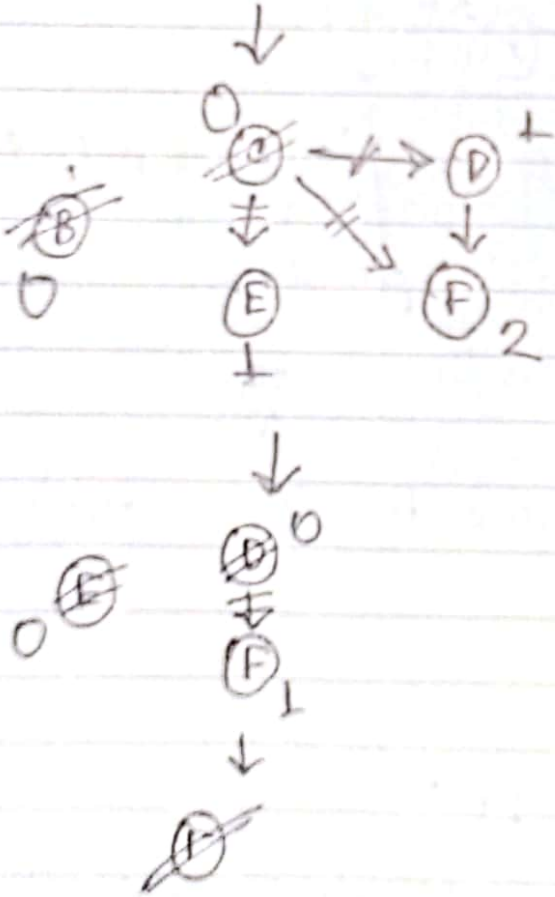
TECHNO Printing & Packaging

Topological sort :-

(i) indegree every vertex
(ii) Print if indegree = 0



Sort → A B C E D F



Indegree, if any node has directed towards that

Code →
DFS inside BFS

Time complexity →
O(V+E)

Date:

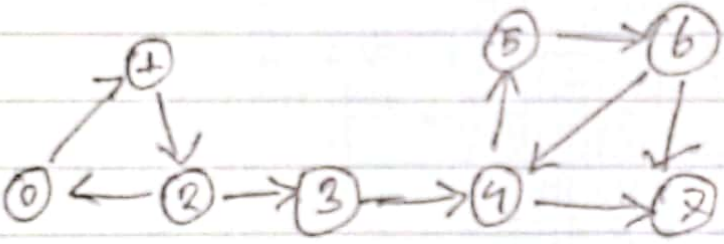
Time Complexity → $O(V+E)$

TECHNO Printing & Packaging Limited

Strongly connected components → (Kosaraju algo)

> Can reach any component to any comp are strongly connected component.

(i) DFS and Push to stack before return
(ii) Reverse all the edge
(iii) Pop from stack and DFS on modified graph
↳ each succesful dfs give a SSC

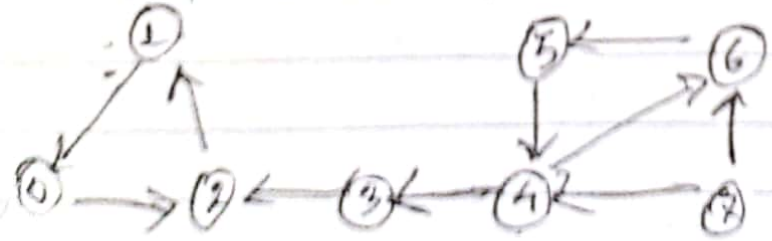


visited →

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

stack →

7	6	5	4	3	2	0	1



SSC → 1, 0, 2, 3 | 4, 6, 5, 7

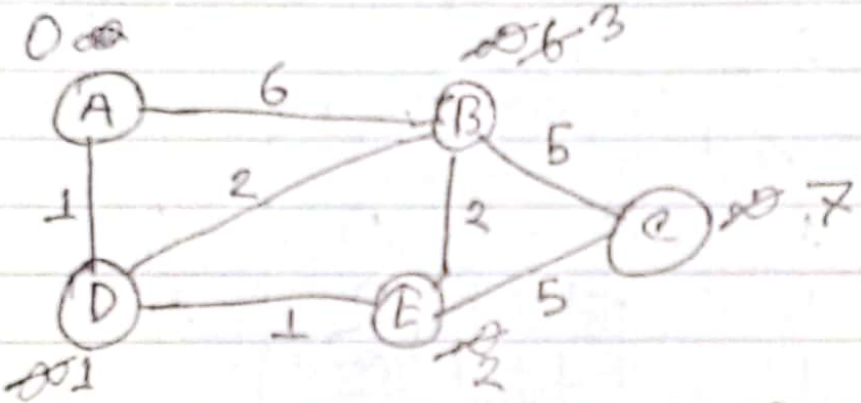
Date:

TECHNO Printing & Packaging Limited

Time Comp → $O((E+V)\log V)$

Dijkstra Algo →

if cost(curr_node) + edge < dist[curr_node]
change → dist[cost + edge]
else →
Remain same



	A	B	C	D	E
dist →	0	3	7	1	2
Prev →	A	D	E	A	D
visited →	T	T	T	T	T

[dist → 0, 3, 7, 1, 2]

(Ans)

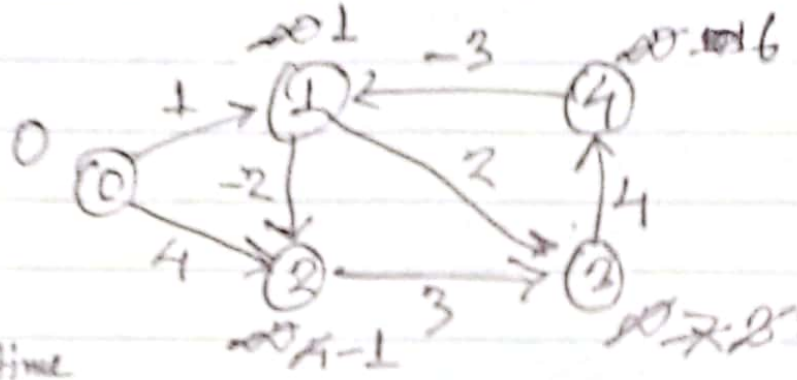
Date:

Time Comp → $O(V*E)$

TECHNO Printing & Packaging

Bellman-ford →

Run dijkstra for (V-1) times and within that we can find shortest path



V = 5
V-1 = 4
4 time dijkstra

How many time

times	0	1	2	3	4
1	0	1	-1	2	6
2	0	1	-1	2	6
3	0	1	-1	2	6

← Values are same and break

Update the dist in this edges

- (0,1)
- (0,2)
- (1,2)
- (1,3)
- (2,3)
- (3,4)
- (4,1)

Date:

Time Comp → $O(E \log V)$

MST → Kruskal algo

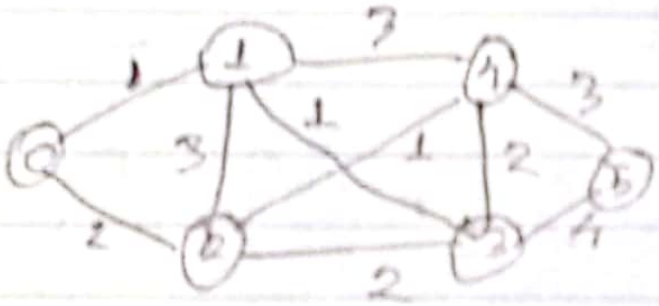
(i) Sort all edge with weights increasing order

(ii) Run (V-1)

↳ Pick smallest edge

↳ Does it creates a cycle in new graph

- if yes then, discard it
- else, include this

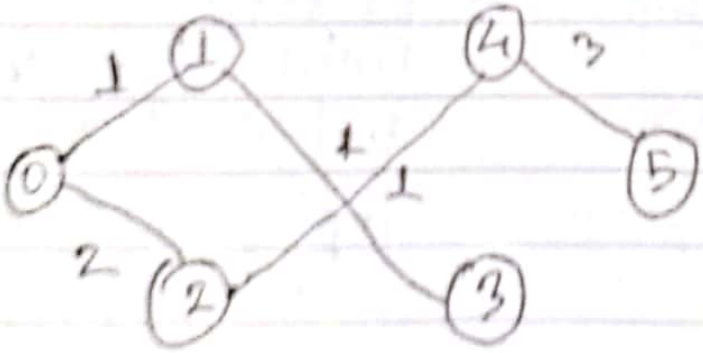


V-0 = 6

V-1 = 5

(0,1)	→ 1
(1,3)	→ 1
(2,4)	→ 1
(0,2)	→ 2
×(2,3)	→ 2
×(3,4)	→ 2
×(1,2)	→ 3
×(1,4)	→ 3
(4,5)	→ 3 ← Stop (V-1)
(3,5)	→ 4

MST =



Date:

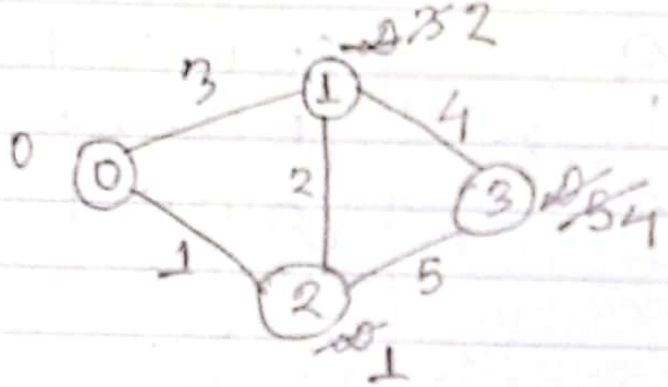
time comp- $O(V^2)$

MST → Prim's algo

Repeat this until All vertex in set

(i) Select node with min weight
(ii) Include selected node in set MST
(iii) Compute all adj edges

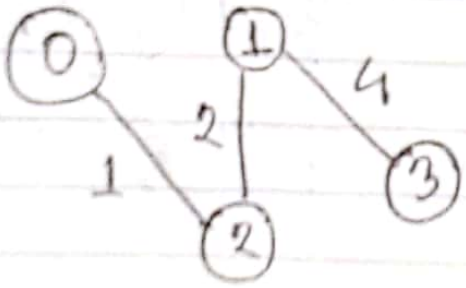
↳ Repeat 1 → 2 → 3 unless all vertex are include in MST



set MST

0, 2, 1, 3

MST



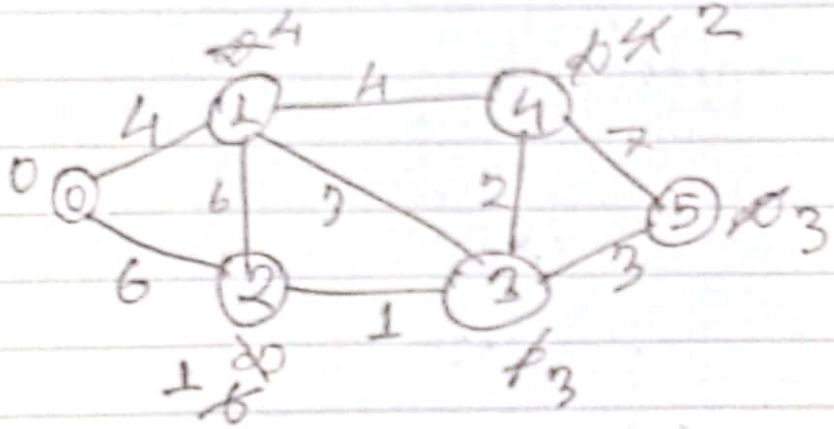
← Selecting the min weight from all adj edges
updated weight = curr_weight

Date:

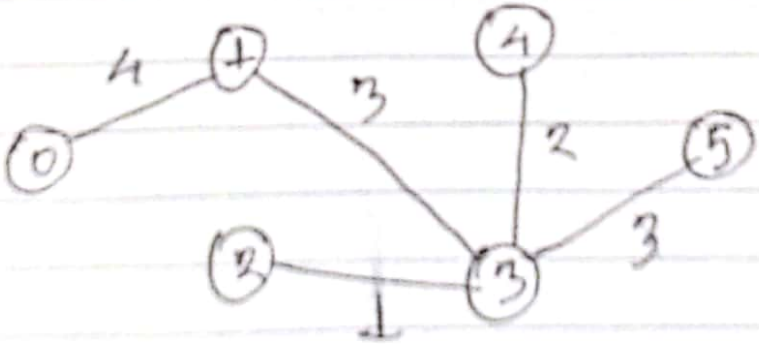
TECHNO Printing & Packaging

	0	1	2	3	4	5
Dist →	0	~~∞~~ 4	~~∞~~ 6	~~∞~~ 3	~~∞~~ 4	~~∞~~ 3
Set MST →	T	T	T	T	T	T
Parent →	-1	0	3	1	3	3

← connection array



MST



Date:

TECHNO Printing & Packaging

Fractional Knapsack →

Thief stealing the maximum items with less weights.

(i) We can find the unit of the product
(ii) Sorting (decending on increasing)
(iii) Pick the amounts

Date:

TECHNO Printing & Packaging

Huffman coding →

(i) compression technique
(ii) Used for reducing the size of data

Process →

(i) Take frequency of each bit and sort the order in increasing order
(ii) 2^n character can be represented and n = number of bits (we will decide an n)
(iii) Make a tree (min-heap)
(iv) Left side - 0 | Right side - 1
(v) traverse and get the code

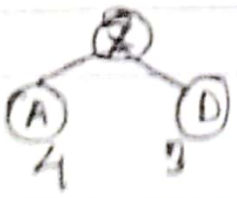
Message - AAAABCBCDDDCBCBB

char	freq	coding
B	5	0
C	4	10
A	4	110
D	3	111

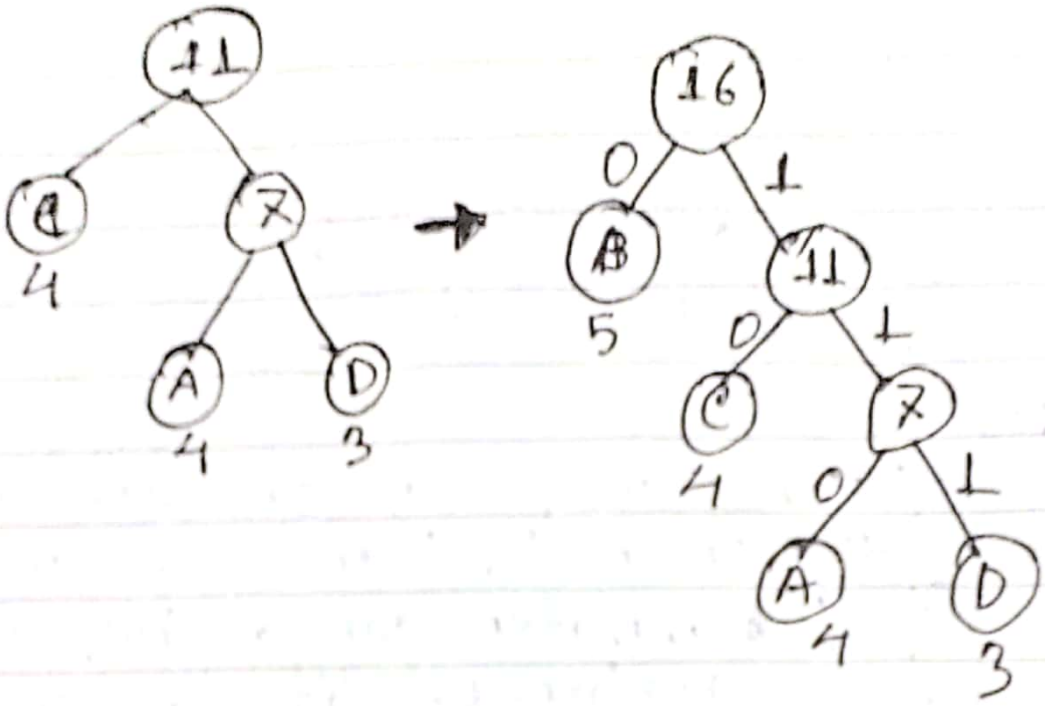
char bit → 4 x 8
freq bit → 5+4+4+3
coding → 1x5+
2x4+
3x4+
3x3+

total bit -

B C A D
5 4 4 3



Date:



Same this tree is used to decode

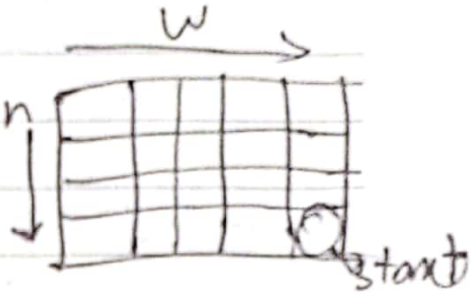
Date:

TECHNO Printing & Packaging Limited

0/1 Knapsack DP →

(i) Element → weights (w)
value (n) take on not

step → (i) ~~Table~~ take table
→ ~~dp[][]~~
dp[n+1][W+1]



(ii) start from top down
(iii) n=0 and w=0 will be zero
(iv) when we are at ith row will consider previous rows also.

Ex

After weights all weights are same

→ bag-weight ↓

P	W		0	1	2	3	4	5	6	7	8
		0	0	0	0	0	0	0	0	0	0
1	2	1	0	0	1	1	1	1	1	1	1
2	3	2	0	0	1	2	2	3	3	3	3
5	4	3	0	0	1	2	5	5	6	7	7
6	5	4	0	0	1	2	5	6	6	7	8

bag-weight = 8

Profit = {1, 2, 5, 6}
weight = {2, 3, 4, 5}
n = 4

Date:

TECHNO
Printing • Packaging

Pseudo code →

```
if i == 0 or j == 0:
    table[i][j] = 0
elif wt[i-1] <= jj:
    table[i][j] = max(val[i-1] + table[i-1][w-wt[i-1]]
                      , table[i-1][j])
else:
    table[i][j] = table[i-1][j]
```

Date:

Time Comp → $O(m \times n)$

TECHNO Printing & Packaging

LCS → Longest common subsequence

~~Longest~~ LCS is a subset can be derivate from another string by deleting some element without changing order in a specific length.

str1 → AABCDEEH

Length	LCS
4	ACDE / ABCH / . . . .
5	ABCDE / AABCH / . . . .
3	BCE / DEE / AEH / . . . .

(i) Take table (m x n) m = len of str1, n = len of str2

(ii) Inorsially, m=0 and n=0, is 0.

(iii) If str1[i] == str2[j]

table[i][j] = 1 + table[i-1, j-1]

else,

table[i][j] = max (table[i-1][j], table[i][j-1])

str2

		ⓐ B'	ⓑ 1(a)	2(b)	3(c)	4(d)
	0	0	0	0	0	0
str1	1(b)	0	0	1	1	1
	2(d)	0	0	1	1	(2)
				b		d

str1 = | b | d |

str2 = | a | b | c | d |

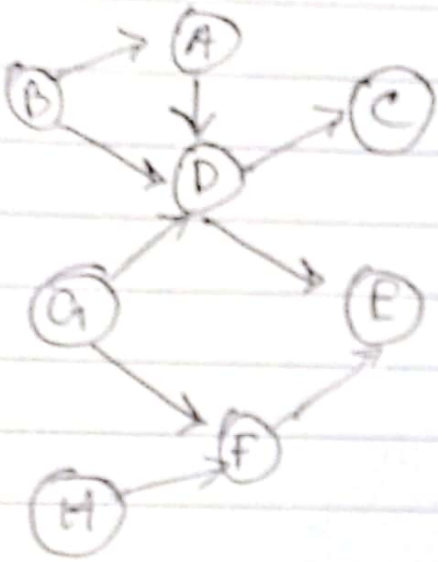
Date:

The peek arrow is LCS(↖) = bd
Top down/bottom = 2 = length of LCS

This is called top down process is to calculate top to bottom and the top ↓ bottom is answer.

Topological sort with dfs →

Order [u → v] all vertices will maintain this.



stack = | C | E | D | A | B | F | G | H |

dfs-stack = | B | A | D | C |

visited = | B | A | D | C | G | E | F | H |

top-sort = T T T T T T T T

Top-sort → HGFBADEC