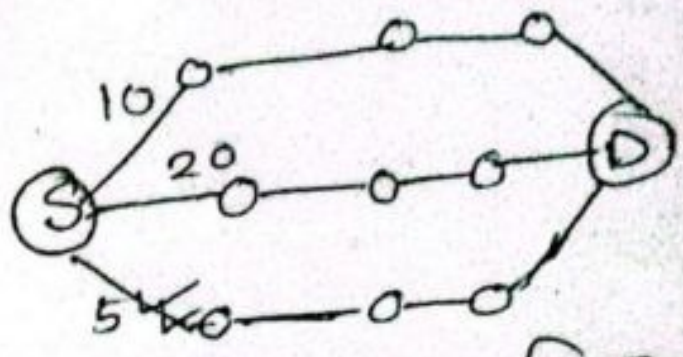
Greedy Algorithm

Pseudocode

~~Note~~ -

```
Algorithm Greedy(a,n)
{
  for i=1 to n do
  {
    x=select(a);
    if Feasible(x) Then
    {
      solution = solution + x;
    }
  }
}
```



cost ~~এখন~~ যেদিকে যাবে সেদিকেই যাবে and সেটাই solution হবে।

Advantages

* It is an algorithm that finds a solution to a problem in the shortest time possible
* It involves making the locally optimal choice at each stage with the hope of finding the global optimal.
* Easy to implement and understand
* Feasible solution
* optimal solution (Min cost, Max Profit, Min Risk)

CamScanner

Yamina Mam

* Top down approach

*Greedy techniques Vs Dynamic Programming

Greedy Method	Dynamic Programming
1) Make our decisions based on the best current situation	1) we select individually in every step
2) There is no assurance of obtainning the optimal solution	2) get the assurance of an obtainning the optimal solution
3) follows Top-down approach	3) Follows both bottom-up and top-down.
4) No chance of backtracking thus making it more efficient in terms of memory	4) Uses memorization due to which the memory complexity increases and making it less efficient.
5) Faster than dynamic	5) Comparatively slower.

* Applications —

(i) Knapsack problem
(ii) Minimum spanning tree
(iii) Activity selection problem
(iv) Huffman coding
(v) Dijasfran's algorithm

Greedy characteristics:

(i) local optimicity
(ii) not back tracking
(iii) not always the best

Activity Selection Problem

→ minimum numberc of activities sleeted

steps

(i) sort the finishing line
(ii) find compatible activity and add to solution

s = starting
f = finishing

Example

	a_1	a_2	a_3	a_4	a_5	a_6	a_7
s_i	2	1	4	1	9	8	5
f_i	5	8	7	3	11	10	9

এখানে finishing ~~time~~ index ক আগে sort করবো

	a4	a2	a3	a1	a5	a6	a7
si	1	1	4	2	9	8	5
fi	3	5	7	8	9	10	11

হবে না কারণ finishing line এর index এর starting line এর index pair ও আসতে হবে এর position ও পরিবর্তন হতে হবে।

	a4	a1	a3	a2	a7	a6	a5
si	1	2	4	1	5	8	9
fi	3	5	7	8	9	10	11

এর minimum costing এর কারণে

মনে রাখতে হবে finishing index এ number ও end হবে তার পাশে যেকোনো number থেকে starting index start হবে। যেমন এখানে $(S_1, f_1) = (1, 3)$ দেওয়া হয়েছে তাই $(S_2, f_2) = (2, 5)$ হতে পারছে না। $S_2 = 4 + হতে$ হবে। যেহেতু হয় নি তাই আমরা এই (S_2, f_2) নেব না solution এ

$A = \{a_4, a_3, a_6\}$

Pseudocode

Greeed activity (si, fi)

n ← length (s)

A ← {1}

j ← 1

for i ← 2 to n

do if si ≥ fj

then A ← A ∪ {i}

j ← i

return A;

} O(n)

Time complexity

T = O(n) + O(n log n)

for sorting algorithm

A.3)

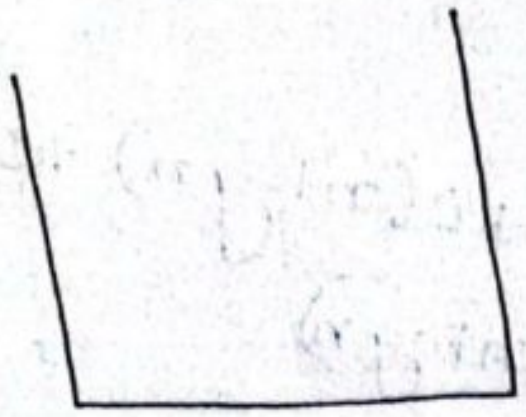
[Greedy T.C = O(n log n)]

knapsack problem

object	1	2	3
p profit	25	24	15
w weight	18	15	10

$\frac{p}{w}$ = 1.3 1.6 1.5

Suppose,



M > 0 and wi ≤ M
M = M - wi;
p = p + pi;
if (M > 0)
p = p + pi × $(\frac{M}{wi})$

① profit মাত্র বেক্সি = $25 + \frac{2}{15} \times 24 = 28.2$ ✗

② weight মাত্র বক্স = $15 + \frac{10}{15} \times 24 = 31$ ✗

③ Greedy about both = $24 + \frac{5}{10} \times 15 = 39$ ✓

Pseudocode:

for (i = 1 to n) } O(n)
calculate $\frac{p}{w}$

→ sort objects in decreasing order of $\frac{p}{w}$ ratio → O(n log n)

→ for i=1 to n
  if (M>0 and wi ≤ M)
    M = M - wi;
    P = P + Pi;
  if (M>0)
    P = P + Pi x (M / wi)

} O(n)

Time complexity T = O(n) + O(n log n) + O(n)
                  = O(n log n)

CamScanner

Algo

25-th sep

0/1 Knapsack Problem

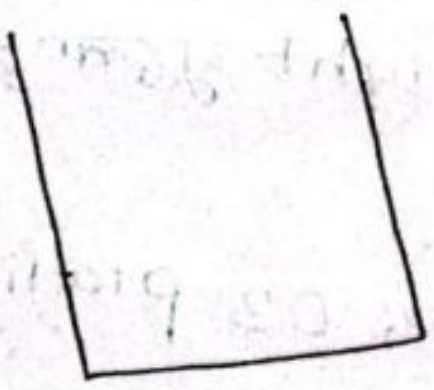
(0-Absent) (1-present)

	01	02	03
P	20	25	60
w	2	4	8

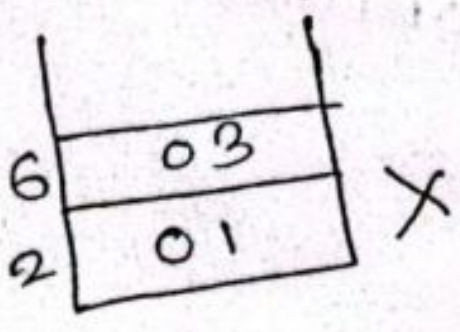
$P/w = \frac{20}{2} = 10$ $\quad \frac{25}{4} = 6.2$ $\quad \frac{60}{8} = 7.5$

~~পাবে~~ X materials ~~নাথতে~~ পারো

Suppose



যদি পাবে 01 column এর weight রাখি তাহলে



X

CamScanner

By using dynamic method it should be solved.

0 → not count
1 → count

উদাহরণ chart ক

	Total profit / ~~weight~~
0 0 0	✗ → X
0 0 1	→ 60
0 1 0	→ 25
0 1 1	→ 60+25=85 ✓ highest
1 0 0	→ 20
1 0 1	→ 20+60=80
1 1 0	→ 20+25=45
1 1 1	→ ~~80+~~ X

এখন 60, 25 profit এর weight গুলো sum করলে পাই 8+4=12

তাই উপরোক্ত জায়গে 02 এবং 03 profit যুক্ত materials দিয়ে fill up করা যাবে।

Question কোথায় greedy algorithm fail here?

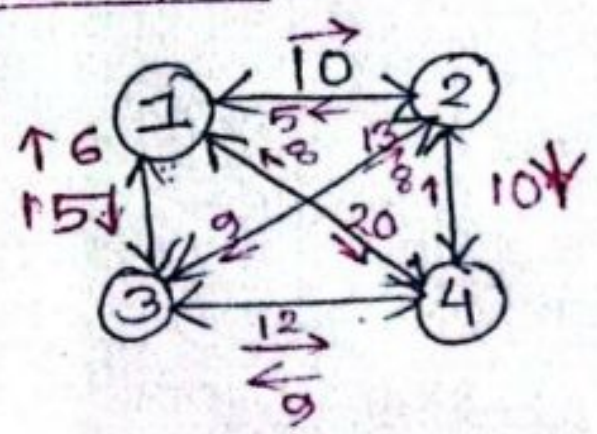
Dynamic

TSP (Travelling Sales Person Problem)

Question Graph / Matrix ~~দুইভাবেই~~ আসতে পারে। Graph দেওয়া থাকলে তা matrix এ রূপান্তর করে লিখলে easier হবে।

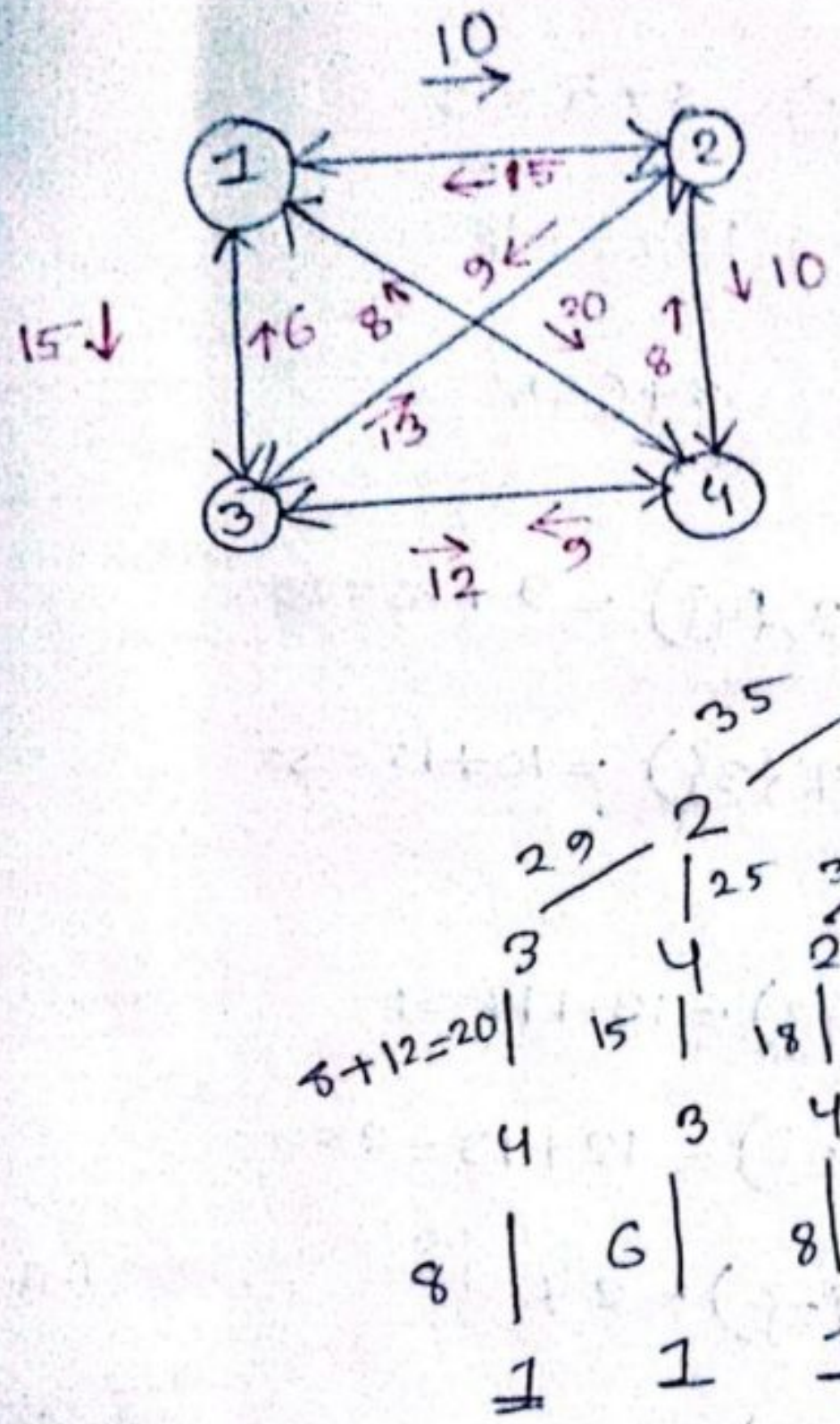
	1	2	3	4
1	0	10	15	20
2	5	0	9	10
3	6	13	0	12
4	8	8	9	0

Example



Formula $g(i,s) = \min_{j \in s} \{c_{ij} + g(j, s-\{j\})\}$

$g(i,s)$ is the shortest path ~~starting from~~ 1 and going through all the vertex in s and terminate at 1



$1 \rightarrow 2 \rightarrow 4 \rightarrow 3 \Rightarrow$

$g(4, \phi) = c_{41} = 8$

$g(2, \phi) = c_{21} = 5$

$g(3, \phi) = c_{31} = 6$

$g(3, \{4\}) = c_{34} + g(4, \phi) = 12 + 8 = 20$

$g(4, \{3\}) = c_{43} + g(3, \phi) = 9 + 6 = 15$

CamScanner

$$g(2,\{4\}) = c_{24} + g(4,\phi) = 10+8 = 18$$

$$g(4,\{2\}) = c_{42} + g(4,\phi) = 8+5 = 13$$

$$g(3,\{2\}) = c_{32} + g(2,\phi) = 13+5 = 18$$

$$g(2,\{3\}) = c_{23} + g(3,\phi) = 9+6 = 15$$

$$g(2,\{3,4\}) = \begin{cases} c_{23} + g(3,\{4\}) = 9+20 = 29 \\ c_{24} + g(4,\{3\}) = 10+15 = 25 \end{cases}$$

$$g(3,\{2,4\}) = \begin{cases} c_{32} + g(2,\{4\}) = 13+18 = 31 \\ c_{34} + g(4,\{2\}) = 12+13 = 25 \end{cases}$$

$$g(4,\{3,2\}) = \begin{cases} c_{43} + g(3,\{2\}) = 27 \\ c_{42} + g(2,\{3\}) = 23 \end{cases}$$ Min

$$g(1,\{2,3,4\}) = \begin{cases} c_{12} + g(2,\{3,4\}) = 5+25 = 35 \\ c_{13} + g(3,\{2,4\}) = 15+25 = 40 \\ c_{14} + g(4,\{2,3\}) = 20+23 = 43 \end{cases}$$

Ans:. 35

Huffman coding (Greedy)

	A	B	C	D	E	F
Freequency	45	13	12	16	9	5
Fixed length	000	001	010	011	100	101
Huffman approach	0	101	100	111	1101	1100

100×3 = 300

100×3 = 300

$(45\times1)+(13\times3)+(12\times3)+(16\times3)+(9\times4)+(5\times4)$

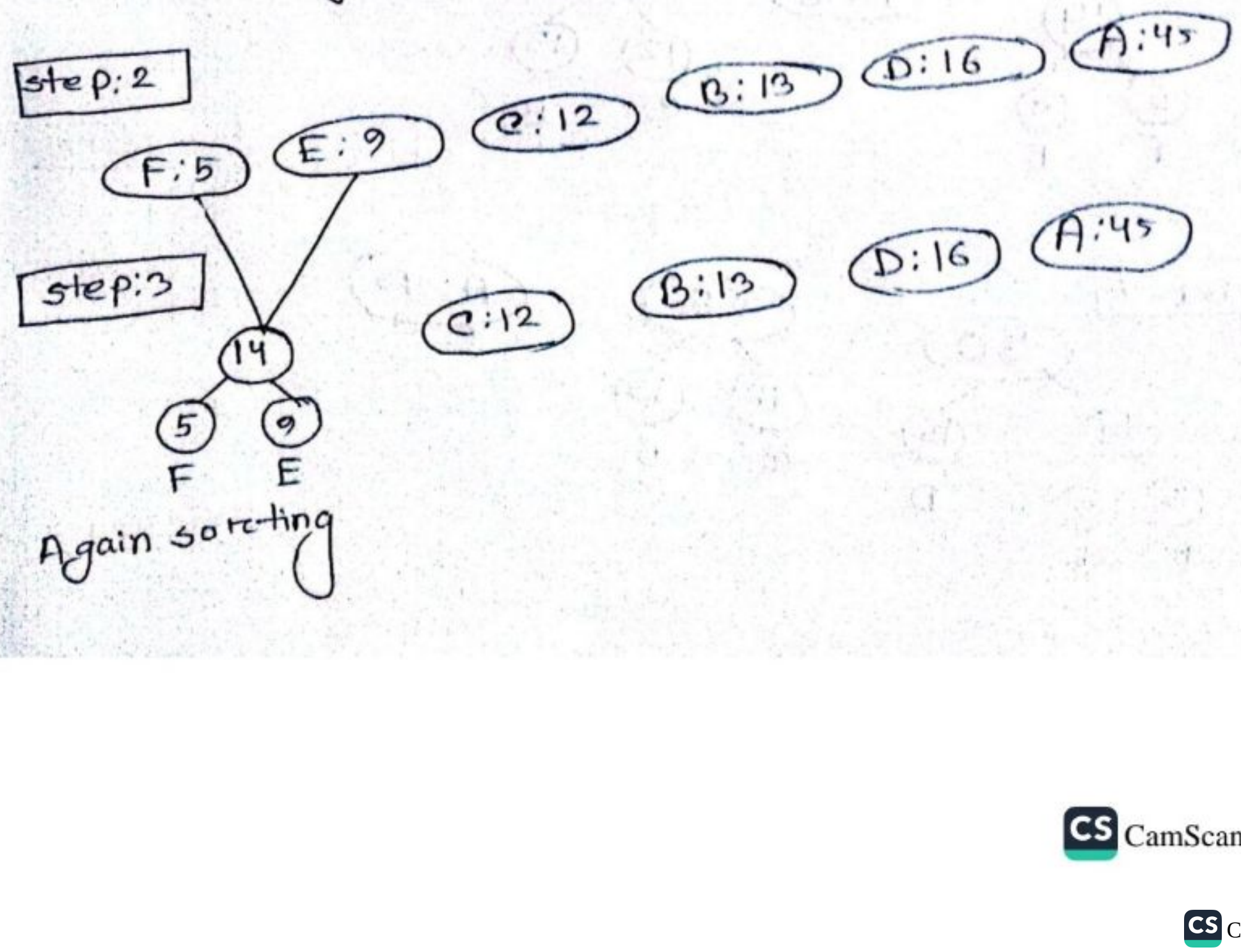
$= 224$

step: 1

A: 45 B: 13 C: 12 D: 16 E: 9 F: 5

After sorting

step: 2



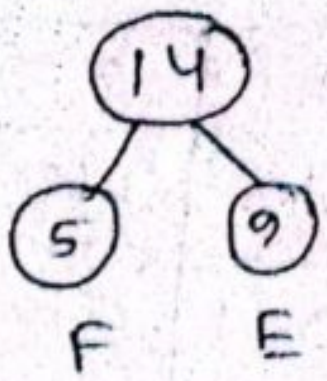
Again sorting

CamScanner

Step: 4

C: 12

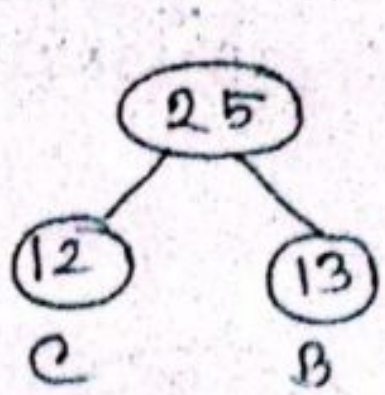
B: 13

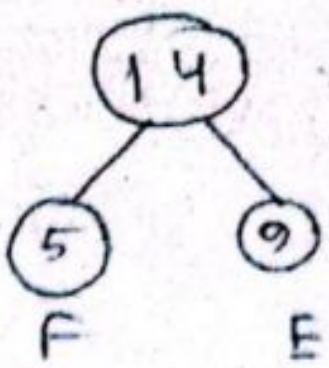


D: 16

A: 45

Step: 5



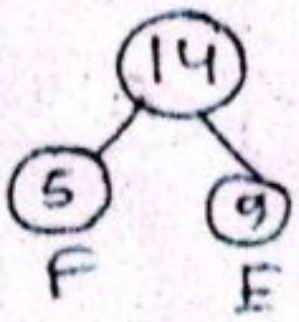


D: 16

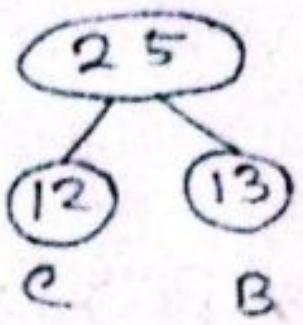
A: 45

Sorting

Step: 6

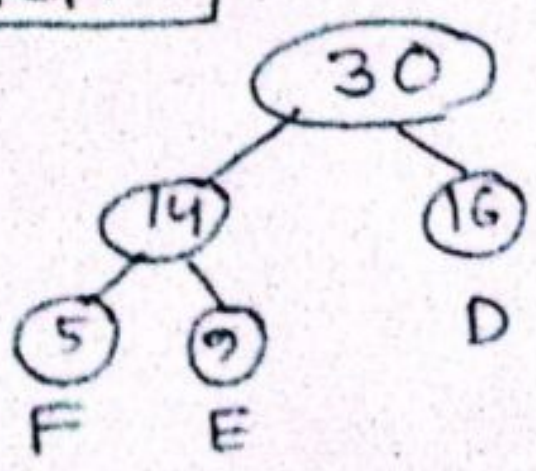


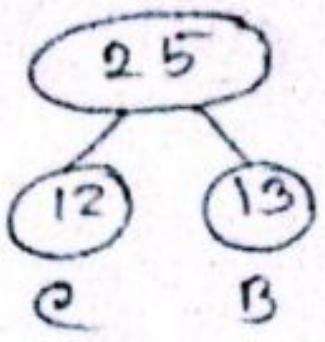
D: 16



A: 45

Step: 7

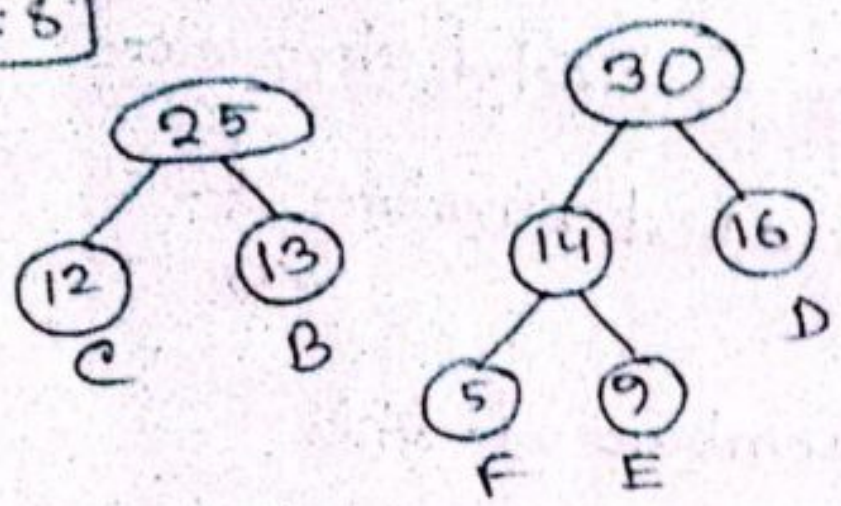




A: 45

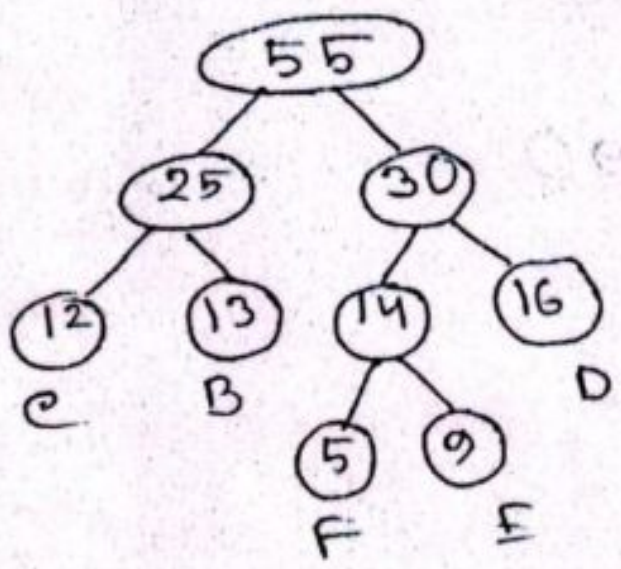
Sorting

Step: 8



A: 45

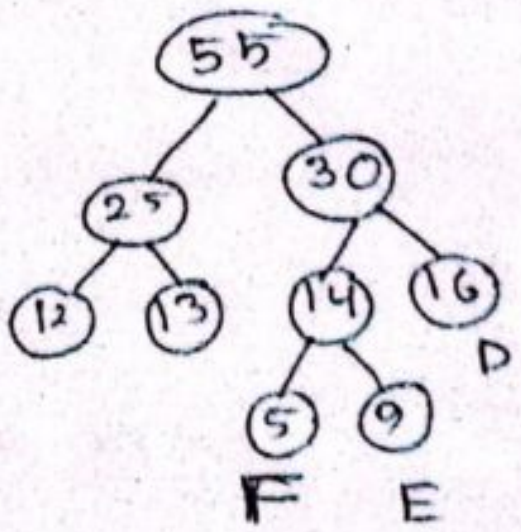
Step: 9



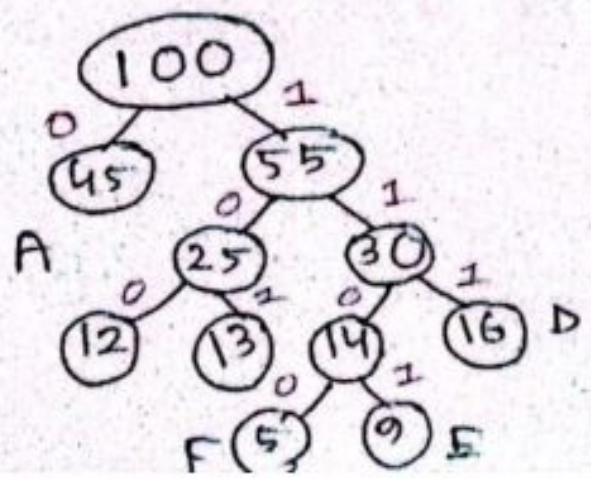
A: 45

Step: 10

A: 45



Step: 11



CamScanner

Q. Receiver এ মোট Bit কত?

Total bit = Total size + Total characters
x actual numbers of bits

+ Total number of bits

Fixed = 300 + 6x8 + 18 = 366

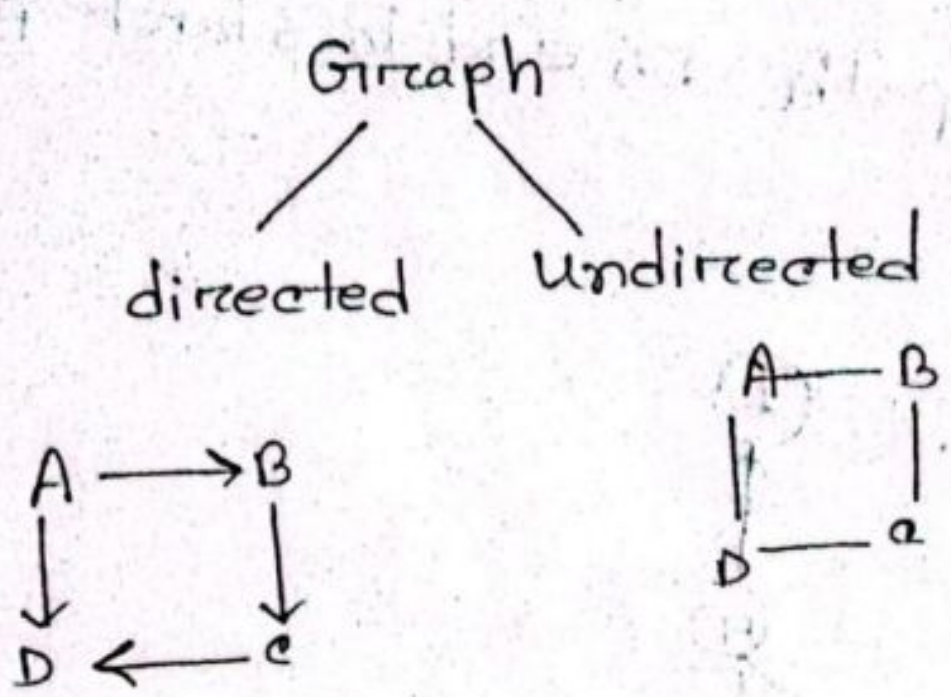
Huffman = 224 + 6x8 + 18 = 290

See

Segment-5

27 Sep

Graph



Adjacent:

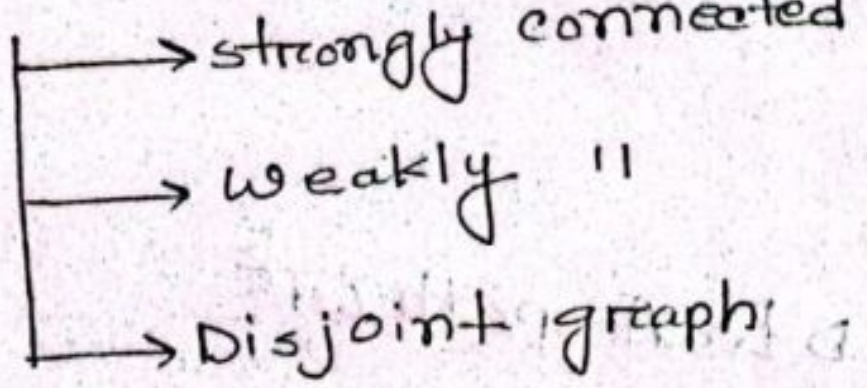
Path :-

Q 1) Difference between graph & Tree → cycle होय ना

2) cycle

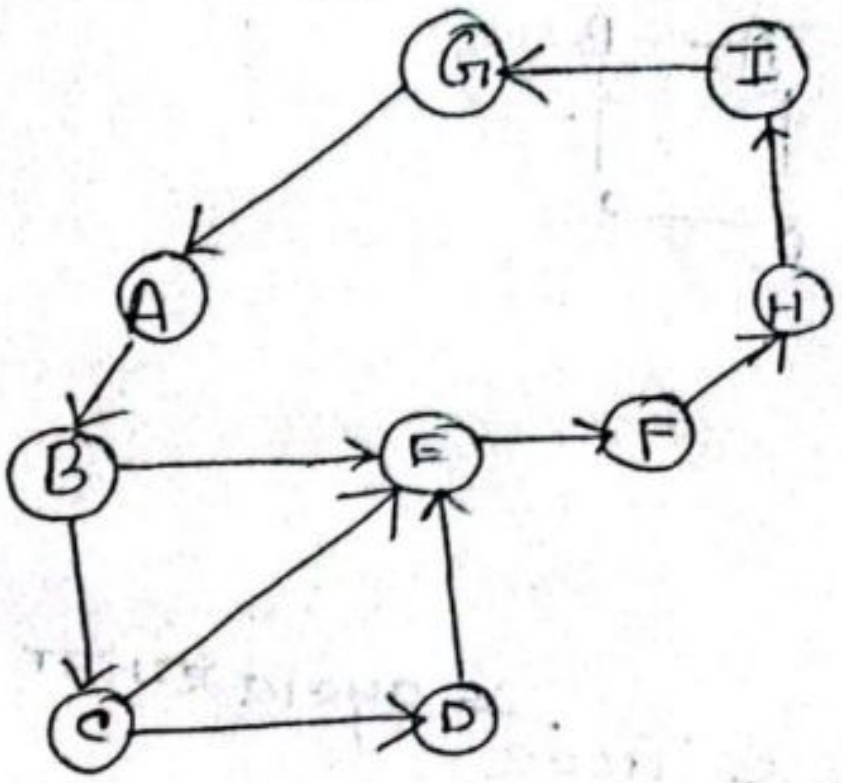
3) Loop/self edge

4) Directed graph

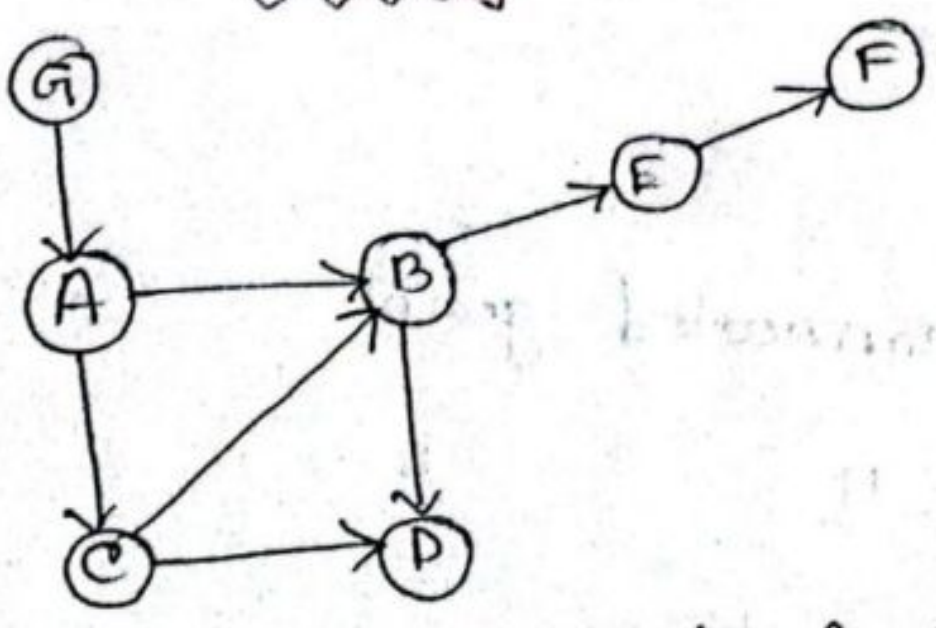


strongly connected graph:

* start to end path আছে end to start path must আকারে



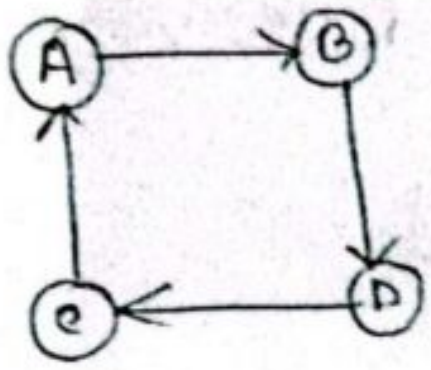
weakly connected graph

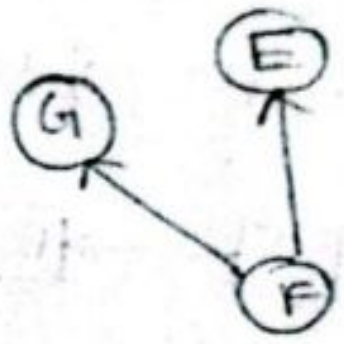


A to D path আছে; D to A path নেই

(A → C → D)

Disjoint Graph



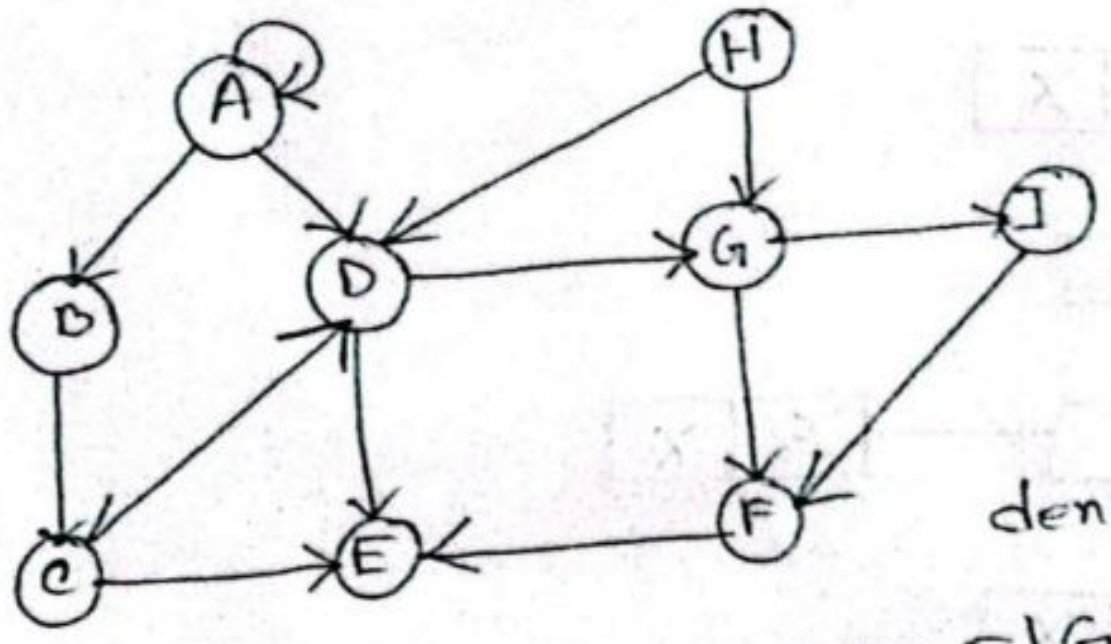


Degree (In degree, Out degree)

Arrow নিজের দিকে — in degree

Arrow অন্যদিকে যুক্ত থাকবে — out degree

* Show adjacency list and adjacency matrix representation for the graph given below;



→ node connected থাকলে 1

node connected না থাকলে 0

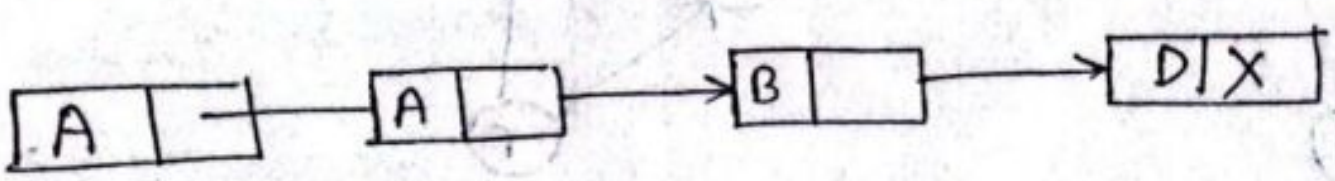
$O(V^2)$

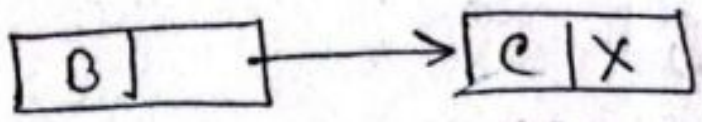
dense graph

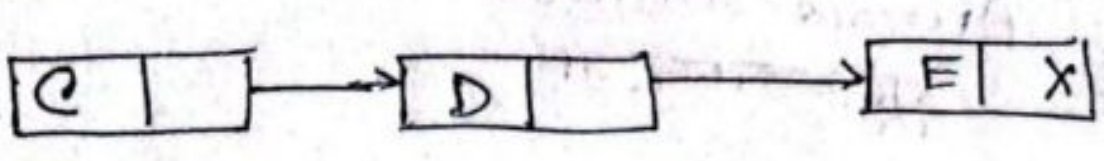
	A	B	C	D	E	F	G	H	I
A	1	1	0	1	0	0	0	0	0
B	0	0	1	0	0	0	0	0	0
C	0	0	0	1	1	0	0	0	0
D	0	0	0	0	1	0	0	0	0
E	0	0	0	0	0	1	0	0	0
F	0	0	0	0	0	0	1	0	0
G	0	0	0	0	0	0	0	1	0
H	0	0	0	0	0	1	0	0	0
I	0	0	0	0	0	0	0	0	0

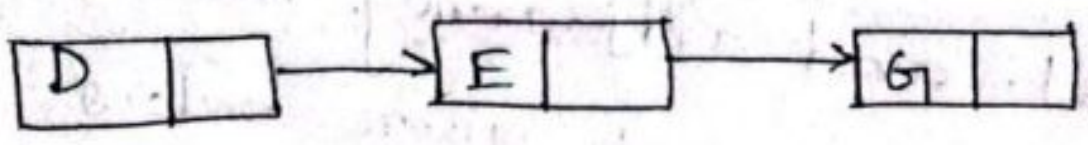
Adjacency list

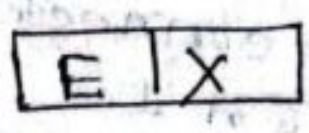
* Diff between adjacency list and matrix

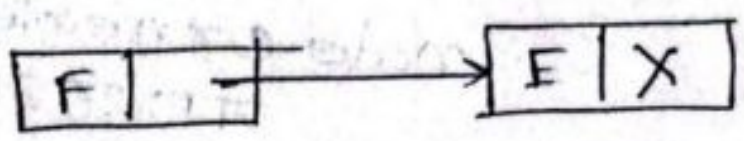


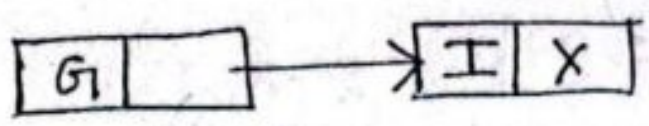


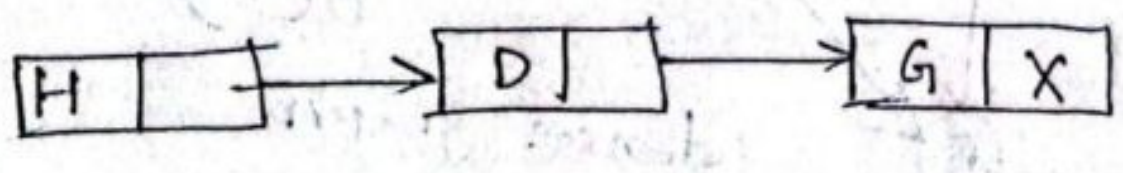


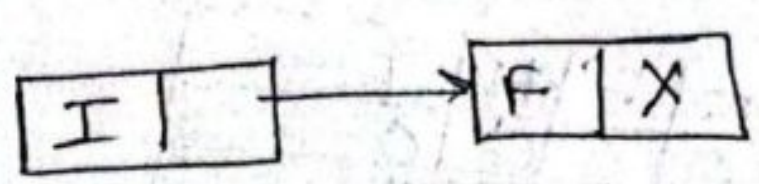












For Directed T.C = O(v+e)

For Undirected T.C = O(v+2e)

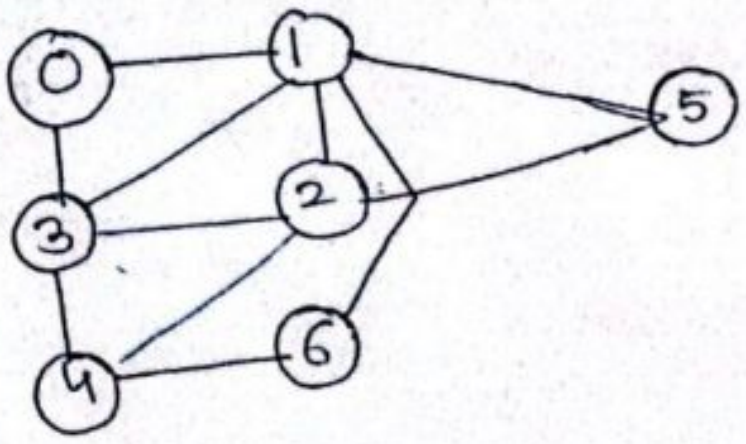
Queue (FIFO, enqueue, dequeue) } Diff
Stack (LIFO, push, pop)

Traverse Graph

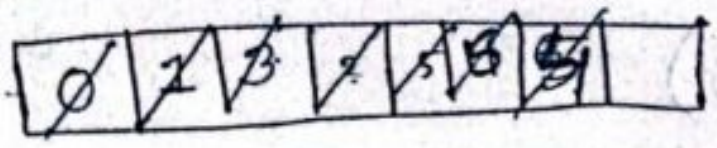
visited the explore vertices 2 ways 1) BFS &

ii) DFS

BFS



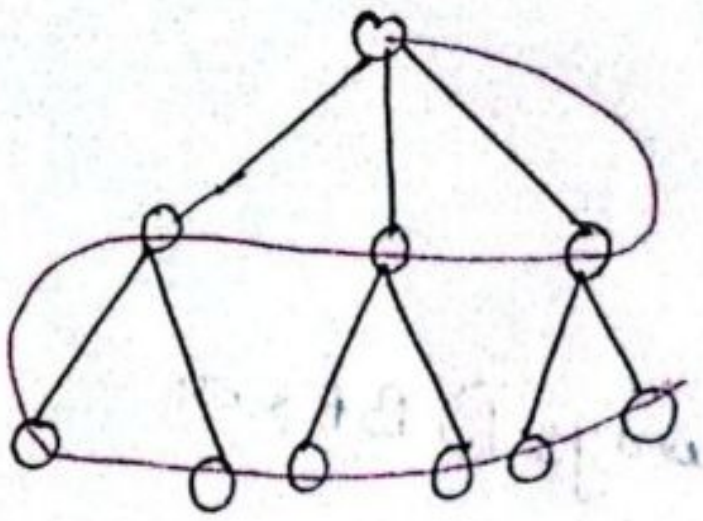
Queue

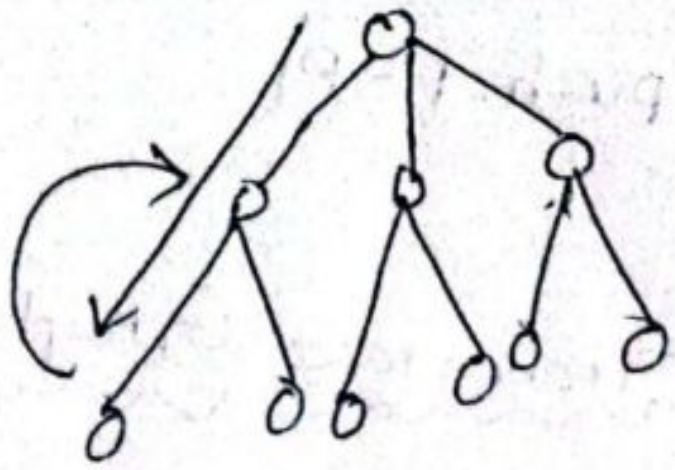


Result

0, 1, 3, 2, 5, 6, 4

BFS | DFS





Backtracking ~~ने~~ Backtracking आहे

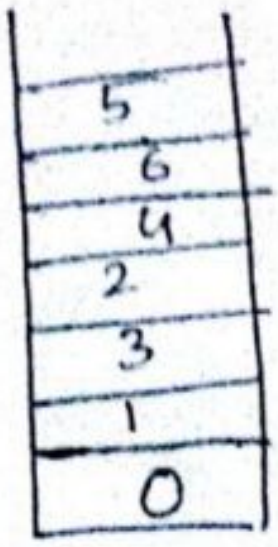
Queue | stack

Pseudocode BFS

```
BFS (G, s)
{
  q = queue ();
  q. enqueue (s);
 while (!q. empty ())
 {
  q. dequeue ();
  for all u ~~daj~~ adj v
 { if (u is not visited)
 { q. enqueue (a);
}}}
```


CamScanner

DFS



6, 4, 5, 2, 3, 1, 0

Result

0, 1, 3, 2, 4, 6, 5

pseudocode

```
visited [v] = true;
for each u adj to v
{
    if (visited [v] == false)
        {
        DFs (u);
        }
}
}
```