Chapter-3 Greedy Method.

Greedy Method technique is a most straight forward and can be applied to a wide variety of problems. The problem has n-inputs and requires us to obtain a subset of sol^{n}s which satisfies given constraints. Any subset that satisfies these constraints is called feasible solution.

We need to find feasible sol^n that either maximizes or minimizes a given objective function. A feasible sol^n that does this is called as optimal solution.

"A Greedy alg. is an alg. that always tries to find the best sol^n for each subproblem, with the hope that this will give an optimal sol^n for the problem. Greedy algs. are simple and straight forward and are easy to invent, easy to implement and most of the time quite efficient."

Control Abstraction for Greedy Method.

3m

```
Alg. Greedy (A, n)
{
   Solution = 0; // Initialise the soln
   for (i=1 to n) do
   {
      x = Select (A);
      if Feasible (Solution, x) then
         solution = Union (Solution, x);
   }
   return (solution);
}
```

Where A is an array of 'n' inputs.
The function 'select ()' selects an input from A.
Feasible is a boolean.
Union combines x with solution and updates the objective function.

Optimal Storage on Tapes.

The Optimal Storage on Tapes is a minimization problem.

Given n programs that are to be stored on a magnetic tape of 'L' size. Each program has its own length 'li'.

The constraints are

$$\sum_{i=1}^{n} l_i \leq L$$

Every time a program is retrieved, the tape is initially positioning at the beginning.
The objective of this problem is to minimize the overall retrieved time.

Eg: let $n = 3$, $l_1 = 5$, $l_2 = 10$, $l_3 = 3$, the max. Storage size $L = 20$.

let T(i) is the time taken from retrieving the program from the tape. There are 3 factorial combinations possible.

$T(1) = [l_1, l_2, l_3] = l_1 + (l_1+l_2) + (l_1+l_2+l_3) = 5 + 15 + 18 = 38$
$T(2) = [l_1, l_3, l_2] = l_1 + (l_1+l_3) + (l_1+l_3+l_2) = 5 + 8 + 18 = 31$
$T(3) = [l_2, l_1, l_3] = l_2 + (l_2+l_1) + (l_2+l_1+l_3) = 10 + 15 + 18 = 43$
$T(4) = [l_2, l_3, l_1] = l_2 + (l_2+l_3) + (l_2+l_3+l_1) = 10 + 13 + 18 = 41$
$T(5) = [l_3, l_1, l_2] = l_3 + (l_3+l_1) + (l_3+l_1+l_2) = 3 + 8 + 18 = 29$
$T(6) = [l_3, l_2, l_1] = l_3 + (l_3+l_2) + (l_3+l_2+l_1) = 3 + 13 + 18 = 34$

From the above calculations, we observed that T(5) is the optimal soln which means programs should be stored in the order of (l_3, l_1, l_2).
∴ T(5) is the min. time taken for retrieving the program from the tape.

KNAPSACK Problem.

We are given 'n' objects and a bag, each object has a weight 'w' and the maximum capacity of the bag is 'M'. Each object is associated with a profit 'P' i.e., earned when it is placed in the bag (KNAPSACK).

we have an option of placing the object as a whole (or) as a fraction. If fraction is not allowed then it is called zero / one knapsack problem [0/1 KP].

Objective: we must fill the bag which maximizes the profit earned and the output is a vector (x_i) where $1 \le i \le n$.

Constraints: The total capacity cannot exceed 'M' at any time. The problem can be stated mathematically as

$$\sum_{1 \le i \le n} P_i x_i \quad \text{subject to} \quad \sum_{1 \le i \le n} w_i x_i \le M.$$

To find the solution for knapsack problem to follow three different strategies.

Strategy1: Select the objects based on increasing order of weights.

Strategy2: Select the objects based on decreasing order of profits.

Strategy 3: Select the objects based on decreasing order of profit / weight $[P_i/w_i]$.

Algorithm:

Assume that objects have been arranged in decreasing order of P_i / w_i.

let 'M' is knapsack (bag) capacity.

'n' is the no. of objects.

w[1:n] is array of weights.

P[1:n] is array of profits.

x is a solution vector.

```
Algorithm Knapsack (M, n, w, P, x)
{
   remaining capacity (Rc) = M
    for i=1 to n do
        X[i] = 0
    end for
    for (i=1 to n do
    {
      if [w[i] < Rc)
          x[i] = 1;
          Rc = Rc - w[i];
       else
          x[i] = Rc / w[i];
          break;
    }
    end for
    sum = 0;
    for (i=0 to n do
       sum = sum + p[i] * x[i];
    end for
}
```

from the abg, we conclude that the time complexity of $T(n) = O(n)$.

Algorithm:

Assume that objects have been arranged in decreasing order of P_i / w_i.

let 'M' is knapsack (bag) capacity.

'n' is the no. of objects.

w[1:n] is array of weights.

P[1:n] is array of profits.

x is a solution vector.

```
Algorithm Knapsack (M, n, w, P, X)
{
    remaining capacity (Rc) = M
    for i=1 to n do
        X[i] = 0
    end for
    for (i=1 to n do
    {
      if (w[i] < Rc)
          x[i] = 1;
          Rc = Rc - w[i];
       else
          x[i] = Rc / w[i];
          break;
    }
    end for
    sum = 0;
    for (i=0 to n do
        sum = sum + p[i] * x[i];
    end for
}
```

From the algo, we conclude that the time complexity of $T(n) = O(n)$.

Consider a knapsack problem of finding the optimal solution where 'M' = 40, n = 4

weights $[w_1, w_2, w_3, w_4] = [20, 25, 10, 15]$.

Profits $[P_1, P_2, P_3, P_4] = [20, 40, 35, 45]$.

Soln

Strategy 1:

Objects	3	4	1	2
weights	10	15	20	25
profits	35	45	20	40

R_c	Object selected	weight	fraction of x added to knapsack.
40	3	10	1 full unit stored
(40-10) 30	4	15	1 full unit stored
(30-15) 15, 0	1	20	3/4th unit stored.

The object 3 and 4 are added to the knapsack (bag) as a full unit. The object 1 is added as 3/4 th of unit.

The object 2 is not selected because the knapsack (bag) exceeds the maximum weight 40.

The solution vector $(x_1, x_2, x_3, x_4) = (3/4, 0, 1, 1)$.

$\therefore$ Profit earned $= (x_1 * P_1 + x_2 * P_2 + x_3 * P_3 + x_4 * P_4)$

$= (3/4 * 20 + 0 * 40 + 1 * 35 + 1 * 45)$

$= 15 + 0 + 35 + 45$

profit earned = 95 accord^n to Strategy 1.

Strategy 2:

Objects	4	2	3	1
weights	15	25	10	20
profits	45	40	35	20

R_c	Object selected	weight	Fraction of x added to Knapsack.
40	4	15	1 full unit stored
(40-15) 25	2	25	1 full unit stored
0			

The object 4 and 2 are added to the knapsack (bag) as 1 full unit. and object 1 & 3 are not selected.

The solution vector $(x_1, x_2, x_3, x_4) = (0, 1, 0, 1)$.

$\therefore$ The profit earned $= (x_1 * P_1 + x_2 * P_2 + x_3 * P_3 + x_4 * P_4)$

$= (0 * 20 + 1 * 40 + 0 * 35 + 1 * 45)$

$= 40 + 45$

profit earned = 85. Accord^n to strategy 2.

Strategy 3:

$$\frac{P_1}{w_1} = \frac{20}{20} = 1 \qquad \frac{P_2}{w_2} = \frac{40}{25} = 1.6$$

$$\frac{P_3}{w_2} = \frac{35}{10} = 3.5 \qquad \frac{P_4}{w_4} = \frac{45}{15} = 3$$

3.5	3	1.6	1
w_3	w_4	w_2	w_1
P_3	P_4	P_2	P_1

Objects	3	4	2	1
weights	10	15	25	20

Rc	Object selected	weight	fraction of x added to knapsack.
40	3	10	1 full unit stored
(40-10)30	4	15	1 full unit stored
(30-15) 15	2	25	3/5 unit stored

Solution vector $(x_1, x_2, x_3, x_4) = (0, 3/5, 1, 1)$.

$\therefore$ The profit earned $= (x_1 * P_1 + x_2 * P_2 + x_3 * P_3 + x_4 * P_4)$

$= (0 * 20 + 3/5 * 40 + 1 * 35 + 1 * 45)$

$= 24 + 35 + 45$

$\therefore$ profit earned = 104 acedn to Strategy 3.

According to strategy1 profit earned = 95
According to strategy2 profit earned = 85
According to strategy3 profit earned = 104.

In the 3rd strategy the profit earned is more, therefore the Optimal solution is 104.

2. Consider knapsack prob for finding the optimal soln where M=15 n=7

	1	2	3	4	5	6	7
weights	4	2	3	2	4	2	1
Profits	2	3	15	10	5	8	4.

Soln

Strategy1:

Objects	7	2	4	6	3	1	5
weights	1	2	2	2	3	4	4
profits	4	3	10	8	15	2	5

Rc	object selected	weight	fraction of x added to knapsack.
15	7	1	1 full unit
(15-1) 14	2	2	1 full unit

(8-3)	5	1	4	1 full unit
(5-4)	1	5	4	1/4 unit

Solution vector $(x_1, x_2, x_3, x_4, x_5, x_6, x_7)$
$= (1, 1, 1, 1, 1/4, 1, 1)$.

$\therefore$ profit earned $= (x_1*P_1 + x_2*P_2 + x_3*P_3 + x_4*P_4 + x_5*P_5 + x_6*P_6 + x_7*P_7)$.

$= (1*2 + 1*3 + 1*15 + 1\cdot10 + \frac{1}{4}\cdot5 + 1\cdot8 + 1\cdot4)$

$= 2 + 3 + 15 + 10 + 1\cdot25 + 8 + 4$

pe $= 43.25$ accd^n to Strategy 1.

Strategy 2:

objects	3	4	6	5	7	2	1
weights	3	2	2	4	1	2	4
Profits	15	10	8	5	4	3	2

R_c	object selected	weight	Fraction of x added to knapsack.
15	3	3	1 full unit.
(15-3) 12	4	2	1 full unit
(12-2) 10	6	2	1 full unit
(10-2) 8	5	4	1 full unit
(8-4) 4	7	1	1 full unit
(4-1) 3	2	2	1 full unit
(3-2) 1	1	4	1/4 unit

Solution vector $(x_1, x_2, x_3, x_4, x_5, x_6, x_7)$
$= (1/4, 1, 1, 1, 1, 1, 1)$

$\therefore$ profit earned $= (x_1*P_1 + x_2*P_2 + x_3*P_3 + x_4*P_4 + x_5*P_5 + x_6*P_6 + x_7*P_7)$

$= (1/4*2 + 1*3 + 1\cdot15 + 1\cdot10 + 1\cdot5 + 1\cdot8 + 1\cdot4)$

$= (0.5 + 3 + 15 + 10 + 5 + 8 + 4)$

Strategy 3:

$\frac{P_1}{w_1} = \frac{2}{4} = 0.5$ $\frac{P_2}{w_2} = \frac{3}{2} = 1.5$ $\frac{P_3}{w_3} = \frac{15}{3} = 5$

$\frac{P_4}{w_4} = \frac{10}{2} = 5$ $\frac{P_5}{w_5} = \frac{5}{4} = 1.25$ $\frac{P_6}{w_6} = \frac{8}{2} = 4$ $\frac{P_7}{w_7} = \frac{4}{1} = 4$

Objects	3	4	6	7	2	5	1
weights	3	2	2	1	2	4	4
Profits	15	10	8	4	3	5	2

R_c	Object selected	weight	fraction of x added to knapsack.
15	3	3	1 full unit
(15-3) 12	4	2	1 full unit
(12-2) 10	6	2	1 full unit
(10-2) 8	7	1	1 full unit
(8-1) 7	2	2	1 full unit
(7-2) 5	5	4	1 full unit
(5-4) 1	1	4	1/4 unit.

Solution vector $(x_1, x_2, x_3, x_4, x_5, x_6, x_7)$
$= (1/4, 1, 1, 1, 1, 1, 1)$

$\therefore$ profit earned $= (x_1 * P_1 + x_2 * P_2 + x_3 * P_3 + x_4 * P_4 + x_5 * P_5 + x_6 * P_6 + x_7 * P_7)$

$= (1/4 \times 2 + 1 \times 3 + 1.15 + 1.10 + 1.5 + 1.8 + 1.4)$

$= (0.5 + 3 + 15 + 10 + 5 + 8 + 4)$

$\therefore$ PE $= 45.5$ accdn to Strategy 3

Accdn to strategy 1 PE is 43.25

$\therefore$ Strategy 2 / Strategy 3 are the more profit Earned.

0/1 Knapsack problem.

This problem is similar to Knapsack problem. Except that the solution vector x_i values can take either 0 (or) 1, but no fractions are allowed.

Problem:

Consider for a 0/1 Knapsack problem. $M = 40$ and $n = 4$

weights: $[w_1, w_2\ w_3\ w_4] : [20\ 25\ 10\ 15]$

profits: $[P_1\ P_2\ P_3\ P_4] : [30\ 35\ 40\ 20]$.

Soln

Strategy 1:

Objects	3	4	1	2
weights	10	15	20	25
profits	40	20	30	35

R_c	Object selected	weights	Knapsack.
40	3	10	1 full unit
(40-10) 30	4	15	1 full unit
(30-15) 15	0		

Solution vector: $[x_1\ x_2\ x_3\ x_4] = [0, 0, 1, 1]$

profit earned $= (x_1 * P_1 + x_2 * P_2 + x_3 * P_3 + x_4 * P_4)$

$= (0 + 0 + 1 * 40 + 1 * 20)$

$\therefore$ PE = 60 acc^{n} to strategy 1.

Strategy 2:

Objects	3	2	1	4
weights	10	25	20	15
profits	40	35	30	20

Rc	Object selected	weights	Knapsack.
40	3	10	1 full unit
(40-10) 30	2	25	1 full unit
(30-25) 5			

Solution vector $[x_1, x_2, x_3, x_4] : [0, 1, 1, 0]$

Profit earned $= (x_1 * P_1 + x_2 * P_2 + x_3 * P_3 + x_4 * P_4)$

$= (0 + 1 * 35 + 1 * 40 + 0)$

$\therefore$ PE = 75 accd^n to strategy 2.

Strategy 3:

$\frac{P_1}{w_1} = \frac{30}{20} = 1.5$ $\quad \frac{P_2}{w_2} = \frac{35}{25} = 1.4$

$\frac{P_3}{w_3} = \frac{40}{10} = 4$ $\quad \frac{P_4}{w_4} = \frac{20}{15} = 1.3$

Objects	3	1	2	4
weights	10	20	25	15
Profits	40	30	35	20

Rc	Object selected	weights	Knapsack.
40	3	10	1 full unit
(40-10) 30	1	20	1 full unit
[30-20) 10			

Solution vector: $(x_1, x_2, x_3, x_4) : (1, 0, 1, 0)$

Profit earned $= (x_1 * P_1 + x_2 * P_2 + x_3 * P_3 + x_4 * P_4)$

$= (1 * 30 + 0 + 1 * 40 + 0)$

$\therefore$ PE = 70 accd^n to strategy 3

In the strategy 2 PE is more

Job-Sequencing with Deadline.

It deals with a set of 'n' jobs. associated with each job 'i' is an integer deadline $d_i \geq 0$ and a profit $P_i \geq 0$. for any job i, the profit P_i is earned if the job is completed by its deadline. To complete a job one have to process the job on a machine for one unit of time.

A feasible soln for this problem is a subset 'J' of jobs such that each job in this subset can be completed by its deadline. The value of the feasible soln 'J' is the sum of the profits of the jobs in 'J'.

i.e., $$\sum_{i \in J} P_i$$

An optimal soln is a feasible soln with max. value.

The job that has the largest profit is added. It forms feasible soln.

Problem:

1. Solve the job sequencing problem. Given $n = 5$, Profits [1, 5, 20, 15, 10] and Deadlines [1, 2, 4, 1, 3]. using Greedy strategy.
[The Max. Deadline is 4 units of time].

Soln:

Since the Max. deadline is 4 units of time, the feasible soln set must have less than or equal to 4 jobs.

Arrange the jobs in the decreasing order of profits.

Jobs	3	4	5	2	1
profits	20	15	10	5	1
deadlines	4	1	3	2	1

Steps	feasible soln	processing sequence	Profit
1	{3}	{3}	20
2	{3,4}	{4,3}	20+15=35
3	{3,4,5}	{4,5,3}	20+15+10=45
(4)	{3,4,5,2}	{4,2,5,3}	20+15+10+5=50.
5	{3,4,5,1}	{4,1,5,3}	20+15+10+1=46.

Step 4 is an optimal soln. The jobs must be processed in the order of {4,2,5,3} and value of the optimal soln is 50.

2. Solve the following problem of job sequencing with the deadlines specifies using Greedy strategy. where no. of jobs, $n=4$
Profits [100, 10, 15, 27]. and Deadlines [2, 1, 2, 1].
Deadline time is 2 units.

Jobs	1	4	3	2
profits	100	27	15	10
deadlines	2	1	2	1

Steps	feasible soln	processing sequence	Profit.
1	{1}	{1}	100
(2)	{1,4}	{4,1}	100+27=127.
3	{1,3}	{1,3}	100+15=115.
4	{1,2}	{2,1}	100+10=110.

Step 2 is an optimal soln. The jobs must be processed in the order {4,1} & the optimal value is 127.

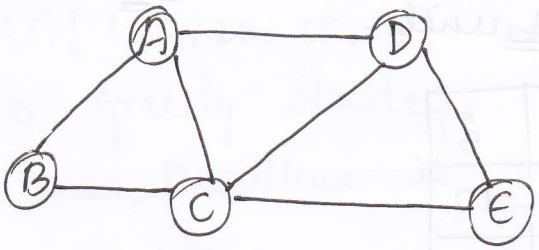
Minimum Spanning Tree.

Let $G = (V, E)$ be an undirected connected Graph. A subgraph $T = (V, E')$ of G is a spanning tree. A spanning tree of a Graph include all the vertices and a subset of edges E'.

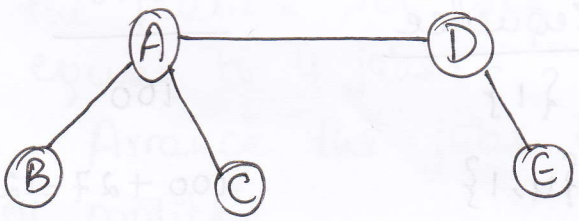
Minimum cost Spanning Tree [MCST]

The problem is to find a fixed connected subgraph, containing all the vertices such that the sum of the cost of the edges in the sub-graph is minimum.

If any cycle had been present then we could have broken it by deleting one of its edge, the graph still be connected but the cost would be smaller and all cost are positive. This subgraph is called the minimum cost spanning tree.



Graph



Spanning Tree

There are 2 algorithms that are popularly used for finding min. spanning tree.

1. Kruskal's alg.
2. Prim's alg.

Kruskal's alg.

The set of ~~Edges~~ Vertices (T) is initially empty, as the alg. progresses edges are added to the set T contains the vertices at every instance. At the end of the alg. only the connected component remains, so that T is then a min. spanning tree of all the vertices of Graph G.

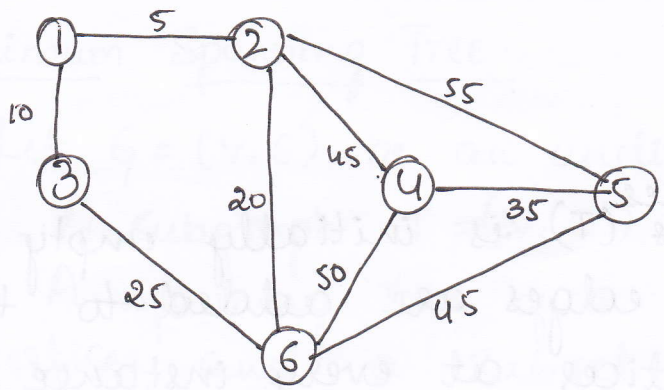
Algorithm

let T be the Spanning Tree
E be the list of edges
n be the no. of nodes given in Graph G.

```
Algorithm Kruskal(T, E, n)
{
  initially T=0;
  while [T ≠ n-1 and E ≠ 0]
  {
    find the min-cost edge in E and call it as (u,v)
    if (u,v) doesnot form a cycle.
      T = T + (u,v); // add (u,v) to T
    else
      delete (u,v); // (u,v) has already become considered
  }
  if (T==n-1)
    Display "Spanning Tree T";
  else
    Display "No-Spanning Tree";
}
```

Eg:

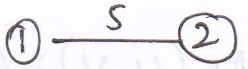
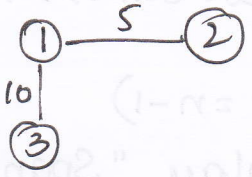
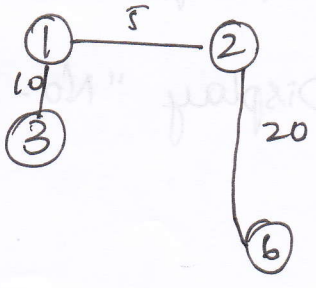
1- Find the MCST for the given Graph using Kruskal's alg.



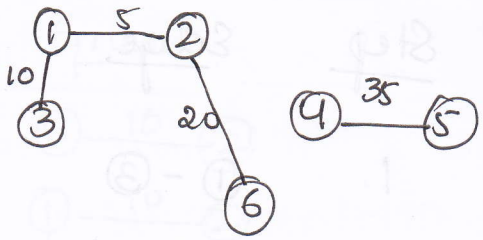
Soln

Arrange the edges in the increasing order of their costs.

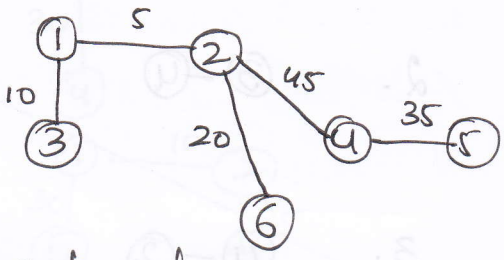
Edge	cost
① – ②	5
① – ③	10
② – ⑥	20
* ③ – ⑥	25
④ – ⑤	35
② – ④	45
⑥ – ⑤	45
⑥ – ④	50
② – ⑤	55

Step	Edge	connected component	Graph.
1	① – ②	T[1, 2]	
2	① – ③	T[1, 2, 3]	
3	② – ⑥	T[1, 2, 3, 6]	

4. ④ – ⑤ T [1, 2, 3, 6, 4, 5]

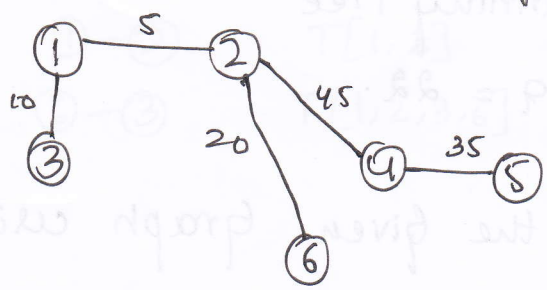


5. ② – ④ T [1, 2, 3, 6, 4, 5]



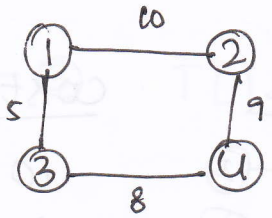
Now T contains the choosed edges
[1-2], [1-3] [2-6] [4-5] [2-4]

Then the min. Spanning tree is as follows.



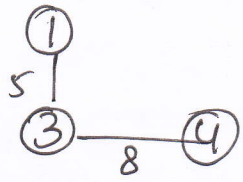
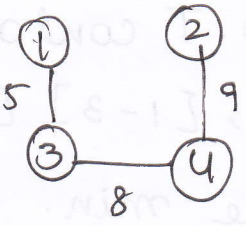
∴ The total min. cost = 5 + 10 + 20 + 45 + 35
= 115

2. Find the MCST for the given below Graph using Kruskal's alg.



Soln

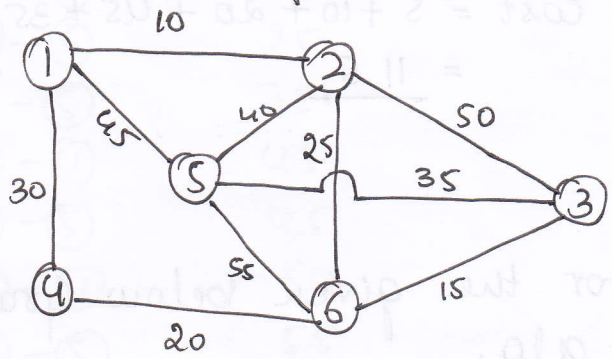
Edge	cost.
① – ③	5
③ – ④	8
④ – ②	9
② – ①	10

Step	Edge	connected component	Graph.
1.	① - ③	T [1, 3]	
2.	③ - ④	T [1, 3, 4]	
3.	④ - ②	T [1, 3, 4, 2]	

∴ Total cost min spanning Tree

$= 5 + 8 + 9 = 22.$

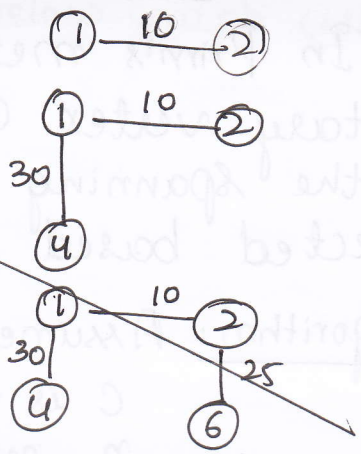
3. Find the MCST for the given Graph using Kruskal's alg.



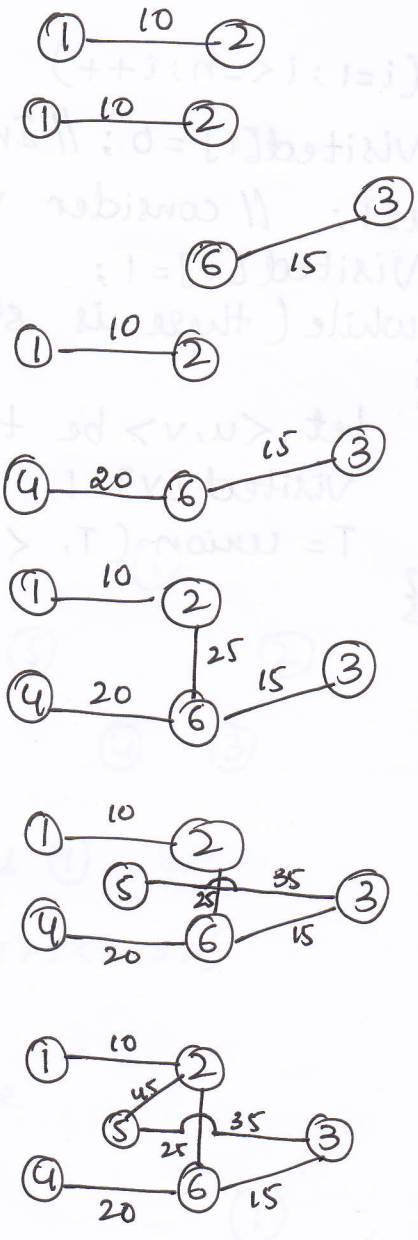
Edge	cost
① — ②	10
① — ④	30
② — ⑥	25
④ — ⑥	20
① — ⑤	45
⑤ — ②	40
⑤ — ⑥	55
⑤ — ③	35
② — ③	50
⑥ — ③	15

Edge	cost
① — ②	10
⑥ — ③	15
④ — ⑥	20
② — ⑥	25
① — ④	30
⑤ — ③	35
⑤ — ②	40
① — ⑤	45
② — ③	50
⑤ — ⑥	55

		component	graph
1.	①—②	T[1,2]	
2.	①—④	T[1,2,4]	
3.	②—⑥	T[1,2,4,6]	
4.	①—⑤		



Step	Edge	connected component	Graph
1.	①—②	T[1,2]	
2.	⑥—③	T[1,2,3,6]	
3.	④—⑥	T[1,2,3,6,4]	
4.	②—⑥	T[1,2,3,6,4]	
5.	⑤—③	T[1,2,3,6,4,5]	
6.	⑤—②	T[1,2,3,6,4,5]	



∴ Total cost min. Spanning Tree

$= 10 + 15 + 20 + 25 + 35 + 45$

$=$

Prim's Algorithm

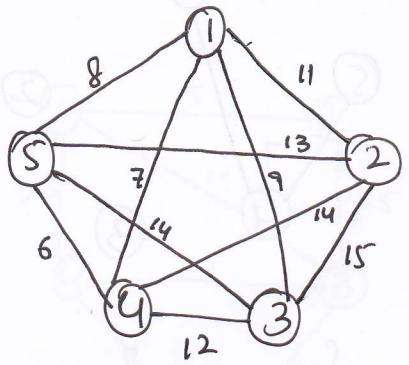
In Prim's method, we start with some orbitary vertex and add the min. cost edge to the spanning tree. The next vertex is selected based on the min. cost edge.

Algorithm: Assume G is connected Graph
C is the cost adjacency matrix.
n - no. of vertices
T - minimum weight Spanning Tree.

```
Alg. Prims (C,n)
{
 for (i=1; i<=n; i++)
    visited[i]=0; // Initialise all vertices as unvisited.
   u=1;  // consider vertex1 as starting.
   visited[i]=1;
   while (there is still unclosed vertex) do
   {
      let <u,v> be the min-cost edge
      visited[v]=1;
      T= union (T, <u,v>); // add edge to Spanning Tree
   }
}
```

Problems.

1. Find the MCST for the given below Graph using Prim's alg.

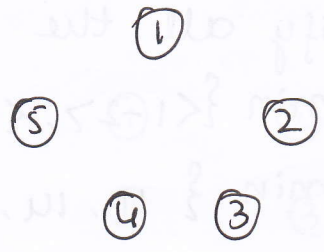


Soln construct the adjacency matrix.

	1	2	3	4	5
1	-	11	9	7	8
2	11	-	15	14	13
3	9	15	-	12	14
4	7	14	12	-	6
5	8	13	14	6	-

Step 1: $T = \{\}$

$G = \{1, 2, 3, 4, 5\}$



Step 2: Assume starting node as (1)

$= \min \{ \langle 1-2 \rangle, \langle 1-3 \rangle, \langle 1-4 \rangle, \langle 1-5 \rangle \}$

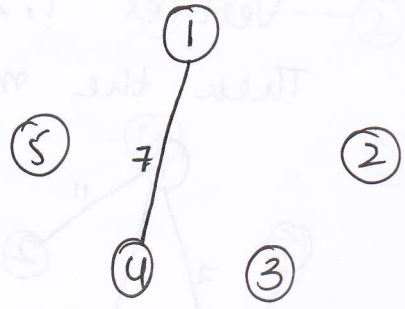
$= \min \{ 11, 9, 7, 8 \}$

$\therefore$ 7 is the minimum edge

i.e., 1 – 4

Now add $T = \{1, 4\}$

$G = \{2, 3, 5\}$



Step 3: Verify all the edges from 1 & 4

$= \min \{\langle 1-2\rangle, \langle 1-3\rangle, \langle 1-5\rangle, \langle 4-2\rangle, \langle 4-3\rangle, \langle 4,5\rangle\}$

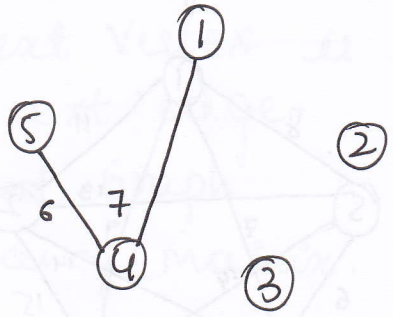
$= \min \{11, 9, 8, 14, 12, 6\}$

$\therefore$ 6 is the min. edge

i.e., 4-5

Now add $T = \{1, 4, 5\}$

$Q = \{2, 3\}$.



Step 4: Verify all the edges from $\{1, 4, 5\}$

$= \min \{\langle 1-2\rangle, \langle 1-3\rangle, \langle 4-2\rangle, \langle 4-3\rangle, \langle 5-2\rangle, \langle 5-3\rangle\}$

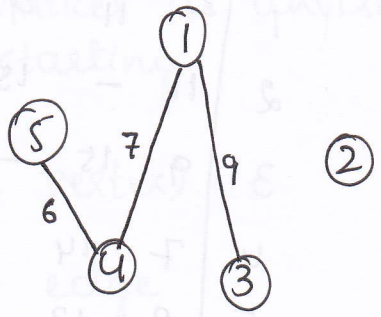
$= \min \{11, 9, 14, 12, 13, 14\}$.

$\therefore$ 9 is the min. edge.

i.e., $\langle 1-3\rangle$

Now add $T = \{1, 3, 4, 5\}$

$Q = \{2\}$



Step 5: Verify all the edges from $\{1, 4, 5, 3\}$

$= \min \{\langle 1-2\rangle, \langle 4-2\rangle, \langle 5-2\rangle, \langle 3-2\rangle\}$

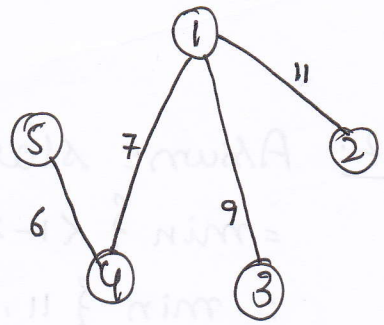
$= \min \{11, 14, 13, 15\}$

$\therefore$ 11 is the min. edge

i.e., $\langle 1-2\rangle$

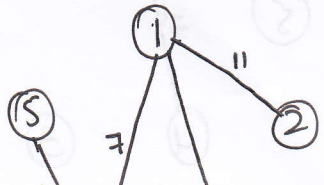
Now add $T = \{1, 4, 5, 3, 2\}$

$Q = \{\}$.



$\therefore$ Now T contains all chosed edges of vertices 1, 2, 3, 4, 5.

Then the min. Spanning tree has

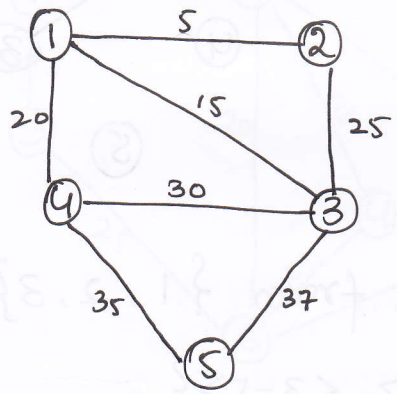


∴ The total minimum cost = <1-4> + <4-5> + <1-3> + <1-2>

= 7 + 6 + 9 + 11

= 33.

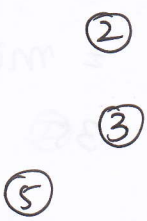
2.



	1	2	3	4	5
1	-	5	15	20	-
2	5	-	25	-	-
3	15	25	-	30	37
4	20	-	30	-	35
5	-	-	37	35	-

Step 1: T = { }

Q = {1, 2, 3, 4, 5}.



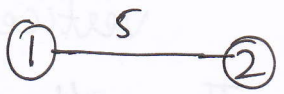
Step 2: Assume starting nodes as (1)

= min {<1-2>, <1-3>, <1-4>, <1-5>}

= min {5, 15, 20}.

∴ 5 is the min. edge

i.e., <1-2>



Now add T = {1, 2}

Q = {3, 4, 5}

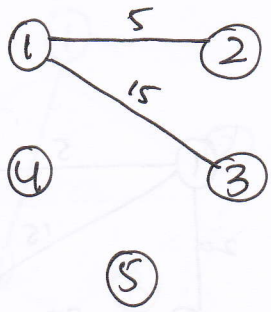
Step 3: Verify all the edges from 1 & 2.

$= \min\{\langle 1-3\rangle, \langle 1-4\rangle, \cancel{\langle 1-5\rangle}, \langle 2-3\rangle, \cancel{\langle 2-4\rangle}, \cancel{\langle 2-5\rangle}\}$

$= \min\{15, 20, 25\}$

$\therefore$ 15 is the min. edge

i.e., $\langle 1-3\rangle$



Now add $T = \{1, 2, 3\}$

$Q = \{4, 5\}$

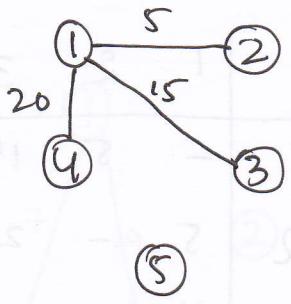
Step 4: Verify all the edges from $\{1, 2, 3\}$

$= \min\{\langle 1-4\rangle, \cancel{\langle 2-4\rangle}, \langle 3-4\rangle, \langle 3-5\rangle\}$

$= \min\{20, 30, 37\}$

$\therefore$ 20 is the min. edge

i.e., $\langle 1-4\rangle$



Now add $T = \{1, 2, 3, 4\}$

$Q = \{5\}$

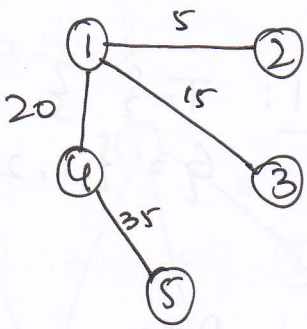
Step 5: Verify all the edges from $\{1, 2, 3, 4\}$

$= \min\{\langle 3-5\rangle, \langle 4-5\rangle\}$

$= \min\{37, 35\}$

$\therefore$ 35 is the min. edge

i.e., $\langle 4-5\rangle$

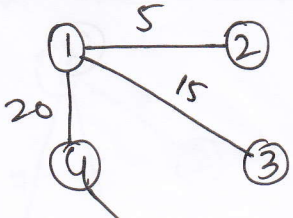


Now add $T = \{1, 2, 3, 4, 5\}$

$Q = \{\}$

$\therefore$ Now T contains all chosen edges of vertices 1, 2, 3, 4, 5

Then the min. Spanning tree has

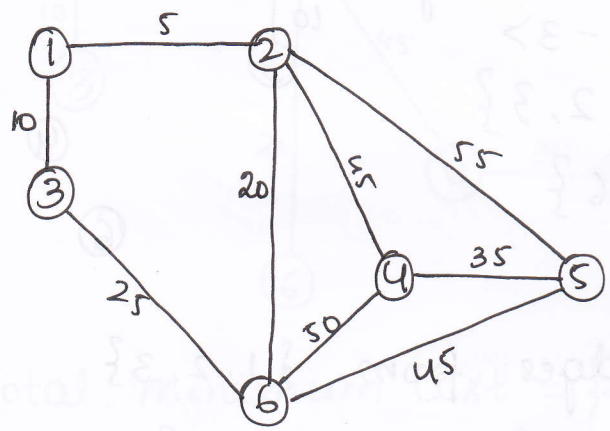


∴ Total min. cost = { <1-2> + <1-3> + <1-4> + <4-5> }

= { 5 + 15 + 20 + 35 }

= 75.

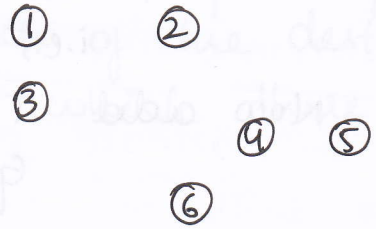
3.



	1	2	3	4	5	6
1	-	5	10	-	-	-
2	5	-	-	45	55	20
3	10	-	-	-	-	25
4	-	45	-	-	35	50
5	-	55	-	35	-	45
6	-	20	25	50	45	-

Step 1: T = { }

Q = {1, 2, 3, 4, 5, 6}



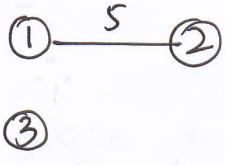
Step 2: Assume starting node as (1)

= min { <1-2>, <1-3>, ~~<1-4>~~, ~~<1-5>~~, ~~<1-6>~~ }

= min { 5, 10 }.

∴ 5 is the min. edge

i.e., <1-2>



Now add T = { 1, 2 }

Step 3: Verify all the edges from 1 & 2

$= \min\{\langle 1-3\rangle, \langle 2-4\rangle, \langle 2-5\rangle, \langle 2-6\rangle\}$

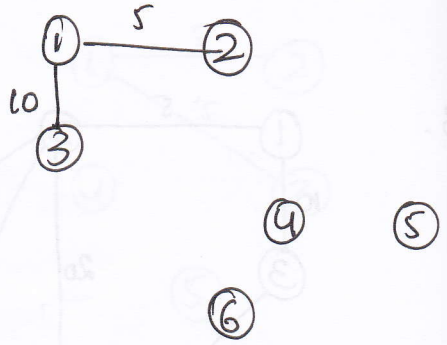
$= \min\{10, 45, 55, 20\}$

$\therefore$ 10 is the min. edge

i.e., $\langle 1-3\rangle$

Now add $T = \{1, 2, 3\}$

$Q = \{4, 5, 6\}$



Step 4: Verify all the edges from $\{1, 2, 3\}$

$= \min\{\langle 2-4\rangle, \langle 2-5\rangle, \langle 2-6\rangle, \langle 3-6\rangle\}$

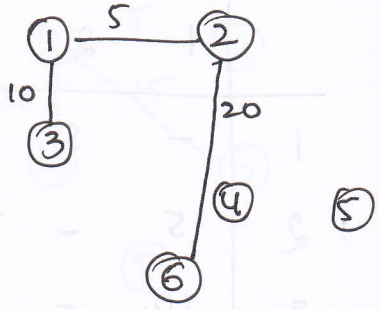
$= \min\{45, 55, 20, 25\}$

$\therefore$ 20 is the min. edge

i.e., $\langle 2-6\rangle$

Now add $T = \{1, 2, 3, 6\}$

$Q = \{4, 5\}$



Step 5: Verify all the edges from $\{1, 2, 3, 6\}$

$= \min\{\langle 2-4\rangle, \langle 2-5\rangle, \langle 6-4\rangle, \langle 6-5\rangle\}$

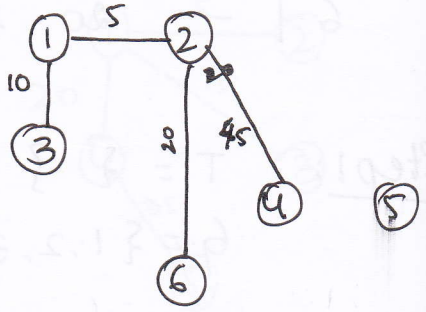
$= \min\{45, 55, 50, 45\}$

$\therefore$ 45 is the min. edge

i.e., $\langle 2-4\rangle$

Now add $T = \{1, 2, 3, 6, 4\}$

$Q = \{5\}$.



Step 6: Verify all the edges from $\{1, 2, 3, 6, 4\}$

$= \min\{\langle 2-5\rangle, \langle 6-5\rangle, \langle 4-5\rangle\}$

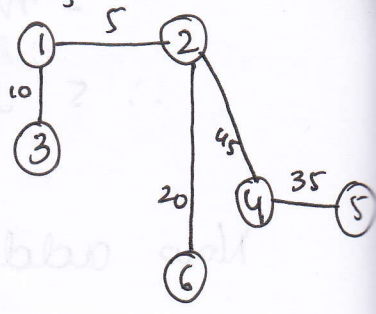
$= \min\{55, 45, 35\}$

$\therefore$ 35 is the min. edge

i.e., $\langle 4-5\rangle$

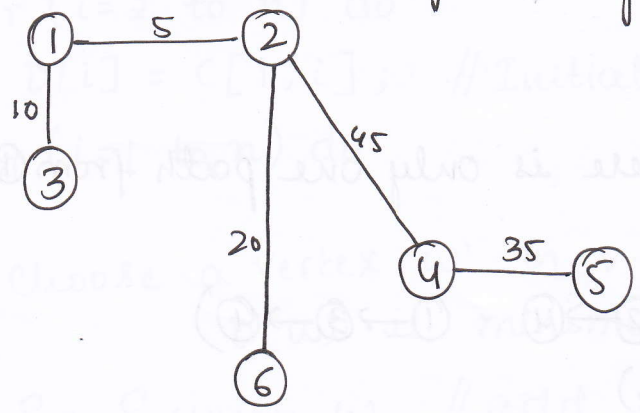
Now add $T = \{1, 2, 3, 6, 4, 5\}$

$Q = \{\}$



∴ Now T contains all chosen edges of vertices 1, 2, 3, 6, 4, 5

Then the min. Spanning tree has



∴ Total minimum cost = {<1-2>+<1-3>+<2-6>+
<2-4>+<4-5>}

= 5 + 10 + 20 + 45 + 35

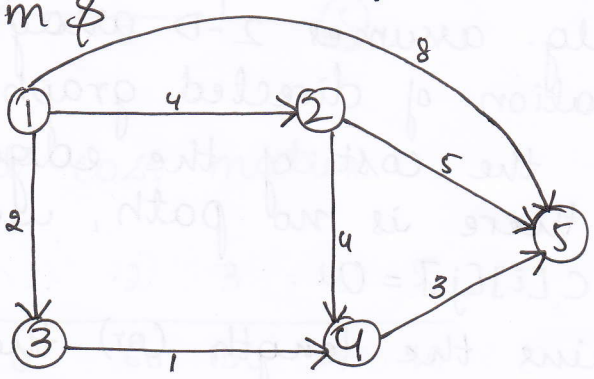
= 115

Single Source Shortest Path.

Dijikstra's Algorithm

The Dijikstra's alg. finds the shortest path from a given vertex to all the remaining vertices in a directed graph.

We have to find out a shortest path from a given source vertex 's' to each of the destinations (other vertices) in the graph to which there is a path from 's'



A motorist travelling from his place to some other place, there may be many routes but he uses the shortest path among them.

Consider the above directed graph, let the source vertex be ①, the routes to different vertices from the vertex ① as follows.

Step 1: ① — ① = 0

Step 2: ① — ② = 4 (There is only one path from ① to ②)

Step 3: ① — ③ = 2

Step 4: ① — ④ = (① → ② → ④ , ① → ③ → ④)

= (8, 3)

∴ 3 is the min. path

(There are two paths from ① to ④. Out of which the shortest path is 1-3-4 of length = 3)

Step 5: ① — ⑤ = (① → ⑤, ① → ② → ⑤, ① → ② → ④ → ⑤, ① → ③ → ④ → ⑤)

= 8, 9, 11, 6

= 6 is the shortest path.

(There are 4 paths from ① to ⑤, out of which the shortest path is ① - ③ - ④ - ⑤ of length = 6)

Dijikstra's algorithm

always produces an optimal solution.

The Dijikstra's alg. assumes 2-D array C[i][j] holds the information of directed graph. G(V, E) and C[i][j] is the cost of the edge from vertex i to j. If there is no path, we assume C[i][j] = ∞ (or) C[i][j] = 0.

D[i] contains the length (or) the current shortest path to vertex i.

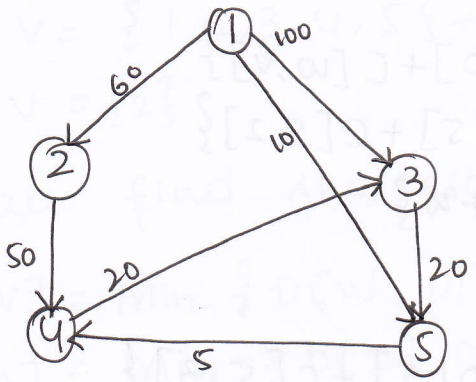
V = set of vertices.

n = no. of vertices.

```
Alg. Dijikstra's (V, C, D, n)
{
    S = { 1 };
    for (i = 2 to n) do
        D[i] = C[1, i];  // Initialise D.
    for (i = 1 to n) do
    {
        Choose a vertex 'w' in V-S such that
                D(w) is minimum.
        S = S union w  // add w to S.
        for each vertex V in V-S do
            D[V] = Min (D[V], D[w] + C[w, V]);
    }
}.
```

Problems

1. Find the optimal solution for the below mentioned directed graph from source ① to all the vertices. using Dijikstra's alg.



Soln

Construct cost matrix.

C	1	2	3	4	5
1	0	60	100	∞	10
2	∞	0	∞	50	∞
3	∞	∞	0	∞	20
4	∞	∞	20	0	∞

Step 1: $S = \{1\}$ & $V = \{1, 2, 3, 4, 5\}$.

let vertex ① be the source vertex.

Step 2: Initialize D.

$D[2] = C[1, 2] = 60$

$D[3] = C[1, 3] = 100$

$D[4] = C[1, 4] = \infty$

$D[5] = C[1, 5] = 10$

Step 3: find w and add it to S.

$= \text{Min}\{D[2], D[3], D[4], D[5]\}$.

$= \text{Min}\{60, 100, \infty, \underline{10}\}$.

$= D[5]$

↓

w

Add w to S

$S = \{1, 5\}$, $V = V - S = \{1, 2, 3, 4, 5\} - \{1, 5\}$

$V = \{2, 3, 4\}$.

~~Step 4: Initialise D (or)~~

Now, find

$D[v] = \text{Min}\{D[v], D[w] + C[w, v]\}$

$\therefore D[2] = \text{Min}\{D[2], D[5] + C[5, 2]\}$

$= \text{Min}\{60, 10 + \infty\}$

$= 60$

$D[3] = \text{Min}\{D[3], D[5] + C[5, 3]\}$

$= \text{Min}\{100, 10 + \infty\}$

$= 100$

$D[4] = \text{Min}\{D[4], D[5] + C[5, 4]\}$

$= \text{Min}\{\infty, 10 + 5\}$

$D[4] = 15$.

Step 4: ~~find w & add it to S.~~

From the above step D[4] is the minimum vertex

$\therefore w = 4$.

Now $S = \{1, 5, 4\}$

$V = V - S = \{1, 2, 3, 4, 5\} - \{1, 5, 4\}$

$V = \{2, 3\}$.

Again find the shortest path.

$D[V] = \min\{D[V], D[w] + C[w, V]\}$

$D[2] = \min\{D[2], D[4] + C[4, 2]\}$

$= \min\{60, 15 + \infty\}$

$= 60$

$D[3] = \min\{D[3], D[4] + C[4, 3]\}$

$= \min\{100, 15 + 20\}$

$D[3] = 35$.

Here w is 3. and add it to S.

Step 5: $S = \{1, 5, 4, 3\}$

$V = \{1, 2, 3, 4, 5\} - \{1, 5, 4, 3\}$

$V = \{2\}$

Again find shortest path to V.

$D[V] = \min\{D[V], D[w] + C[w, v]\}$

$D[2] = \min\{D[2], D[3] + C[3, 2]\}$

$= \min\{60, 35 + \infty\}$

$D[2] = 60$

Here $w = 2$. and add it to 2.

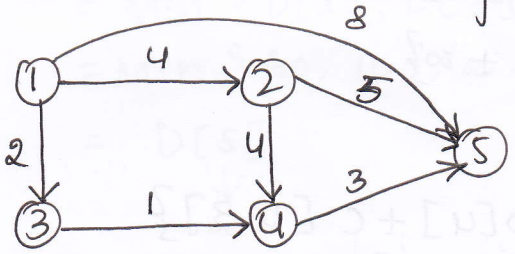
$S = \{1, 2, 3, 4, 5\}$

$V = \{\}$.

Step 6: from the Source vertex 1 to all other vertices

D[2] = 60 → least path from ① – ②
D[3] = 35 → least path from ① – ③
D[4] = 15 → least path from ① – ④
D[5] = 10 → least path from ① – ⑤.

2. Find the shortest path from the source vertex ① to all other vertices. for the given below mentioned directed graph using Dijikstra's alg



Soln construct cost matrix.

C	1	2	3	4	5
1	0	4	2	∞	8
2	∞	0	∞	4	5
3	∞	∞	0	1	∞
4	∞	∞	∞	0	3
5	∞	∞	∞	∞	0

Step 1: $S = \{1\}$ & $V = \{1, 2, 3, 4, 5\}$.
let vertex ① be the source vertex.

Step 2: Initialise D.

D[2] = C[1, 2] = 4
D[3] = C[1, 3] = 2
D[4] = C[1, 4] = ∞
D[5] = C[1, 5] = 8

Step 3: find w and add it to S.

= Min { D[2], D[3], D[4], D[5] }

Add w to S.

$S = \{1,3\}$, $V = V - S = \{1,2,3,4,5\} - \{1,3\}$

$V = \{2,4,5\}$.

Now, find.

$D[v] = \min\{D[v], D[w] + C[w,v]\}$

$\therefore D[2] = \min\{D[2], D[3] + C[3,2]\}$

$= \min\{4, 2+\infty\}$

$= 4$

$D[4] = \min\{D[4], D[3] + C[3,4]\}$.

$= \min\{\infty, 2+1\}$

$= 3$

$D[5] = \min\{D[5], D[3] + C[3,5]\}$

$= \min\{8, 2+\infty\}$

$= 8$

Step 4: from the above step D[4] is minimum vertex.

$\therefore w = 4$.

Now $S = \{1,3,4\}$

$V = V - S = \{1,2,3,4,5\} - \{1,3,4\}$.

$V = \{2,5\}$.

Again find the shortest path.

$D[v] = \min\{D[v], D[w] + C[w,v]\}$

$D[2] = \min\{D[2], D[4] + C[4,2]\}$

$= \min\{4, 3+\infty\}$

$= 4$

$D[5] = \min\{D[5], D[4] + C[4,5]\}$

$= \min\{\cancel{D[5]}\,8, 3+3\}$

$= 6$

Here $w = 2$ and add it to S.

Steps: $S = \{1, 3, 4, 2\}$

$V = V - S = \{1, 2, 3, 4, 5\} - \{1, 3, 4, 2\}$

$V = \{5\}$.

Again find shortest path to V

$D[V] = \text{Min}\{D[V], D[w] + c[w, v]\}$

$D[5] = \text{Min}\{D[5], D[2] + c[2, 5]\}$

$= \text{Min}\{6, 4 + 5\}$

$= 6$

Here $w = 5$ and add it to S.

$S = \{1, 2, 3, 4, 5\}$.

$V = \{\}$.

Step6: from the source vertex 1 to all other vertices.

$D[2] = 4 \rightarrow$ least path from ① - ②

$D[3] = 2 \rightarrow$ least path from ① — ③

$D[4] = 3 \rightarrow$ least path from ① — ④

$D[5] = 6 \rightarrow$ least path from ① - ⑤