Unit 3

Greedy method

* Greedy methods

n inputs → Select for Optimal Solution → Feasible Solution yes ↓ includes

Depends upon local maximum → Greedy methods

Feasible Solution → yes ↓ includes

Greedy Algorithms are simple and straight forward. They take decisions on the basis of information in hand without worrying about the effect these decisions may have in the future. It never changes it's mind - once an instance is included in the solution it is there for good, once an instance is excluded from the solution, It is never reconsider, they are easy to invent, easy to implement and work efficiently.

Greedy Algorithms are Used to Solve Optimisation problem

examples include, Finding the shortest route from one node to another through a network or finding the Best order to set of George on computers.

In such a context, a It works by choosing the arc or job that seems most promising

at any instance, it never reconsider this decision whatever soln may arise later, there is no need to evaluate alternatives.

Coin example:-

We have the coin of 100 RS, 25 RS, 10 RS, 5 RS and 1 RS and we have to pay 289 Rupees by giving the minimum amount of coin

2	100rs coins	= 200
2	25rs coins	= 50
3	10rs coins	= 30
1	5rs coins	= 5
4	1rs coins	= 4
		289

logical answer

main answer:-

function {make-change} (n) set of coins
{make change from n units using the least possible number of coins. The costant C specifies the coinage}

const C = {100, 25, 10, 5, 1}

S ← ∅ {S is a set that will hold the soln}

S ← 0 {S is the sum of the items in S}

while S ≠ n do
n ← the largest item in C such that S+n≤n
if there is no such item then return "no soln found"

S ← S ∪ {a coin of value x}

S ← S + x
return S

* Genral Structure of Greedy Algorithm

This Group of Algorithm tries to solve the problem which have the following general structure, there are an input of some values and an object function. The method gives an optimal solution to the problem by considering the input one at a time, checking to see if it can be included in the set of values which give an optimal solution and then check if it is feasible solution all of n inputs may not be included only those needed to form the optimal solution will be included. each input may consume some resource which is generally available in limited quantity we may say that the feasible solution represent basic or essential requirements of the problem & the optimal solution denotes most specific and desirable requirements of the Problem

The flow of data is shown in figure, the strategy used for obtaining optimal solution is called Greedy method.

The word 'Greedy' refers to Allocating maximum possible values of some limited resource to the first elements which enters the optimal solution.

The feasibility of the soln is expressed in the terms of obeying the constraint of the resource.

The greedy method depends upon local (short range) maximum.

★ Greedy Algorithm Advantages

In the principle Advantages of this algorithm that they are usually state forword

Easy to understand and code

It is used for obtaining optimal solution

In this method the optimal solution is genrated without divising previously genrated soln.

Always taking the Best available choice is usually easy with analysing the run-time for greedy Algorithm will generally be much easier than for other technique (like divide & conquer)

Greedy Algorithms can often be implemented more efficiently than Other Algorithm

★ Greedy Algorithm Disadvantages

The principle disadvantage is that, for many problems, there is no greedy Algorithm

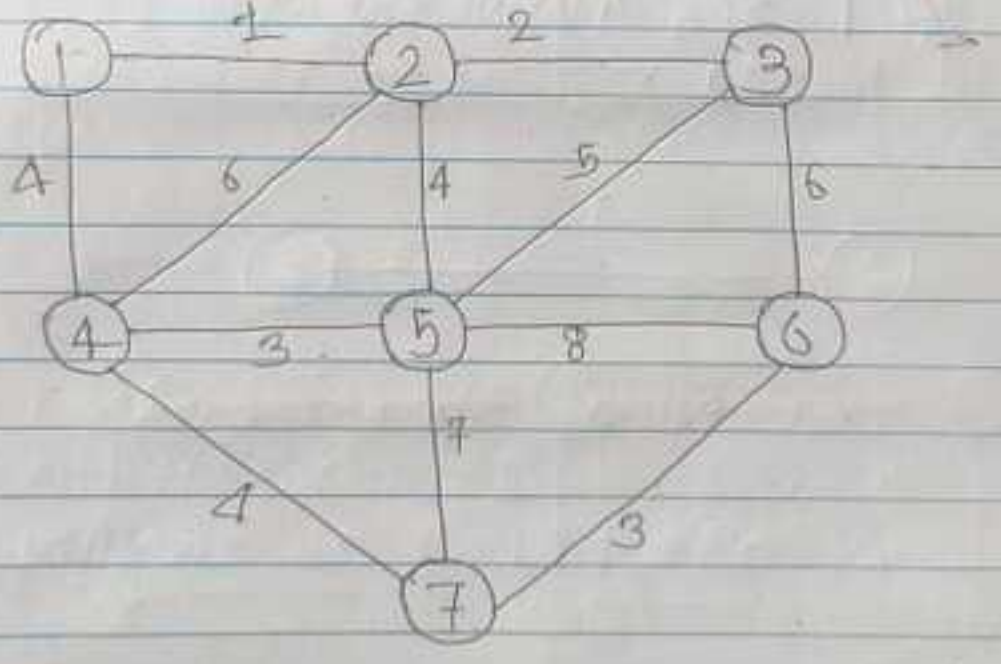
Despite their simplicity, correct greed Algorithm can be subtle.

once you have formula, write Greed Approach, designing Greedy Algorithms can be easy, however finding the Right approach can be harmed. It is a possible that most optimal Short term solution may lead worst possible long term outcome.

Greedy Algorithm do not always produce optimal solution

☆ Minimum Spanning Tree

Q. write a Kruskal Algorithm to generate minimum cost spanning tree: simulate Algorithm for the graph given below

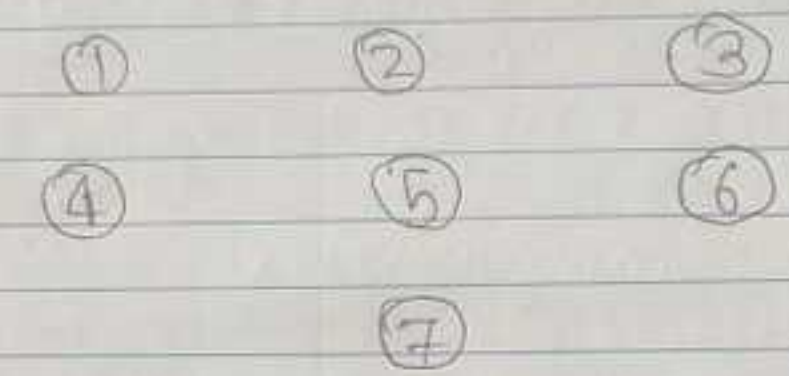


Consider the given graphs, Sort the ages in increasing order and select the age with minimum weight

Edge	Cost
[1,2]	1
[2,3]	2
[4,5]	3
[6,7]	3
[1,4]	4
[2,5]	4
[4,7]	4
[3,5]	5
[2,4]	6
[3,6]	6
[5,6]	8
[5,7]	7

First, we will select all the vertices / nodes. Care should be taken that for not forming a circuit / cycle and all the vertices should be visited

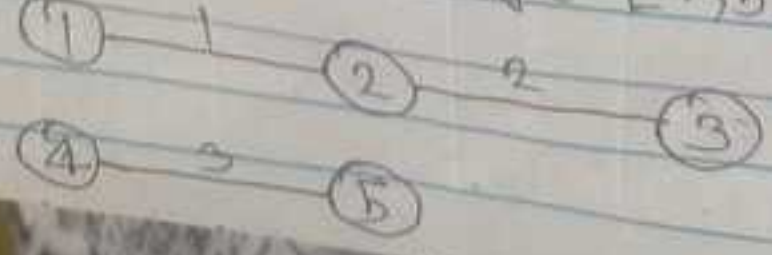
Now we will create the Nodes as follows



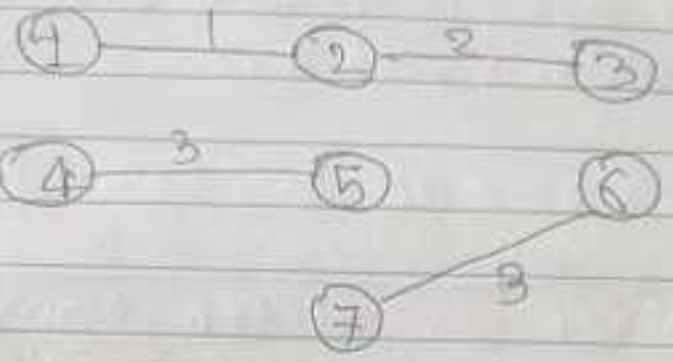
Step 1 :- Select Shortest age which is [1, 2] with a cost of 1, so add with to T and draw the age [1, 2] as shown below

Step 2 :- The Next Shortest age is [2, 3] with a cost of 2, so add it to T and draw the age [2, 3] as shown below

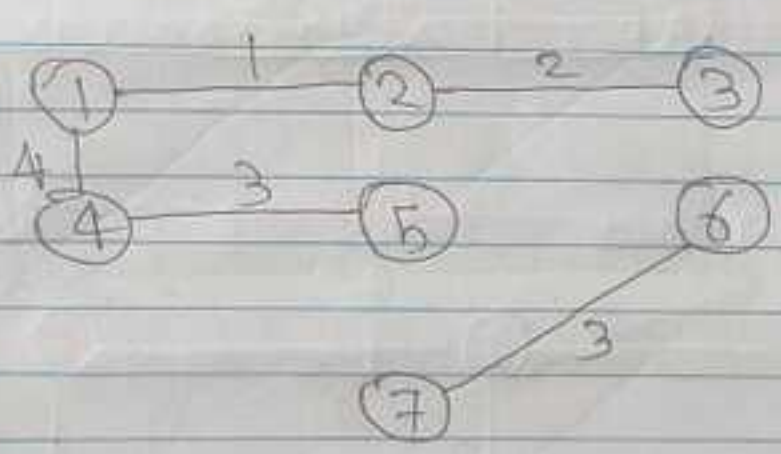
Step 3 :- The Next Shortest age is [4, 5] with a cost of 3, so add it with T and draw the age [4, 5] as shown below



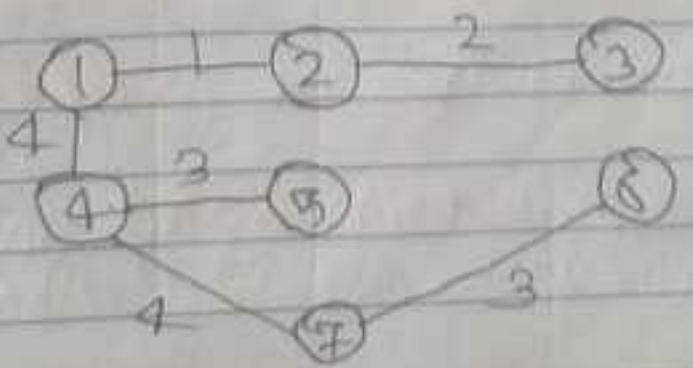
Step 4 :- The next shortest age is [6,7] with a cost of 3, so add it with T and draw the age [6,7] as below



Steps :- The next shortest age is [1,4], [4,7], [2,5] with a cost of 4 so we select [1,4] is on first come first serve basis, so add it to T and draw the [1,4] as shown below.



Step 6 :- The next shortest age is [4,7], [2,5] with a cost of 4, so we select [4,7] is on FCFS basis, so add it into T and draw the age [4,7] as shown below



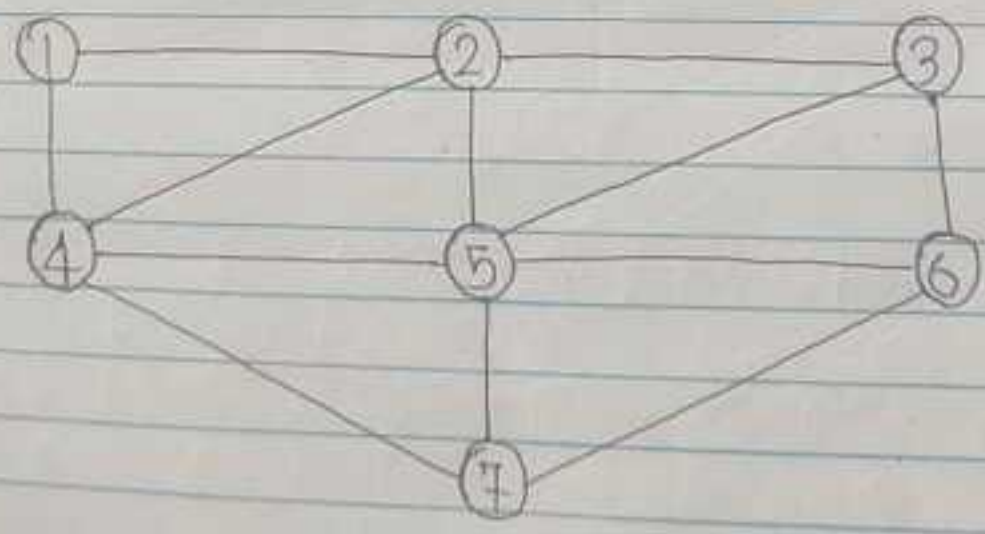
The above tree is the required minimum Spanning tree which is generated by Kruskal's Algorithm.

In this there are n nodes and n-1 edges & the total length is 17.

The solution proceed as follows :-

Step	Edge Consider	connected componend
Initialization	—	{1}, {2}, {3}, {4}, {5}, {6}, {7}
1	{1,2}	{1,2}, {3}, {4}, {5}, {6}, {7}
2	{2,3}	{1,2,3}, {4}, {5}, {6}, {7}
3	{4,5}	{1,2,3}, {4,5}, {6}, {7}
4	{6,7}	{1,2,3}, {4,5}, {6,7}
5	{1,4}	{1,2,3,4,5} {6,7}
6	{2,5}	Rejected
7	{4,7}	{1,2,3,4,5,6,7}

★ Prim's Algorithm



★ General Characteristics of greedy Algorithm

① We have some problem to solve in an optimal way, to construct the solution of

our Problem, we have set of values, The coin that are available, the ages of graph that may be use to build Path, The set of jobs to be scheduled, etc.

② As the Algorithm proceeds, we accumulate to other sets, one contain the values that have already been considered & choosen, while the other contains the values that have been considered & rejected.

③ There is a function that checks whether particular set of values provides a solution to our problem, ignoring Questions of optimality for the time being.

For instance to the coin we have choosen add upto the amount to be paid

④ ~~A second function checks whether~~ ~~to~~ the selected age provide a path to the node we wish to reach? have all the jobs been selected or schedule?

④ A second function checks whether a set of values is visible i.e whether or not it is possible to complete the set by adding further value, so as to obtain atleast one solution to our problem

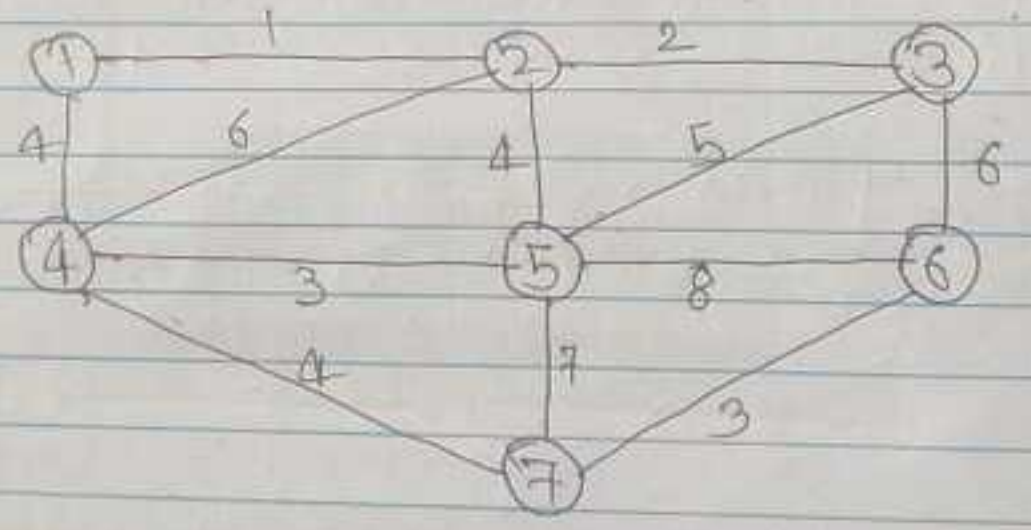
⑤ Another function the selection function indicate at any time of the remaining values that have neither been choosen nor rejected is the most promising.

(6) Finally an Objective function gives a value of a solution we have formed the no of coins we use to make change, the length of a path we constructed, the time needed to process all the jobs in the schedule or whatever other values we are trying to optimise.

The objective function Does not appear explicitly in the greedy algorithm.

* Prim's Algorithm

Q. Write a Prim's Algorithm to generate minimum spanning tree : simulate Algorithm for the graph given below



1 2 3

4 5 6

7

Now, we will consider all the vertices first then we will select an age with minimum

weight, Algorithm Proceeds by Selecting adjacent age with minimum weight.

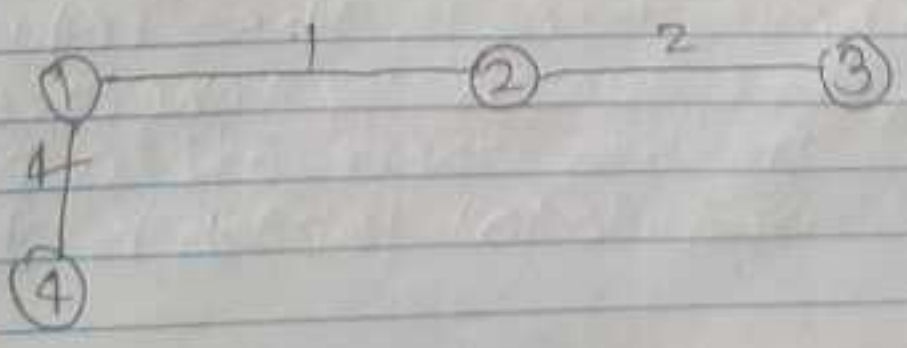
care should be taken that for not forming a circuit, all the vertices should be visited.

we arbitarily choose node 1 as a starting node

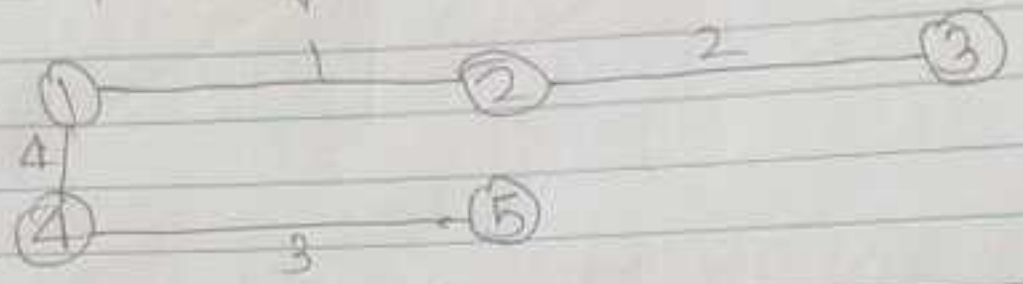
Step 1:- Select the Arbitary root, let the root be the node 1 as a starting point, select the shortest age from root is {1,2} with a cost of 1, so add it to T and draw age {1,2} as shown below

Step 2 :- The Next Shortest age is {2,3} with a cost of 2 and add it to T and draw the age {2,3} as shown below.

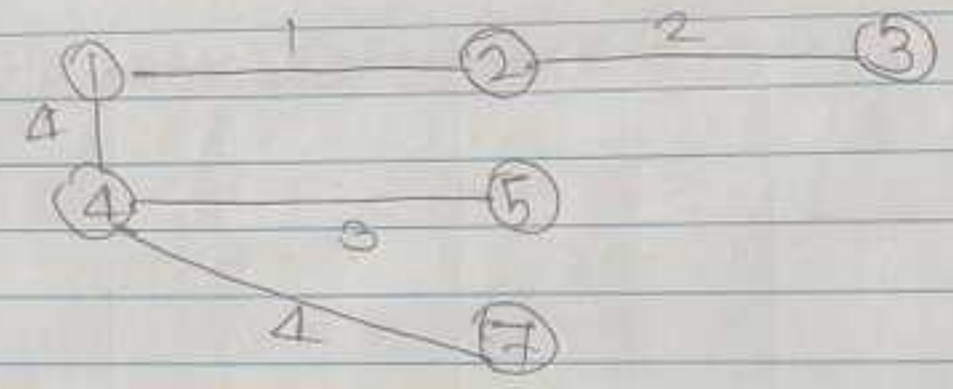
Step 3 :- The Next Shortest age is {1,4} with a cost of 4, add it to T and draw the age {1,4} as shown below



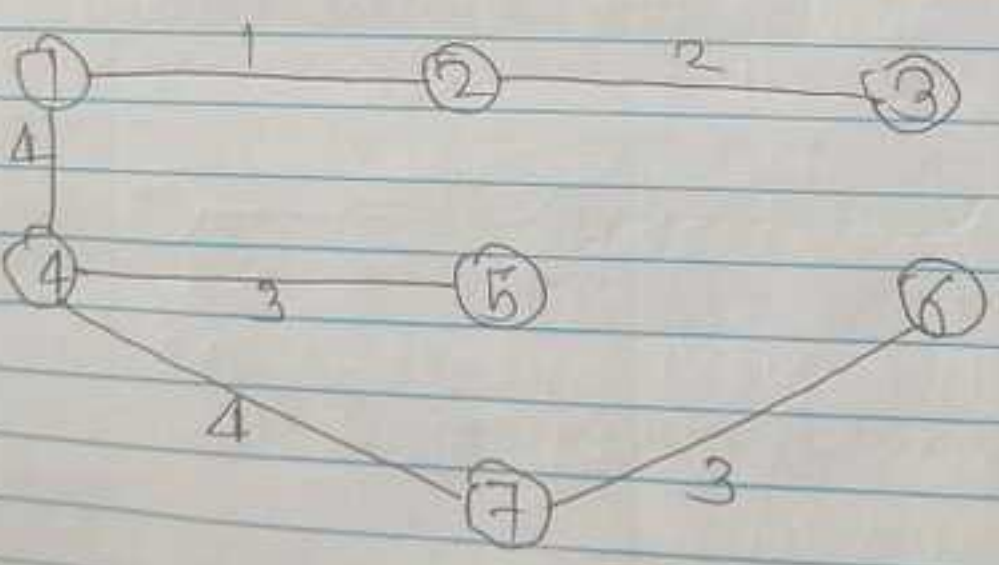
Step 4 :- The next shortest age is {4,5} with cost of 3, so add it to T and draw the age {4,5} as shown below



Step 5 :- The next shortest age is {4,7} with a cost of 4, so add it to T and draw the age {4,7} as shown below



Step 6 :- The Next shortest age is {7,6} with cost of 3, So add it to T and draw the age {7,6} as shown below



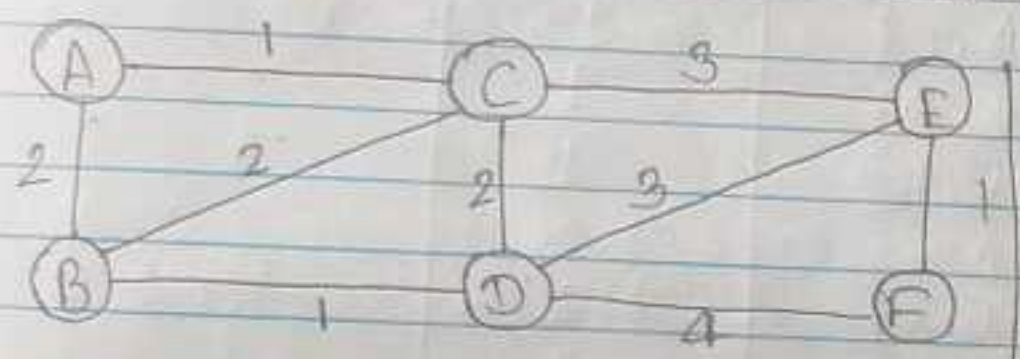
The above tree is the required minimum spanning tree which is generated by Prim's Algorithm, in this there are n nodes and n-1 edges and total length is 17.

The solution proceeds as follows

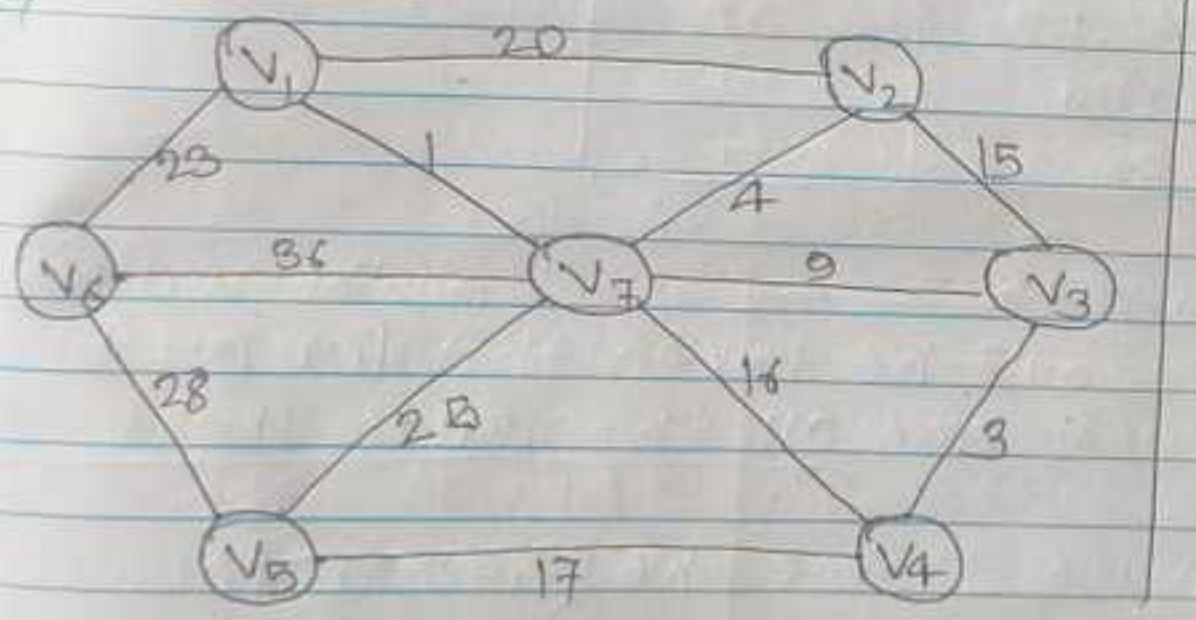
Step	Edge considered ie {u,v}	Connected components
Initialization	- - - - - - - - -	{1} {2} {3} {4} {5} {6} {7}
1	{1, 2}	{1, 2} {3} {4} {5} {6} {7}
2	{2, 3}	{1, 2, 3} {4} {5} {6} {7}
3	{1, 4}	{1, 2, 3, 4} {5} {6} {7}
4	{4, 5}	{1, 2, 3, 4, 5} {6} {7}
5	{4, 7}	{1, 2, 3, 4, 5, 7} {6}
6	{7, 6}	{1, 2, 3, 4, 5, 6, 7}

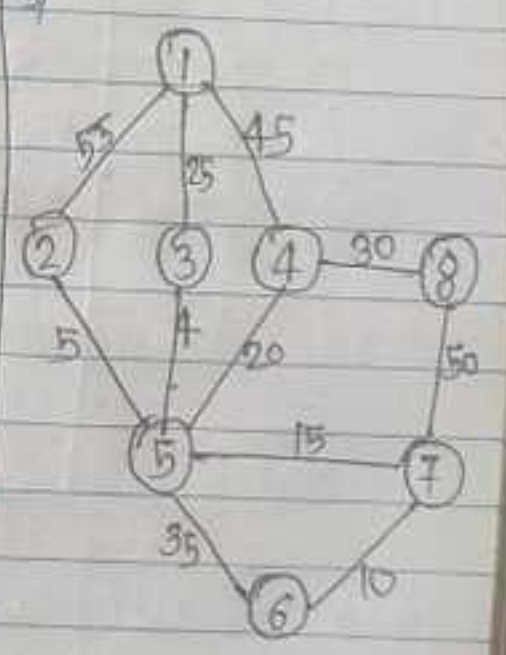
Q. Solve below two problems by Kruskal & Prim's Algorithm

1)



2)

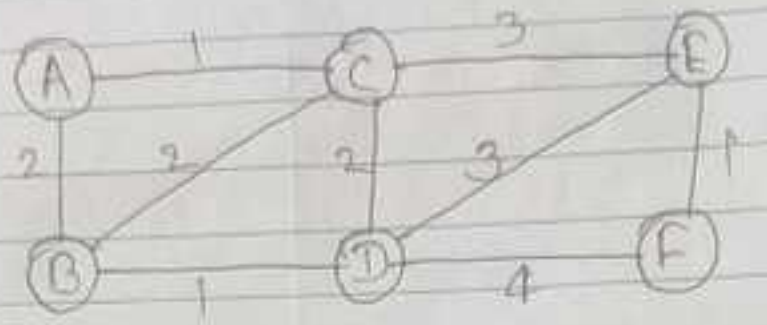




Solution :-

Now, we will solve these both by Kruskal and Prim's algorithm as follows

17



consider the given graph, sort the ages in increasing order and select the age with minimum weight

Edge	cost
{A, C}	1
{B, D}	1
{E, F}	1
{A, B}	2
{B, C}	2
{C, D}	2
{C, E}	3
{D, E}	3
{F, D}	4

first, we will ~~take care~~ select all the vertices) nodes. care should be taken that for not ~~Returning~~ a circuit/cycle and all the vertices should be visited

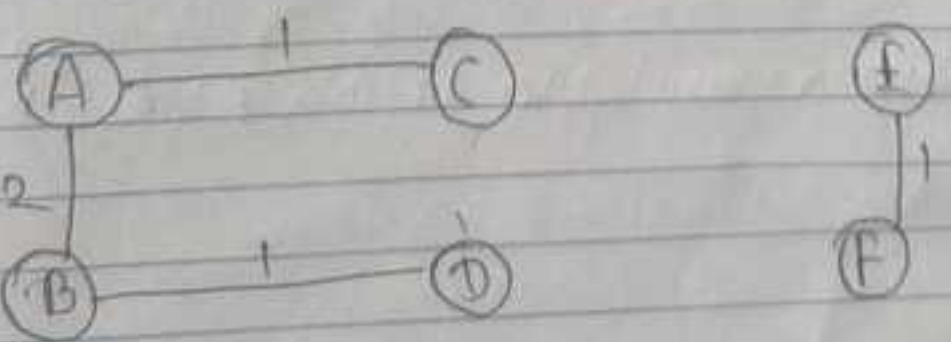
Now, first we frame the graph as follows

Step 1 :- Select the Shortest age which is {A, C} with cost of 1, so add it with T and draw the age {A, C} as follow

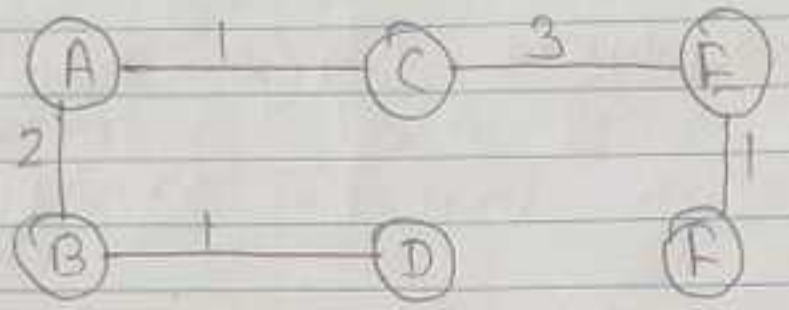
Step 2 :- Select the Next Shortest age is {B, D} with a cost of 1, So add it with T and draw the age {B, D} as follow

Step 3 :- The Next Shortest age is {E, F} with a cost of 1, So add it with T and draw the age {E, F} as follow

Step 4 :- The Next Shortest age is {A, B} and {B, C}, ~~with a~~ and {C, D} with a cost of 2 So, we select {A, B} is on First come First serve Basis, so add it to T and draw the age {A, B} as follow



Step 5:- We cannot add ages {B,C} and {C,D} because it can form circuit / cycle. hence the Next Shortest Node is the {C,E} and {D,E}, So we select {C,E} on the FCFS basis, add it to T and draw the age {C,E} as follows



The above tree is the required minimum spanning tree which is genrated by kruskals algorithm

In this there are n nodes and n-1 ages are present & the total length is 8

The solution proceed as follows

Step	Edge consider	connected component
Initialization	-	{A}, {B}, {C}, {D} {E}, {F}
1	{A, E}	{A, C}, {B}, {D}, {E}, {F}
2	{B, D}	{A, C}, {B, D}, {E}, {F}
3	{E, F}	{A, C}, {B, D}, {E, F}
4	{A, B}	{A, C}, {B, D}, {E, F}
5	{B, E}	{A, B, C, D}, {E, F}
6	{C, D}	Rejected
7	{E, F}	Rejected
		{A, B, C, D, E, F}

Similarly, Solve according to Prim's Algorithm.

Minimum Spanning Tree

The Minimum Spanning Tree Problem is to be find ~~the~~ a subset T of the edges of G, such that all nodes remain connected when only the edges in T are used, and the sum of all the length of the edges in T is as small as possible.

Each Edge has assign a non-negative length. as G is connected, atleast one solution exist If G has edges of length 0 then there may exist several solutions whose total length is the same but if it involves different numbers of edges.

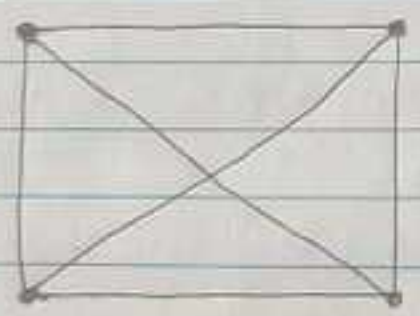
For example, given two solutions with equal total length, we prefer the one with least edges. Here, the problem may have several different solutions of equal values, instead of talking about length, we can associate a cost to each edge. The problem is then to find a subset of the edges whose total cost is as small as possible.

Let, $G' = (N = A)$ be the Partial graph formed by the nodes of G and the edges in T, and suppose there are n nodes in N. A connected graphs with n nodes must have atleast n-1 edges, so this is the minimum no. of edges there can be ~~a~~ in T.

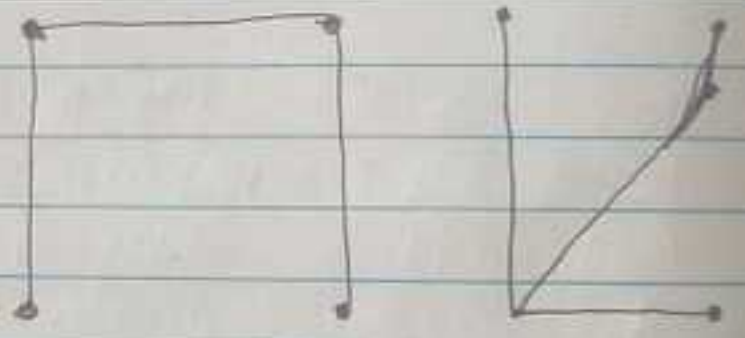
If Graph G' is connected and T has more than n-1 edges and contains atleast one cycle, hence if G' we can remove atleast one of these without this connecting G', Provided we choose an edge that is a Part of a cycle

T must have exactly n-1 edges, and since G' is connected it must be a Tree. The Graph G' is called minimum Spanning Tree for the Graph G

Following figure shows the example of minimum Spanning Tree



Undirected Graph



Some of it's spanning tree

★ Types of minimum Spanning Tree

A greedy method to obtain the minimum spanning tree would construct the tree edge by edge where each next edge is choosen according some optimization criteria.

There are two ways, in which this Criteria can be acheived

1) Kruskal's Algorithm
2) Prim's Algorithm

1) Kruskal's Algorithm

Edges are consider in nondecreasing order of weight. The set T of edges at each stage is such that it is possible to complete T into a tree, thus T may not be a tree at all stages of the algorithm this also result in a minimum cost tree this algorithm is called the Kruskal's Algorithm.

2) Prim's Algorithm

The set of edges is selected so far always forms a tree, the next edge to be added in such that not only it adds a minimum weight but also forms a tree with the previous edge, it can be be shown that this algorithm results in a minimum cost tree, this is called the Prim's Algorithm.

★ Job Sequencing with deadline

Given a set of n jobs each having a deadline (an integer) d[i] and Profit P[i] associated with it for a job i, ther Profit is earned if and only if the job is completed within it's deadline. Each job takes one unit of time. on a machine (Processor) and only one machine is available. we want to maximize the Profit. The jobs are arranged in decreasing order of Profit in an array.

A Physical solution to the Problem is a subset J of the ~~smaller~~ n jobs, such that each of them can be completed within it's deadlin

The value of feasible soln is $\sum P[i]$ for an $i \in J$. and optimal solution is feasible solution with maximum Profit for maximum value.

Consider that, there are n jobs that are to be executed at any time $t = 1, 2, 3, \ldots$

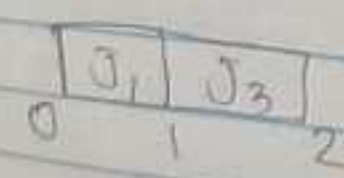
Only exactly one job is to be executed, the Profit P are given. These Profit are gained by corresponding jobs for obtaining feasible solution we should take care that the jobs get completed within a given their deadlines

we will follow following rules to obtain the feasible soln.

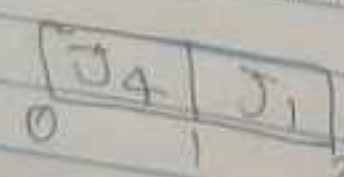
1) each job take 1 unit of time
2) If job starts before or at it's deadline, profit is obtained, otherwise no Profit
3) Goal is to schedule jobs to maximize the total profit.
4) Consider all possible schedules and compute the minimum total time in the system.

	J_1	J_2	J_3	J_4
Profit	50	15	10	25
Deadline	2	1	2	1

Uni Processor
No Preemption
1 Unit of time



50 + 10 = 60



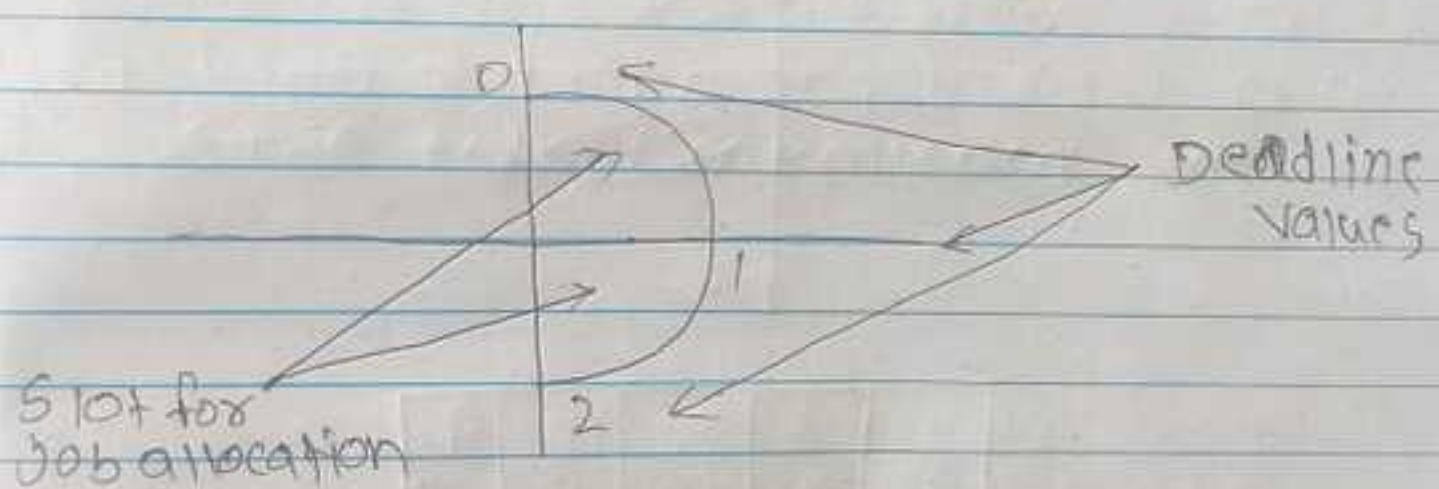
50 + 25 = 75

Optimal solution

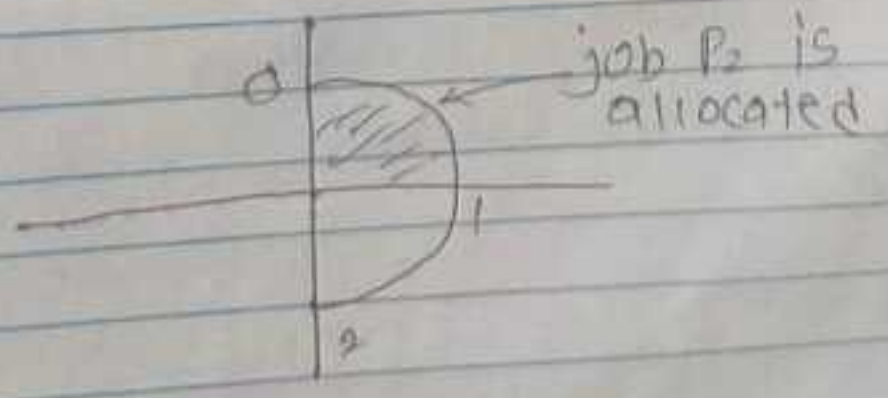
Q) Let, n=4 Profit vector $P = \{50, 35, 20, 25\}$ deadline vector $D = \{2, 1, 2, 1\}$ assume execution time of each job is 1 time unit. Sort the jobs according to their profit in descending order as follows

i	1	2	3	4
P_i	35	30	25	20
d_i	1	2	1	2

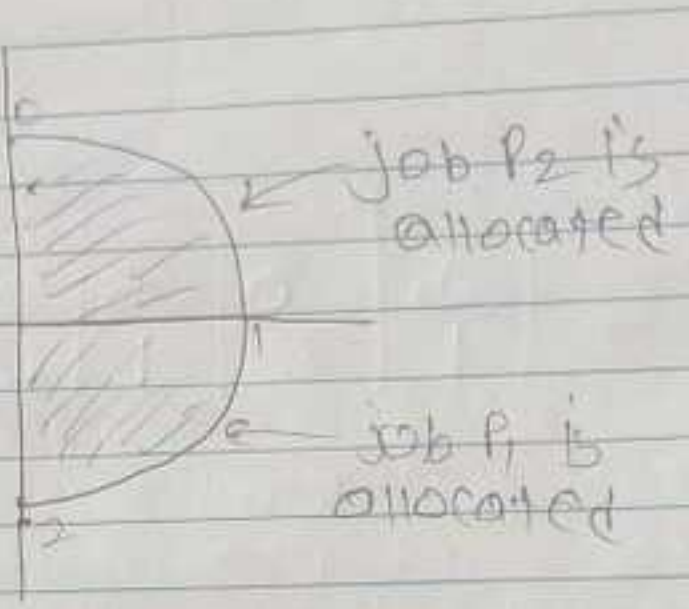
NOW, the maximum deadline is 2 so we consider maximum deadline as clockwise structure. We have plotted {0-2} structure in order to allocate the jobs in the specific slot.



The deadline specifies the maximum time to complete the specific job. Now allocate the job based on deadline starting with P_2 the deadline specified is 1, we will allocate the 0-1 slot to P_2 as the shown below



Next job we have is P_1, the deadline specified is 2, we will allocate 1-2 slot to P_1, as shown below



we have occupied all the given slot. We have executed P_2 and P_1 with Profit 35 and 30. The jobs must be done in the order shown first 2 then 1, In order to obey the deadlines Thus the maximum profit 35+30=65.

Q.27

i	1	2	3	4
P_i	100	10	15	27
d_i	2	1	2	1

Solve this Similar to Q1

Q3) $n=4$ $(P_1, P_2, P_3, P_4) = (50, 10, 15, 30)$

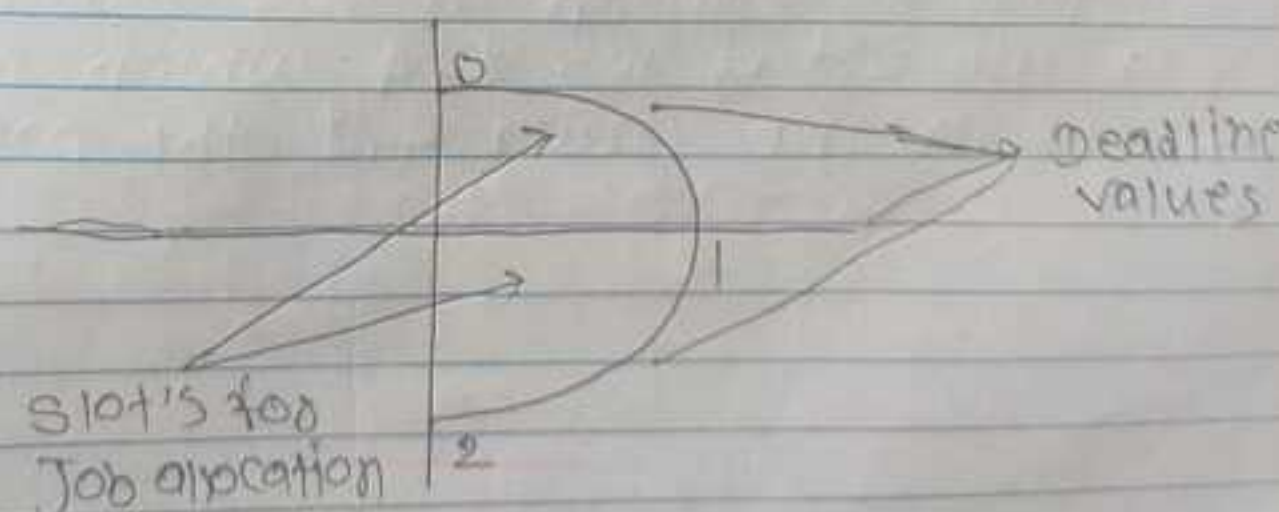
$(d_1, d_2, d_3$

with $n=4$ and the value $(P_1, P_2, P_3, P_4) = 50, 10, 15, 30$ and $(d_1, d_2, d_3, d_4) = (2, 1, 2, 1)$ Find the Optimal solution.

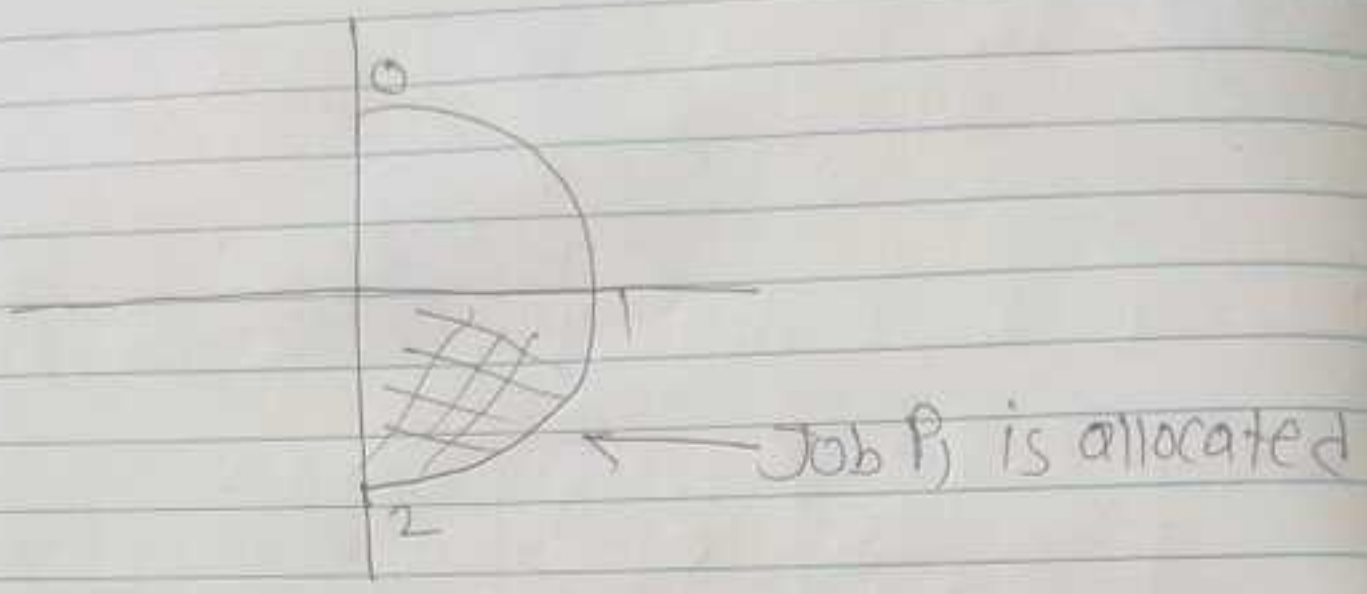
Now, sort the job according to their Profit descending order as

i	1	4	3	2
P_i	50	30	15	20
d_i	2	1	2	1

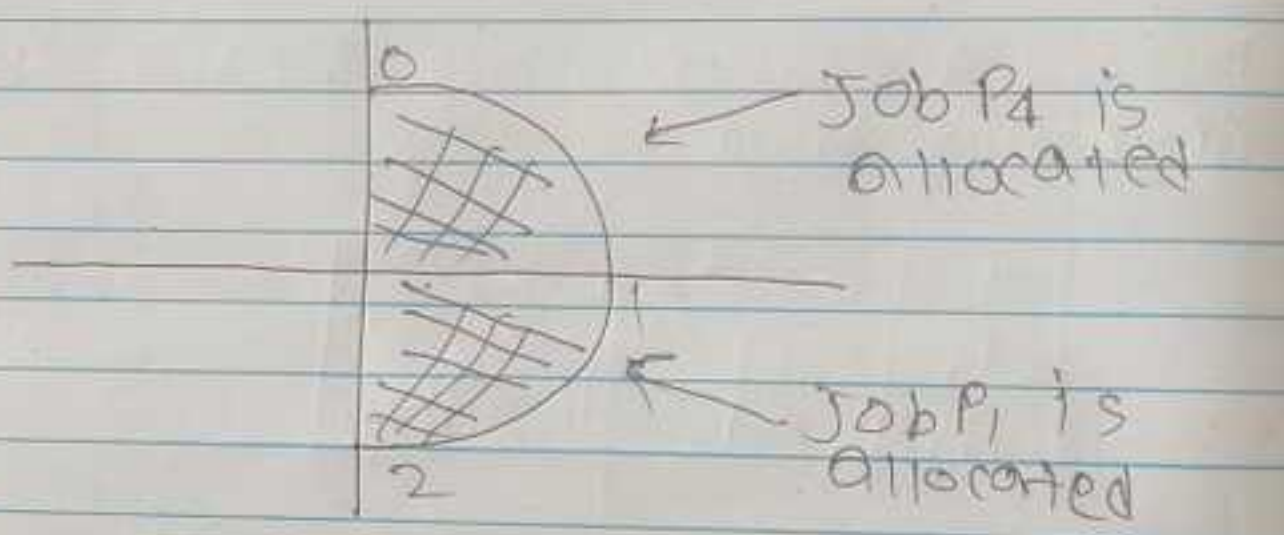
The maximum deadline is 2 so, we consider the maximum deadline as clockwise structure as follows.



Now allocate based on deadlines, Starting with P_1, the deadline specified is 2, we will allocate 1-2 slot to P_1 as follows



Next, we have is P4, the deadline specified is 1, we will allocate 0-1 slot to P4 as shown below



we have occupied all the given slots. We have execute P1 and P4 with Profit 50 and 30

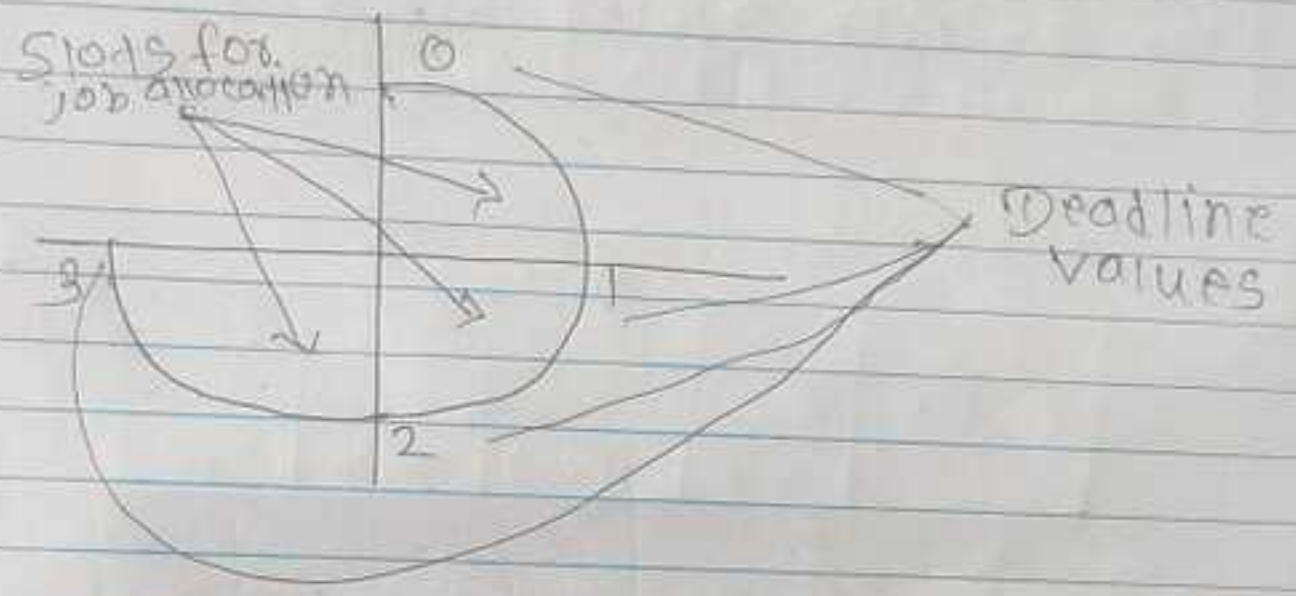
Our optimal solution in this case is, therefore to execute set of job 1, 4 which can only be done in the order 4, 1. the maximum profit is 50 + 30 = 80

Q.3)

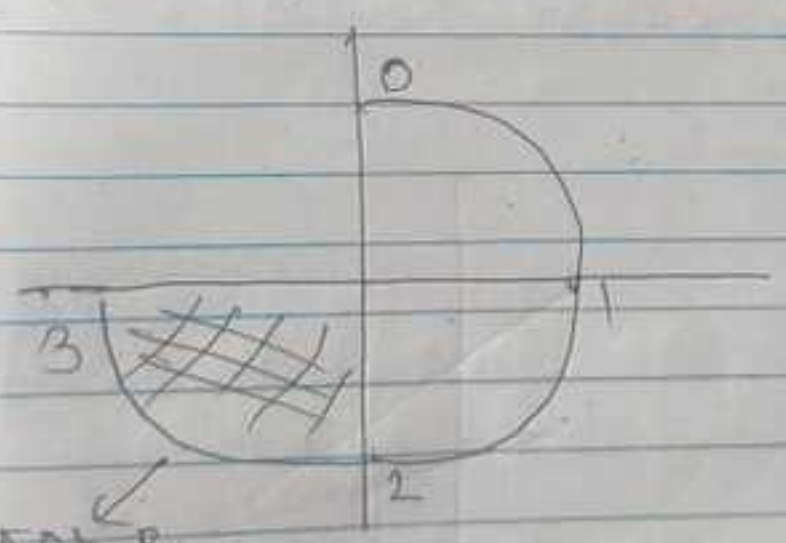
i	1	2	3	4	5	6
P_i	20	15	10	7	5	3
d_i	3	1	1	3	1	3

→

The maximum deadline is 3, so we consider the maximum deadline as clockwise structure as follows



Now, allocate based on deadline, starting with P_1, the deadline specified is 3, we will allocate 2-3 slot to P_1 as follows



ESC F1 F2 F5 F6 F7

✡ KN

M
of n
vi a
in
limi

Our
bag
of
the

In
can
mo
of
xi
xi
ar

②. E
a
w
A

Soln:-

fun

1)

2

bag pack
* KNAPSACK Problem

It is defined as- a theif considers taking M pounds of loot. The loot is in the form of n atoms, each with weight Wi and value Vi any amount of an atom can be put in the knapsack as long as the weight limit M is not exceeded.

Our Aim is to fill the KNAPSACK that is bag in a way that maximises the value of the included object while respecting the capacity constraint.

In this Problem, we assume that the object can be broken into smaller pieces, so we may decide to carry only a fraction x_i of object i where ~~xi is gre~~ $x_i \geq 0$ and $x_i \leq 1$, in this case objects i contributes $x_i w_i$ to the total weight in the knapsack and $P_i x_i$ to the value of the profit.

Q. Explain the knapsack Algorithm to find an optimal solution to the instance n=5 where, capacity M = 100 $W = \{10, 20, 30, 40, 50\}$, $P = \{20, 30, 66, 40, 60\}$

Soln:-

There are at least four possible selection function at each stage for this Problem.

1) The first solution is a random solution (optional)

2) ~~the~~ Second, we might choose the most valuable remaining object, arguing that

this increases the value of the load as quickly as possible

3> we might choose lightest remaining object, on the ground that this uses up capacity as slow as possible.

4> we might avoid this extremes by choosing the object whose value per unit weight is as high as possible.

1. Random solution :-

	1	2	3	4	5
Weight	10	20	30	40	50
Profit	20	30	66	40	60
Pi/Wi	2.0	1.5	2.2	1.0	1.2

If we select the random object to fill a knapsack in any order then we choose first object B then object 4 and finally we fill knapsack with object 3. The value of the solution obtained in this way is 80+40+20 = 120 (weight = 50+40+10 = 100)

2. Max Profit :-

If we select the object in order of decreasing value then we choose first object 3, then object 5. Now the next object is 4 but we want only 20 so we choose fractional part of object 4 that is we select only 0.5 part of object 4 as

$$= \frac{\text{weight required}}{\text{actual weight}} = \frac{20}{40} = 0.5$$

The Profit value of the fraction part is
= actual Profit of Object x weight required
actual weight

$= \frac{40}{40} \times 20 = 20$

Thus Profit of the solution obtain in this way is 66 + 60 + 20 = 126 weight = (30 + 50 + 20 = 100)

3> Minimum weight :-

If we select the object in order of increasing weight then we choose object 1, 2, 3 and 4 in the order and now check the knapsack is full profit value of this solution is 20 + 30 + 66 + 40 = 156 (10 + 20 + 30 + 40 = 100 weight =

4> Max Pi/Wi

If we select the objects in order of decreasing Pi/Wi Ratio, we choose first object 3 then object 1, object 2 now the next object is 5 it's weight is 50 but we want only 40, so we choose the fractional part of object 5 i.e we select only 0.8 part of object 5.

fractional part = weight required / actual weight = 40/50 = 0.8

Thus Profit value of a fractional part of object 5 is actual profit of a object upon actual weight multiplied by weight require
= 60/40 =

Profit value or Partial = actual Profit × weight required / actual weight = 60/30 × 40 = 8.48

Thus the maximum value is 20+30+66+...
= 164

Soln	x_1	x_2	x_3	x_4	x_5	value
Random	1	0	0	1	1	120
max Profit	0	0	1	0.5	1	146
min weight	1	1	1	1	0	156
max P_i/w_i	1	1	1	0	0.8	164 (optimal solution)

Q. Explain the Knapsack algorithm to find an optimal solution to the instance n = 7 with m=15, capacity (M) = $(w_1, w_2, \dots w_7) = (2,3,5,7,1,4,1)$
$(P_1, P_2, P_3, \dots P_7) = (10,5,15,7,6,18,3)$

i) max profit
ii) min weight
iii) max P_i/w_i

Soln:-

i) max profit

	1	2	3	4	5	6	7
weight	2	3	5	7	1	4	1
profit	10	5	15	7	6	18	3
P_i/w_i	~~0.2~~	~~0.6~~	~~0.33~~	~~1~~	~~2.16~~	~~0.22~~	~~0.3~~

i)
max profit

If we select the object of decreasing value the select the object 6, 3, 7 and

the next object is 4 and it's weight is 7 we want only 4 so we choose fractional part of object 4 i.e we select only 0.57 part of object 4 as follows

$$\text{fractional part} = \frac{4}{7} = 0.57$$

$$\text{Profit value of fractional part of obj 4} = \frac{\text{actual profit}}{\text{actual weight require}} \times \text{weight} = \frac{7}{4} \times 4 = 4$$

Thus Profit value of soln is $18 + 15 + 10 + 4 = 47$
weight = $4 + 5 + 2 + 4 = 15$

ii) min ~~Profit~~ weight

If we select the Object in order of Increasing weight then we choose the object 5, 7, 1, 2, 6 and the next object is 3 and it's weight is 5 but we want only 4 so we choose fractional part of object 3 that is we select only 0.8 part of Object 3 as follows

$$\text{fractional part} = \frac{4}{5} * = 0.8$$

$$\text{profit value} = \frac{\text{actual profit}}{\text{actual weight require}} \times \text{weight} = \frac{15}{5} \times 4 = 12$$

Total

Q. Explain Greedy Algorithm and solve Knapsack Prob
weight = (3,4,6,5), Profit = (2,3,1,4) and m = 8

iii) max p_i/w_i

If

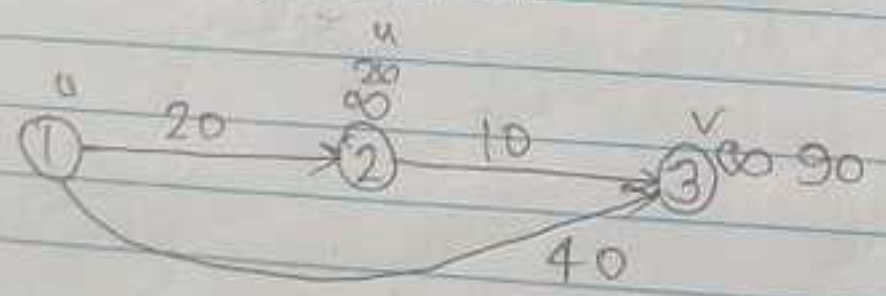
Q.

Sno

Q1 E
Path

S

★ Dijikstra's Algorithm
(Shortest Job first)



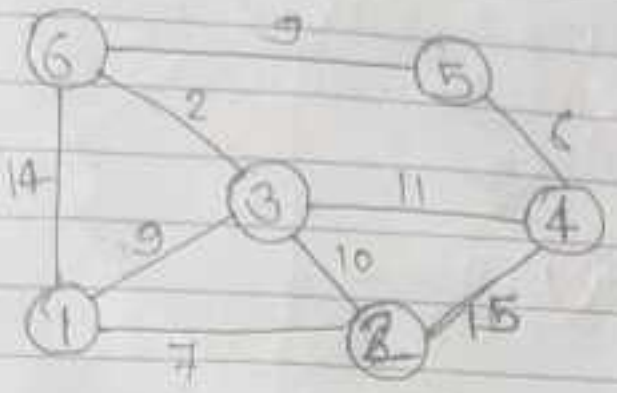
Relaxation

if $d(u) + c(u,v) < d(v)$
$d(v) = d(u) + c(u,v)$
$d(v) + c(u,v)$
$0 + 20 < \infty$
$0 + 40 < \infty$

$d(u) + c(u,v)$
$20 + 10 < d(v)$
$30 < 40$

Th
fro
V_4
Sel
Th
30

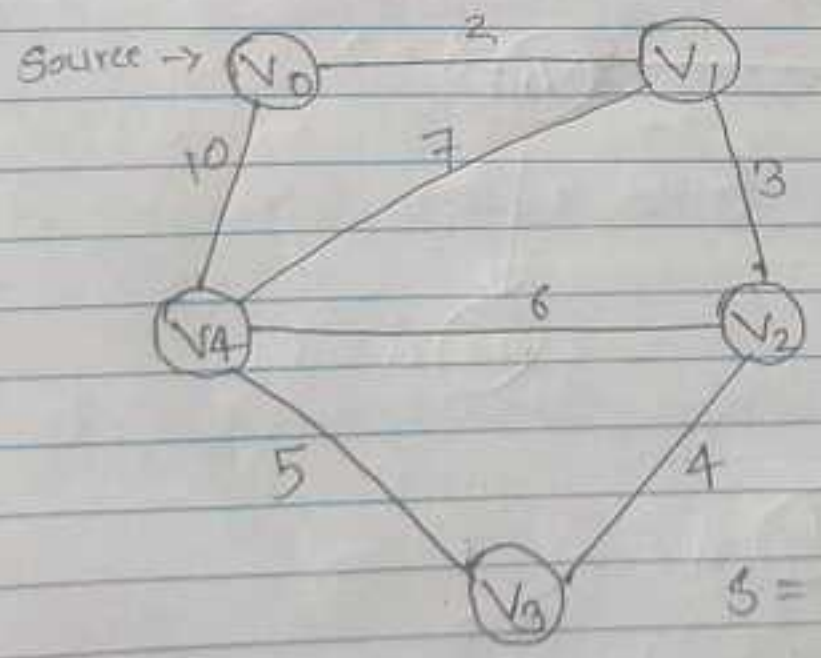
Q.



Shortest method

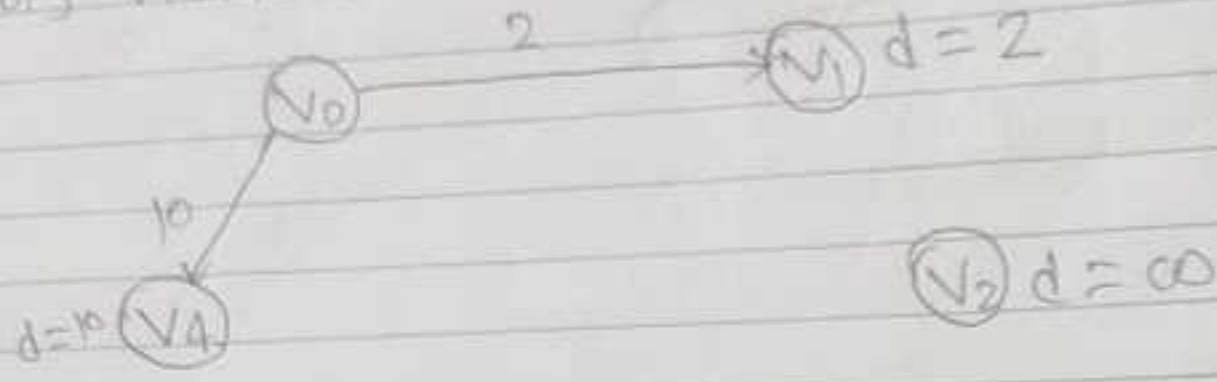
Source	2	3	4	5	6
	∞	∞	∞	∞	∞
1, 2	(1)	9	∞	∞	14
1, 2, 3	(7)	(9)	22	∞	14
1, 2, 3, 6	(7)	(9)	20	∞	11
1, 2, 3, 6, 4	(7)	(9)	(20)	20	11
1, 2, 3, 6, 4, 5				(20)	

Q] Explain Dijistra's Algorithm and find the Shortest Path from single source to other nodes of a graph



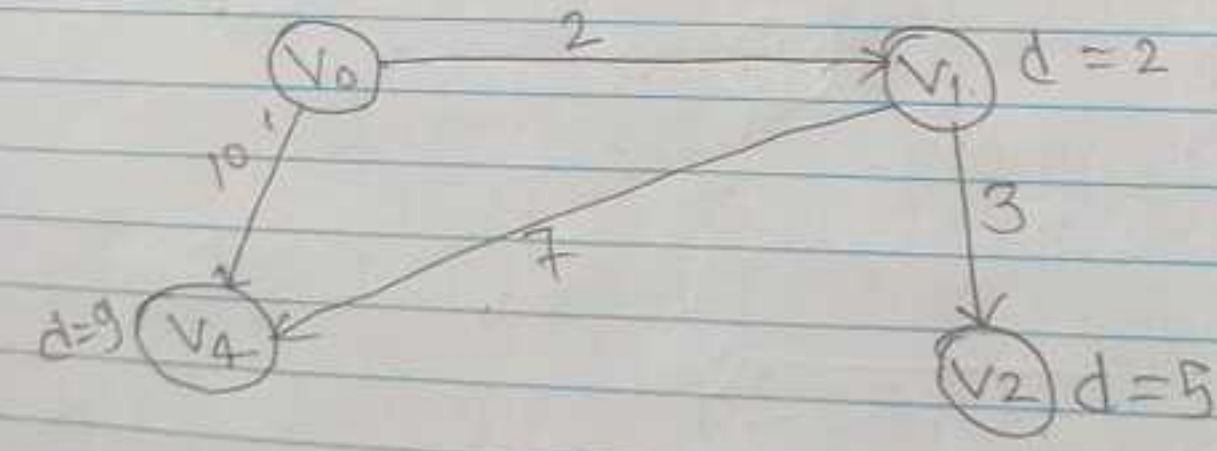
The starting node is V_0, there are two ages from V_0 ie V_0 to V_1 with a cost of 2 and V_0 to V_4 with a cost of 10. The minimum cost is selected from available ages which is V_0 to V_1. The variable d represents the total cost from source node V_0. The remaining of value of d are shown as

because, from Source node other nodes are still be visited, the node V_1 is now marked as visited as shown below



$$S = \{V_0\}$$

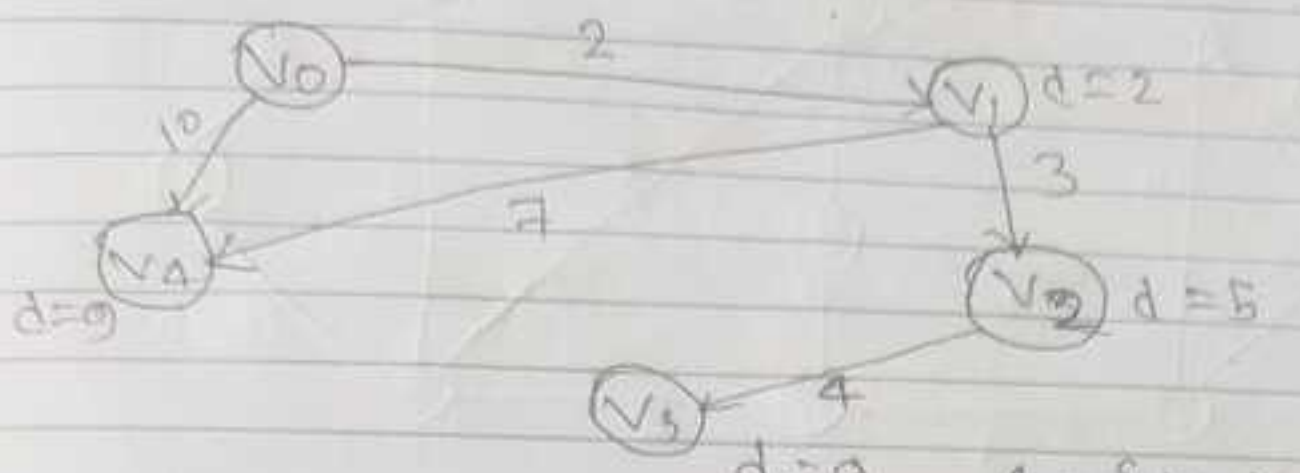
Now, from the node V_1, there are two nodes that can be reach that is V_4 and V_2 with cost of 7 and 3. so from the Source node V_0 we can reach V_4 with cost of 9 instead of direct age that is V_0 to V_4 with a cost of 10 and V_2 with a cost of 5 as shown below



V_3 d = ∞

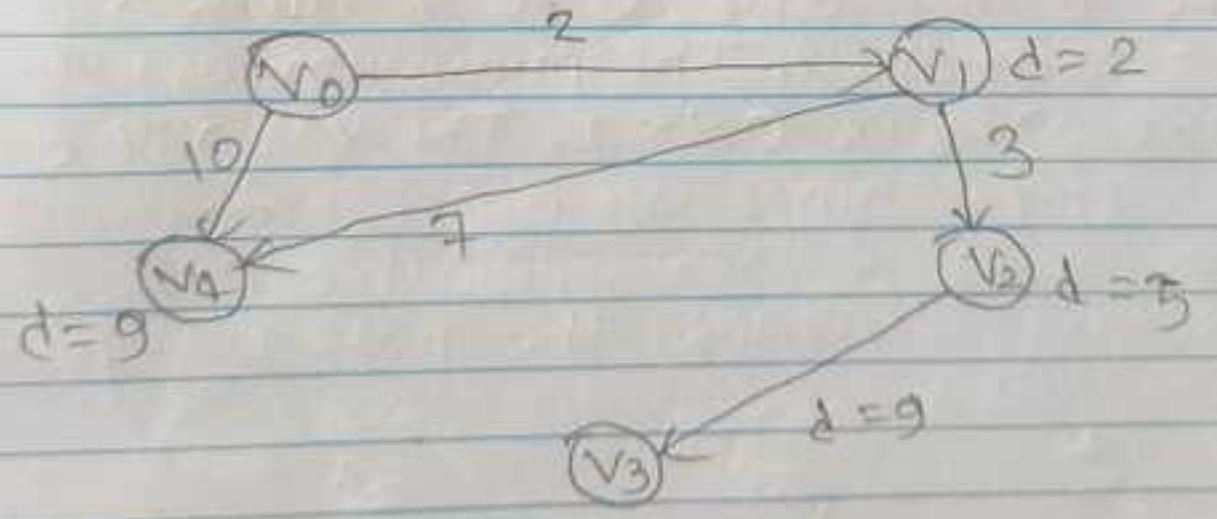
$$S = \{V_0, V_1\}$$

The minimum cost is selected from the available ages which is V_1 to V_2. The node V_2 is now marked as visited as shown below



$S = \{ V_0, V_1, V_2 \}$

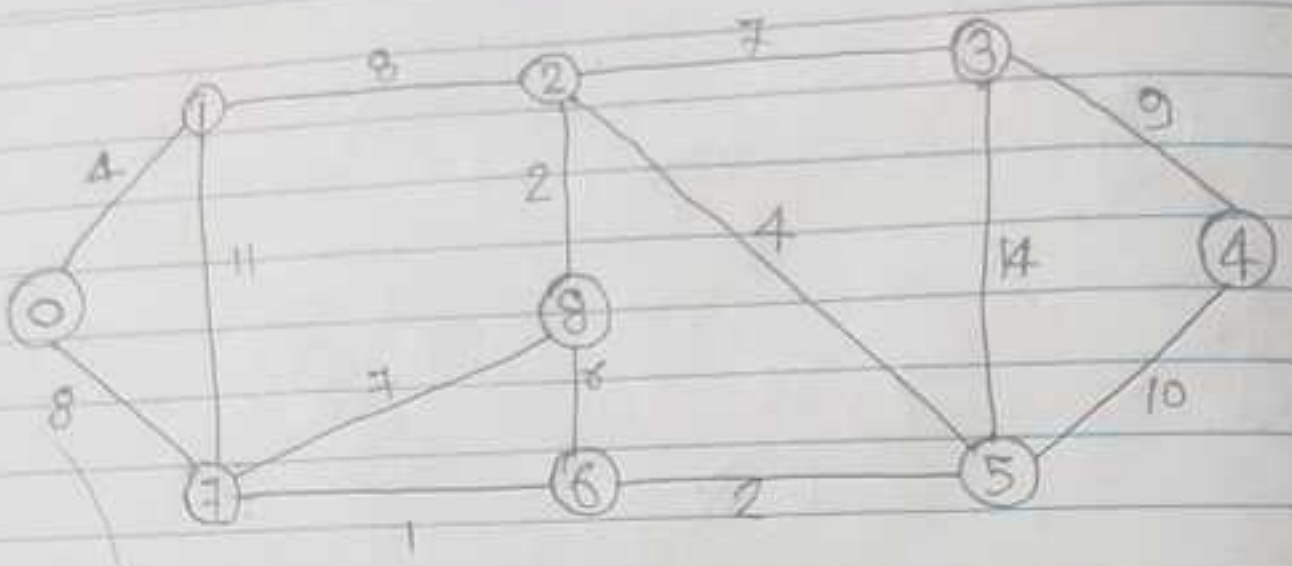
Now from the node V_2, there is a node ~~which~~ that can be reach i.e V_3 with cost of 4. So from the source node v_0 we can reach V_3 with cost of 9 as shown in figure The values of d are updated as shown in above figure the minimum cost is selected from the available ages which is V_2 to V_3. The node V_3 is now visited as shown in figure. finally the node V_4 is visited and marked as shown in figure. The variable d represents the minimum cost from source node V_0 from all other nodes



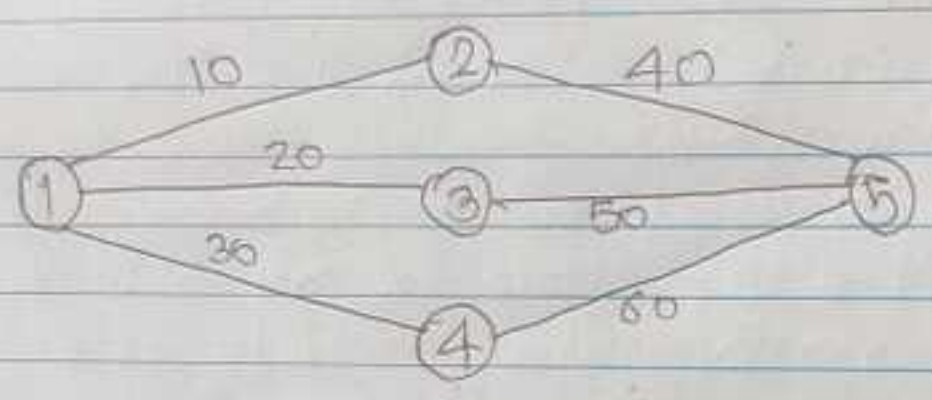
$S = \{ V_0, V_1, V_2, V_3 \}$

$S = \{ V_0, V_1, V_2, V_3, V_4 \}$

Q.1



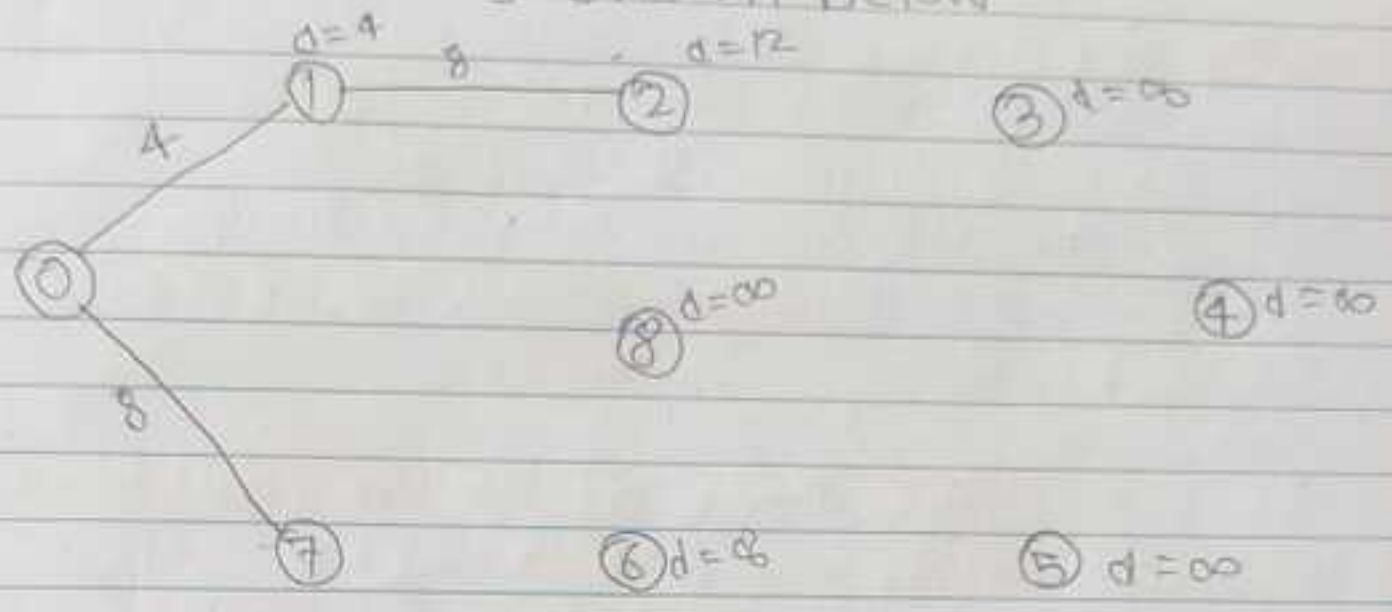
Q.3



Q.2

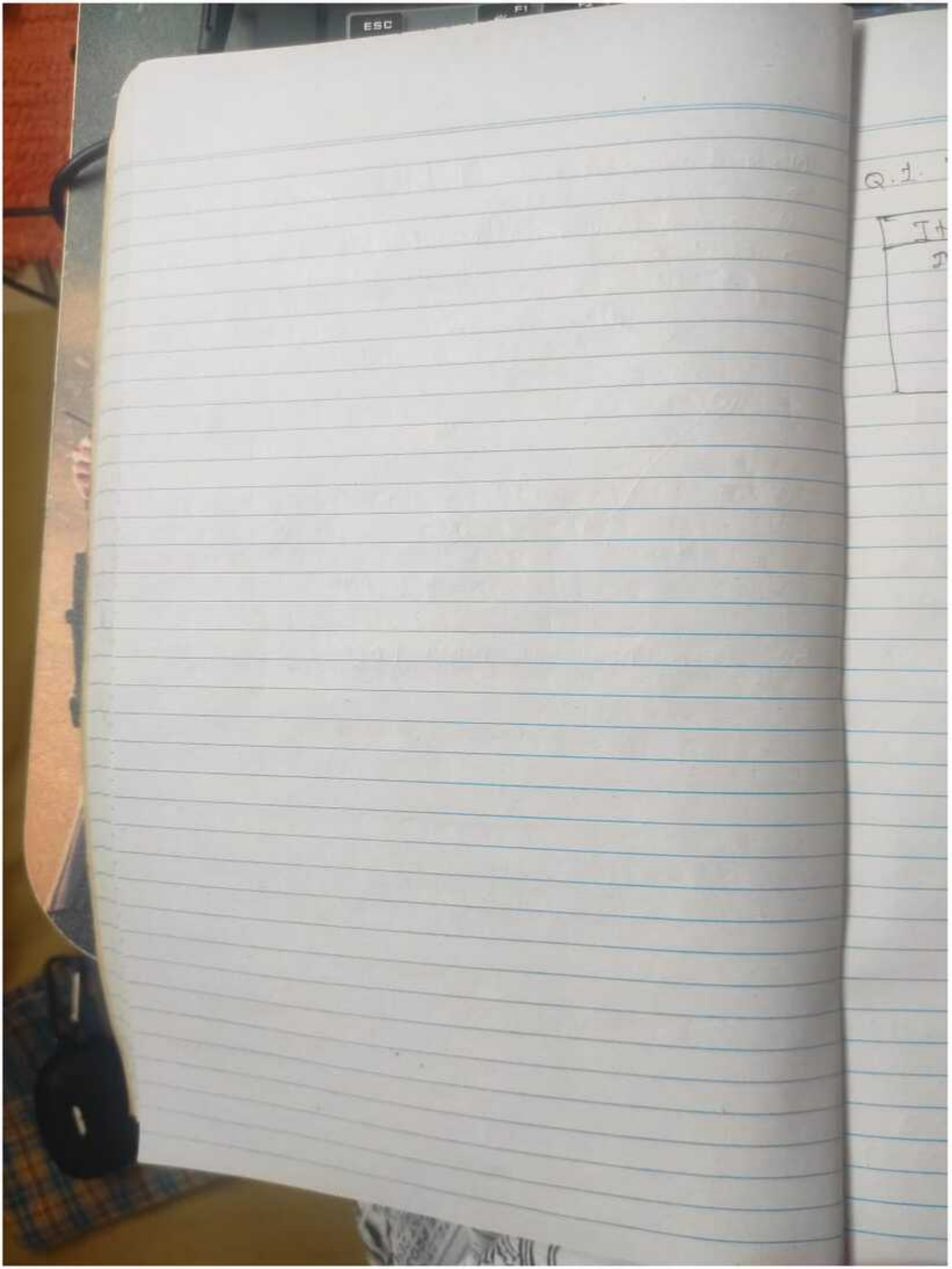
The Starting node is 0, from node 0 we can reach to the 0 to 1 and 0 to 7 with a cost of 4 and 8 respectively. The minimum cost selected from a viable ages is 0 to 1. The variable d represents the total cost from source node 0. The remaining values of d are shown as ∞ because, from source node other nodes are still be visited, the node 1 is now marked as visited as shown below

Now, from the node 1, there are two nodes that can be reach that (7) and (8) with a cost of 11 and 8, so now choose path 1 to 2 with a cost of 8 as shown below



$S = \{0, 1\}$

So, from now we mark node 2 as visited


ESC
F1
Q.1.

Q.1. Table

Iteration	S	W	$d[v_1]$	$d[v_2]$	$d[v_3]$	$d[v_4]$
Initial	$\{v_0\}$	....	2	∞	∞	10
1	$\{v_0, v_1\}$	v_1	2	5	∞	9
2	v_0, v_1, v_2	v_2	2	5	9	9
3	v_0, v_1, v_2, v_3	v_3	2	5	9	9
4	$\{v_0, v_1, v_2, v_3, v_4\}$	v_4	2	5	9	9