26/12/18

UNIT-III

GREEDY METHOD

* The General Method * Knapsack problem.
* Job sequencing with deadlines.
* Minimum cost spanning trees.
* Prims and Kruskals algorithms.
* An optimal Randomized Algorithm.
* Optimal merge patterns
* Single source shortest path.

* The Greedy method is one of the approach for solving a problem.
* Greedy method is also known as optimization (or) minimization problem.
* Optimization problem is nothing but which requires either a minimum result (or) maximum result.
* Optimization problem gives a solution that is called optimal solution (or) minimal solution.
* In the greedy method we can find out number of solutions, from these solutions we can select only feasible solutions.
* The greedy method is also known as selection procedure problem.

General Method (or) Abstraction Method:-

Greedy Method (or) Abstraction Method:-

```
Algorithm Greedy(a,n)
{
  solution:=0;
  for i:=1 to n do
  {
    x:= solution(a);
    if feasible(solution, x) then
      solution:= Union(Solution, x);
    }
  return solution;
}
```

*Knapsack problem:-

* Knapsack is nothing but BAG.

* Knapsack problem is also known as container loading problem

* Knapsack problem is also known as fractional knapsack problem.

*0 (zero) represents item is not considerable 1 represents item is considerable.

* The knapsack problem follows maximized $\sum p_i x_i$ subject to $\sum w_i x_i \leq m$

maximized $\sum_{1 \leq i \leq n} p_i x_i$ subject to $\sum_{1 \leq i \leq n} w_i x_i \leq m$

* Here i represents $1 \leq i \leq n$

x_i means $0 \leq x_i \leq 1$

eg: find out optimal solution for the knapsack $m=15$ and $n=7$

profits : 10, 5, 15, 7, 6, 18, 3.

weights: 2, 3, 5, 7, 1, 4, 1

→ Given that no. of objects $n = 7$

→ For each item profits are $P_i = 10\ \ 5\ \ 15\ \ 7\ \ 6\ \ 18\ \ 3$

→ For each object weights are $w_1=2\ \ w_2=3\ \ w_3=5\ \ w_4=7\ \ w_5=1$

$w_6=4\ \ , w_7=1$

→ consider profit by weight ratio method $\frac{P_i}{w_i}$.

$$\frac{P_i}{w_i} = \frac{P_1}{w_1}, \frac{P_2}{w_2}, \frac{P_3}{w_3}, \frac{P_4}{w_4}, \frac{P_5}{w_5}, \frac{P_6}{w_6}, \frac{P_7}{w_7}$$

$$\frac{P_1}{w_1} = \frac{10}{2} = 5 \qquad \frac{P_2}{w_2} = \frac{5}{3} = 1.6 \qquad \frac{P_3}{w_3} = \frac{15}{5} = 3$$

$$\frac{P_4}{w_4} = \frac{7}{7} = 1 \qquad \frac{P_5}{w_5} = \frac{6}{1} = 6 \qquad \frac{P_6}{w_6} = \frac{18}{4} = 4.5 \qquad \frac{P_7}{w_7} = \frac{3}{1} = 3$$

objects O_i :	1	2	3	4	5	6	7
profits P_i :	10	5	15	7	6	18	3
weights w_i :	2	3	5	7	1	4	1
profits / weights $\left(\frac{P_i}{w_i}\right)$:	5	1.6	3	1	6	4.5	3

Now solutions are $x_1, x_2, x_3, x_4, x_5, x_6$ & x_7

x_i is in between 0 to 1

i.e $0 \le x_i \le 1$

the bag capacity $m = 15$

According to $\frac{\text{profit}}{\text{weight}}$ ratio method we consider maximum element 1st

Now adding to the knapsack

Hence $x_5 = 1$, the remaining bag capacity is

Now add 2 kgs to the knapsack

hence $x_1 = 1$

Now add object 6 4 kgs to the knapsack.

hence $x_6 = 1$.

Now add object 3 5kgs to the knapsack

hence $x_3 = 1$

Now add object 7 1kg to the knapsack

hence $x_7 = 1$ $\quad 15 - 13 = 2$.

Now add object 2 3kg to the knapsack $\frac{2}{3}$

hence $x_2 = 2/3$

We can not add object 4 because bag contains 15 kgs.

hence $x_4 = 0$.

$$\sum_{1 \le i \le n} w_i x_i = w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4 + w_5 x_5 + w_6 x_6 + w_7 x_7$$

$$= 2 \times 1 + 3 \times \frac{2}{3} + 5 \times 1 + 7 \times 0 + 1 \times 1 + 4 \times 1 + 1 \times 1$$

$$= 2 + 2 + 5 + 0 + 1 + 4 + 1$$

$$= 15$$

$\sum w_i x_i \le m$

$15 \le 15$.

$$\sum p_i x_i = p_1 x_1 + p_2 x_2 + p_3 x_3 + p_4 x_4 + p_5 x_5 + p_6 x_6 + p_7 x_7$$

$$= 10 \times 1 + 5 \times \frac{2}{3} + 15 \times 1 + 7 \times 0 + 6 \times 1 + 18 \times 1 + 3 \times 1$$

$$= 10 + 3.33 + 15 + 0 + 6 + 18 + 3$$

$$= 55.3$$

hence the optimal solution is $(x_1, x_2, x_3, x_4, x_5, x_6, x_7) =$ $(1, \frac{2}{3}, 1, 0, 1, 1, 1)$ and the maximum profit is 55.3

eg: Find maximum optimal solution for $n=3$ & $m=20$?

profits $P_1, P_2, P_3 = 25, 24, 15$ weights $w_1, w_2, w_3 = 18, 15, 10$

Given that no of objects $n=3$

For each item profits are $P_1=25$; $P_2=24$; $P_3=15$.

For each object weights are $w_1=18$; $w_2=15$; $w_3=10$.

consider $\frac{\text{profit}}{\text{weight}}$ ratio $\frac{P_i}{w_i}$

$$\frac{P_i}{w_i} = \frac{P_1}{w_1}, \frac{P_2}{w_2}, \frac{P_3}{w_3}$$

$$\frac{P_1}{w_1} = \frac{25}{18} = 1.38$$

$$\frac{P_2}{w_2} = \frac{24}{15} = 1.6$$

$$\frac{P_3}{w_3} = \frac{15}{10} = 1.5$$

objects O_i:	1	2	3
profits P_i:	25	24	15
weights w_i:	18	15	10
$\frac{\text{profits}}{\text{weights}}$ $\left(\frac{P_i}{w_i}\right)$:	1.38	1.6	1.5

Now solutions are x_1, x_2, x_3.

x_i is in between 0 to 1

i.e $0 \le x_i \le 1$

The bag capacity $m=20$

According to $\frac{\text{Profit}}{\text{weight}}$ ratio method we consider maximum element 1st

Now add 1 to the knapsacks

Hence $x_2 = 1$ the remaining bag capacity is

Now add object 3 to the knapsacks

Hence $x_3 = \frac{5}{10} = \frac{1}{2}$

we can not add object 1 because bag contains 20kg

Hence $x_1 = 0$

$$\sum w_i x_i = w_1x_1 + w_2x_2 + w_3x_3$$
$$= 0 + 15 \times 1 + 10 \times \frac{1}{2}$$
$$= 0 + 15 + 5$$
$$= 20$$

$$\sum p_i x_i = p_1x_1 + p_2x_2 + p_3x_3$$
$$= 25 \times 0 + 24 \times 1 + 15 \times 0.5$$
$$= 0 + 24 + 7.5$$
$$= 31.5$$

Hence the optimal solution is $(x_1, x_2, x_3) = (0, 1, 1/2)$ and the maximum profit is 31.5

(or)

x_i ① $x_1\ x_2\ x_3 \Rightarrow 0 \le x_i \le 1$

$\sum p_i x_i$ ② $\Rightarrow p_1x_1 + p_2x_2 + p_3x_3$

Weights $\sum w_i x_i$ ③ $\Rightarrow w_1x_1 + w_2x_2 + w_3x_3 \le m$

$= 18 \times 1 + 15 \times 0 + 10 \times \frac{1}{2} \le 20$

$= 18 + 0 + 5 \le 20$

$= 20$

$\Rightarrow 18 \times 0 + 15 \times 1 + 10 \times \frac{1}{2} \le 20$

$= 0 + 15 + 5 \le 20$

$= 20.$

$\Rightarrow 18 \times 1 + 15 \times \frac{2}{15} + 10 \times 0$

$= 18 + 2 \le 20$

$= 20.$

	x_1	x_2	x_3
①	1	0	1/5
②	0	1	1/2
③	1	2/15	0

Profits

① $25 \times 1 + 24 \times 0 + 15 \times \frac{1}{5}$

$= 25 + 0 + 3 = 28$

② $25 \times 0 + 24 \times 1 + 15 \times \frac{1}{2}$

$24 + 7.5 = 31.5$

③ $25 \times 1 + 24 \times \frac{2}{15} + 15 \times 0$

$2.5 + \frac{16}{15} \times 0$

$= 25 + 3.2 = 28.2$

2-1-19

Q) Find optimal solution for the following knapsack problem

$n=7$, $m=15$ profits $(P_1, P_2, P_3, P_4, P_5, P_6, P_7) = (10, 5, 15, 7, 6, 18, 3)$

weights $(w_1, w_2, w_3, w_4, w_5, w_6, w_7) = (4, 3, 6, 6, 2, 5, 1)$

Sol: Given that no. of objects $n=7$

for each item profits are $P_1 = 10$, $P_2 = 5$, $P_3 = 15$, $P_4 = 7$, $P_5 = 6$, $P_6 = 18$, $P_7 = 3$

For each object weights are $w_1 = 4$, $w_2 = 3$, $w_3 = 6$, $w_4 = 6$, $w_5 = 2$, $w_6 = 5$, $w_7 = 1$.

Consider $\frac{\text{profit}}{\text{weight}}$ ratio $\frac{P_i}{w_i}$

$$\frac{P_i}{w_i} = \frac{P_1}{w_1}, \frac{P_2}{w_2}, \frac{P_3}{w_3}, \frac{P_4}{w_4}, \frac{P_5}{w_5}, \frac{P_6}{w_6}, \frac{P_7}{w_7}.$$

$\frac{P_1}{w_1} = \frac{10}{4} = 2.5$

$\frac{P_2}{w_2} = \frac{5}{3} = 1.6$

$\frac{P_3}{w_3} = \frac{15}{6} = 2.5$

$\frac{P_4}{w_4} = \frac{7}{6} = 1.1$

$\frac{P_5}{w_5} = \frac{6}{2} = 3$

$\frac{P_6}{w_6} = \frac{18}{5} = 3.6$

$\frac{P_7}{w_7} = \frac{3}{1} = 3$

objects O_i:	1	2	3	4	5	6	7
profits P_i:	10	5	15	7	6	18	3
weights w_i:	4	3	6	6	2	5	1
$(\frac{P_i}{w_i})$:	2.5	1.6	2.5	1.1	3	3.6	3

Now solutions are $x_1, x_2, x_3, x_4, x_5, x_6, x_7$.

x_i is in between 0 to 1.

i.e $0 \le x_i \le 1$

The bag capacity $m = 15$

According to Profit/Weight ratio method we consider max element 1st

Now add object6 5kgs to the knapsacks, hence $x_6 = 1$

Now add object5 2kgs to the knapsacks, hence $x_5 = 1$

Now add object7 1kg to the knapsacks, hence $x_7 = 1$

Now add object1 4kg to the knapsacks

Hence $x_1 = 1$

Now add object3 6kg to the knapsacks.

Hence $x_3 = \frac{1}{2}$

We can not add object2 because bag contains 15kgs

Hence $x_2 = 0$.

We can not add object4 because bag contain 15kgs

Hence $x_4 = 0$

$$\sum w_i x_i = w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + w_5x_5 + w_6x_6 + w_7x_7$$

$$= 4\times1 + 3\times0 + \overset{3}{\cancel{6}}\times\frac{1}{\cancel{2}} + 6\times0 + 2\times1 + 5\times1 + 1\times1$$

$$= 4 + 3 + 2 + 5 + 1$$

$$= 15$$

$$\sum w_i x_i \le m$$

$$15 \le 15$$

$$\sum P_i x_i = P_1x_1 + P_2x_2 + P_3x_3 + P_4x_4 + P_5x_5 + P_6x_6 + P_7x_7$$

$$= 10\times1 + 5\times0 + \overset{7.5}{15}\times\frac{1}{2} + 7\times0 + 6\times1 + 18\times1 + 3\times1$$

$$= 10 + 0 + 7.5 + 0 + 6 + 18 + 3$$

$$= 44.5$$

Hence the optimal soln is $(x_1, x_2, x_3, x_4, x_5, x_6, x_7) = (1, 0, \frac{1}{2}, 0, 1, 1, 1)$ and the maximum profit is 44.5

KNAPSACK ALGORITHM:-

```
Algorithm GreedyKnapsack(m, n)
{
 for i:=1 to n do
 x[i]:=0.0;
 U:=m;
 for i:=1 to n do
 {
  if (W[i]>U) then break;
  x[i]:=1.0;
  U:=U-W[i];
 }
 if (i<=n) then
   x[i]:=U/W[i];
}
```

NOTE:- The time complexity for the Greedy Knapsack is O(n).

Job Sequencing with Deadlines:-

* Consider that there are 'n' jobs that are to be executed
* At any time T = 1, 2, 3.... only exactly one job is to be executed.
* Each job takes 1 unit of time
* If job starts before (or) as its deadline profit is obtain otherwise no profit.
* Consider all possible schedules and compute the minimum total time in the system.
* Goal is to schedule jobs to maximize the total profit.

Ex:- What is the job sequencing with deadlines let n=5, profits $(P_1, P_2, P_3, P_4, P_5) = (20, 13, 10, 4, 1)$ and deadlines $(D_1, D_2, D_3, D_4, D_5) = (2, 1, 2, 3, 3)$

Sol:- Given that no.of jobs n = 5

profits are $P_1 = 20$, $P_2 = 13$,

$P_2 = 13$

$P_3 = 10$

$P_4 = 4$

$P_5 = 1$

corresponding Deadlines are $D_1 = 2$

$D_2 = 1$

$D_3 = 2$

$D_4 = 3$

$D_5 = 3$

The No.of possible feasible solutions are,

Job sequence	allowed or Not	profit
1	allowed	20
2	Allowed	13
3	allowed	10
4	allowed	4
5	allowed	1

Job sequence	allowed (or) NOT	profit
(1-2)	not allowed	
(1-3)	allowed	20+10 = 30
(1-4)	allowed	20+4 = 24
(1-5)	allowed	20+1 = 21
(2-1)	allowed	13+20 = 33
(2-3)	allowed	13+10 = 23
(2-4)	allowed	13+4 = 17
(2-5)	allowed	13+1 = 14
(3-1)	allowed	10+20 = 30
(3-2)	not allowed	
(3-4)	allowed	10+4 = 14
(3-5)	allowed	10+1 = 11
(4-1)	not allowed	
(4-2)	not allowed	
(4-3)	not allowed	
(4-5)	allowed	4+1 = 5
(5-1)	not	
(5-2)	not	
(5-3)	not	
(5-4)	allowed	1+4 = 5
(1-2-3)	not allowed	
(2-1-4)	allowed	13+20+4 = 37
(2-3-4)	allowed	13+10+4 = 27
(2-1-5)	allowed	13+20+1 = 34
(2-3-5)	allowed	13+10+1 = 24

P	Deadlines	
20	1	2
13	2	1
10	3	2
	4	3
4	5	3
1		

Hence the optimal solution is (2,1,4) and the maximum profit is 37.

Find an optimal solution and maximum profit for the following Greedy Job sequencing with Deadlines let n=4, profits $(P_1, P_2, P_3, P_4) = (100, 10, 5, 27)$ Deadlines $(D_1, D_2, D_3, D_4) = (2,1,2,1)$

Given that no.of jobs n=4

profits are $P_1 = 100$

$P_2 = 10$

$P_3 = 5$

$P_4 = 27$

Corresponding deadlines are $D_1 = 2$

$D_2 = 1$

$D_3 = 2$

$D_4 = 1$

The no.of possible feasible solutions are.

Job Sequence	allowed (or) Not	Profit
1	allowed	100
2	allowed	10
3	allowed	5
4	allowed	27
(1-2)	not allowed	
(1-3)	allowed	100+5 =105
(1-4)	not allowed	
(2-1)	allowed	10+100=110
(2-3)	allowed	10+5=15
(2-4)	allowed	10+27=37
(3-1)	allowed	5+100 =105
(3-2)	not allowed	
(3-4)	not allowed	
(4-1)	allowed	27+100 = 127
(4-2)	allowed	27+10 = 37
(4-3)	allowed	27+5 = 32

	Deadlines	Profits
1	2	100
2	1	10
3	2	5
4	1	27

Hence the optimal solution is (4,1) and the maximum profit is 127.

Minimum cost Spanning Tree:-

Spanning Tree:-

* A spanning tree of a graph $G=(V,E)$ is a subgraph of G that is a tree and contains all the vertices of G certain no circuit

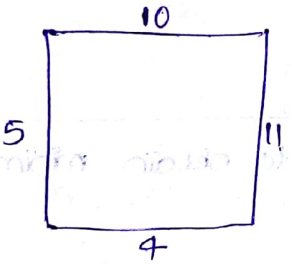
* An edge of spanning tree is called a branch

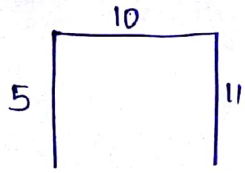
* An edge in the graph that is not in the spanning tree is called a chord

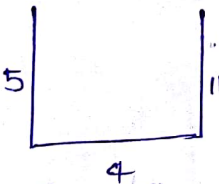
* Spanning trees are represented by using two graph searching algorithms

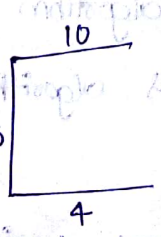
1. Breadth First Search (BFS)
2. Depth First Search (DFS)

eg:-











cost = 5+10+11 = 26

cost = 5+4+11 = 20

cost = 5+10+4 = 19

cost = 10+11+4 = 25

Minimum cost spanning trees:-

* If a graph $G=(V,E)$ is weighted graph then which contains least weight among all spanning trees. is called Minimum spanning tree.



Algorithm:-

```
Algorithm MST(G, W, T)
{
  T[] = 0;
  Mindist = 0;
  put the n nodes in T and no edges;
  While T has lessthan n-1 edges do
  {
  choose a remaining edge e of minimum weight;
  Assign the minimum weight to mindist;
  Delete e from the graph;
  if(e does not create a cycle in T)then
  Add e to T;
  }
  return mindist;
}
```

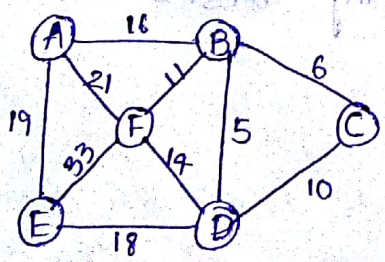
These are two important algorithms to obtain minimum spanning tree. They are 1. prims algorithm

2. kruskals algorithm.

Prims Algorithm:-

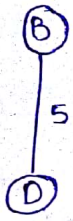
* In this method least weight edge is selected.
* Adjacent minimum weight edge is selected.
* In this way the procedure is continue until all the vertices are connected without forming cycles.

Eg:-

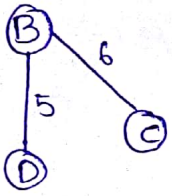


Step:1 Initially the total weight is '0'.

Step:2 Identify least weight edge. Now the total weight is $0+5=5$

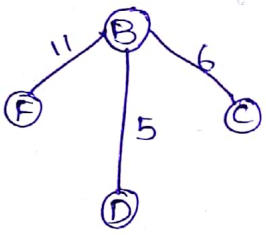


Step:3 Identify next least weight edge.



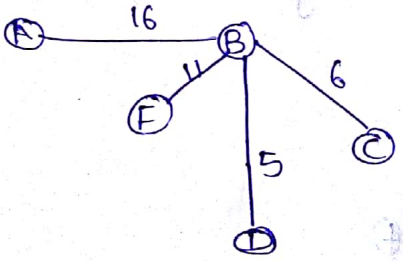
Now the total weight is $0+5+6=11$

Step:4 Identify next least weighted edge



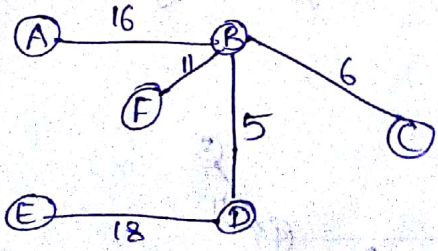
Now the total cost is $5+6+11=22$

Step:5 Identify next least weight edge



Now the total cost is $5+6+11+16=38$

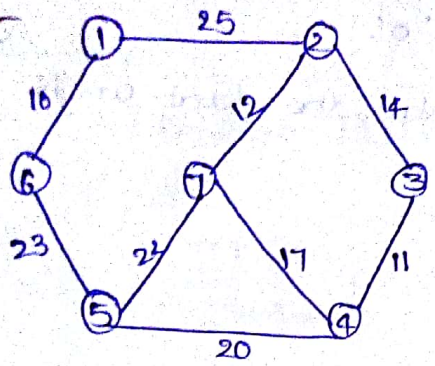
Step:6 Identify next least weight edge



Now the total cost is $5+6+11+16+18=56$

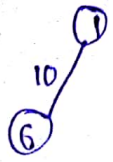
The total minimum cost is 56.

Eg:-



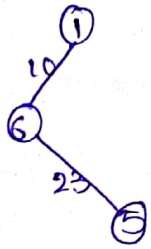
Step:1 Initially the total weight is '0'

Step:2 Identify least weight edge.



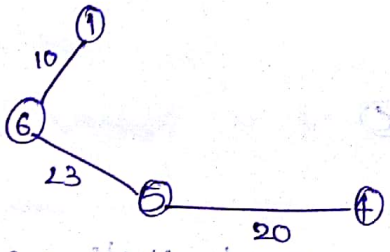
Now the total weight is 0+10 = 10

Step:3 Identify least weight edge



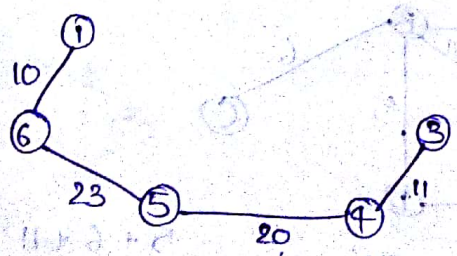
Now the total weight is 10+23 = 33

Step:4 Identify least weight edge.



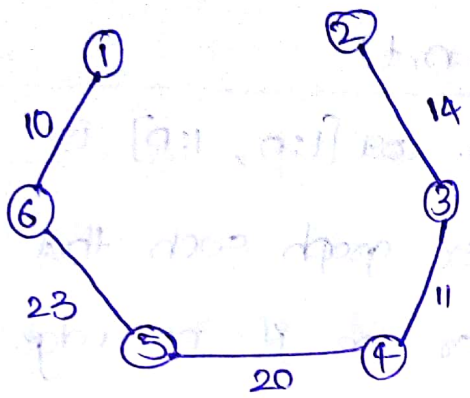
Now the total weight is 10+23+20 = 53.

Step:5 Identify least weight edge.



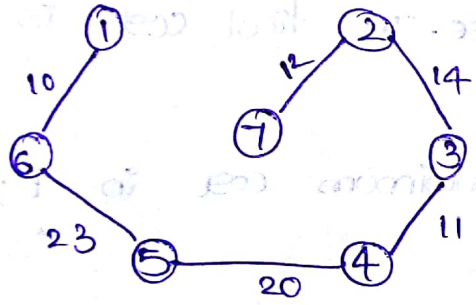
Now the total weight is 10+23+20+11 = 64

Step:6 Identify next least weight edge.



Now the total weight is $10+23+20+11+14=78$

step:7 Identify the least weight edge



Now the total weight is $10+23+20+11+14+12=90$

Prims Algorithm:- ~~prim (E, cost, n, t)~~

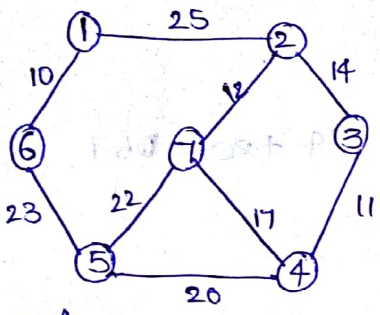
```
//E is the set of edges in G. cost[1:n, 1:n] is the cost
//Adjacency matrix of an n vertex graph such that cost[i,j] is
//either a positive real number or ∞ if no edge (i,j) exists
//A minimum spanning tree is computed and stored as a set of
//edges in the array t[1:n-1, 1:2]. (t[i,1], t[i,2]) is an edge in
//the minimum cost spanning tree. The final cost is returned.
Algorithm Prim(E, cost, n, t)
{
  Let (k,l) be an edge of minimum cost in E;
      mincost := cost[k,l];
      t[1,1] := k; t[1,2] := l;
    for i=1 to n do //initialize near
      if (cost[i,1] < cost[i,k]) then near[i] := l;
        else near[i] := k;
      near[k] := near[l] := 0;
      for i := 2 to n-1 do
      { //find n-2 additional edges for t
      let j be an index such that near[j] ≠ 0 and
      cost[j, near[j]] is minimum;
       t[i,1] := j; t[i,2] := near[j];
      mincost := mincost + cost[j, near[j]];
        near[j] := 0;
       for k := 1 to n do //update near[].
     if ((near[k] ≠ 0) and (cost[k, near[k]] > cost[k,j]))
        then near[k] := j;
      }
    return mincost;
}
```

Note:- The time complexity of greedy prims algorithm is $O(n^2)$

Kruskals Algorithm:-

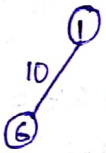
* In a kruskals algorithm always the minimum cost edge has to be selected.
* It is not necessary that selected optimum edge is adjacent
* In this way the procedure is continued untill all vertices are connected without forming cycles.

eg:-



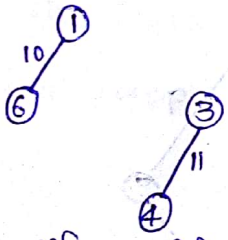
Step:1 Initial the total weight is "zero".

Step:2 Identify minimum cost edge from the graph.



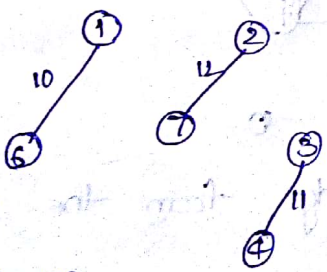
Total weight 0+10 = 10.

Step:3 Identify minimum edge cost.



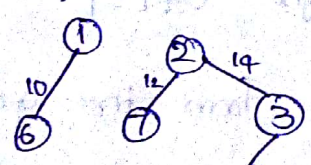
Total weight = 0 +10+11
= 21

Step:4 Identify minimum edge cost



Total weight = 0+ 10+11 +12 = 33

Step:5 Identify minimum edge cost



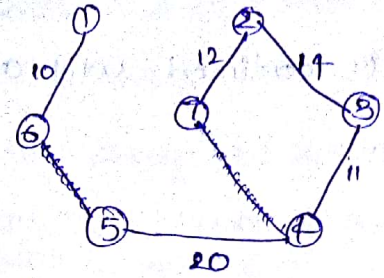
Total weight = 0+10+11+12+14
= 47

Step:6 Identify minimum edge cost



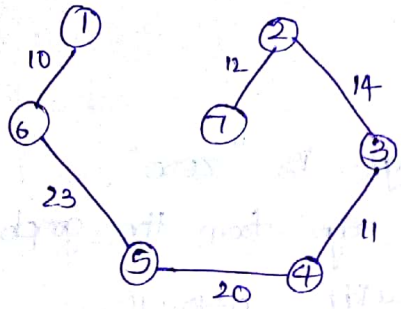
* In this step, the next minimum edge cost makes from 7 to 4. It makes a closed cycle. Hence not selected.

Step:7 Identify next minimum edge.



Total weight 0+10+11+12+14+20 = 67

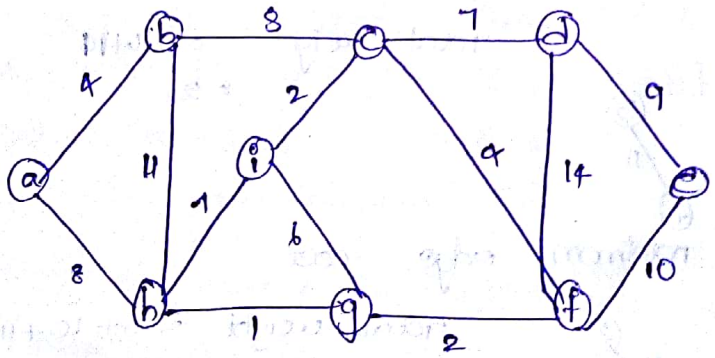
Step:8



Total weight 0+10+11+12+14+20+23 = 90

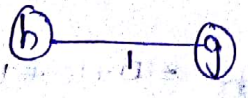
The minimum spanning tree is 90.

eg:-



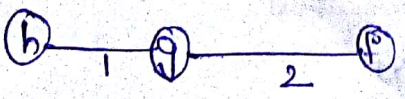
Step1: The min Initial weight is 0

Step:2 Identify minimum cost edge from the graph



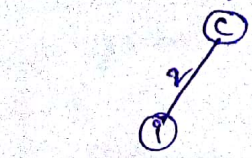
Total weight 0+1 = 1

Step:3 Identify minimum cost edge from the graph



Total weight 0+1+2 = 3

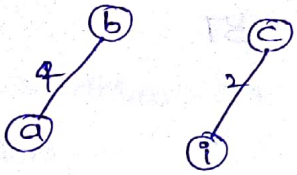
Step:4 Identify minimum cost edge from the graph.



Total weight = 0+1+2+2 = 5



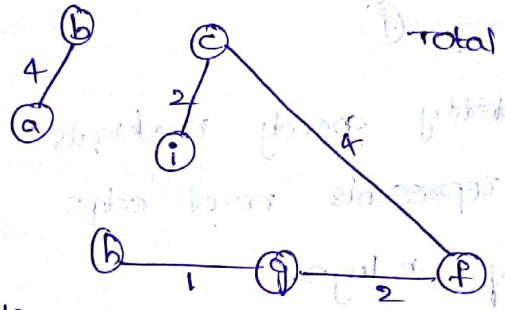
Step:5 Identify minimum cost edge from the graph.



Total weight = 0+1+2+2+4+4 = 9



Step:6 Identify minimum cost edge from the graph

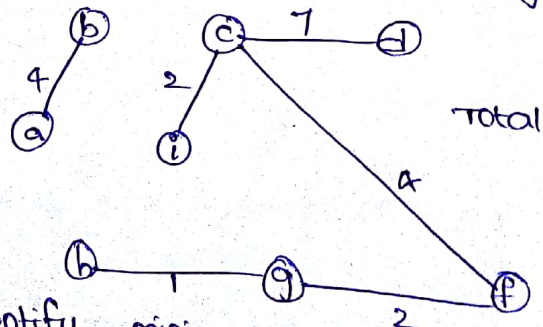


Total weight = 0+1+2+2+4+4 = 13

Step:7 In this step, the next minimum edge cost makes from i to g It makes a closed cycle. Hence Not selected.

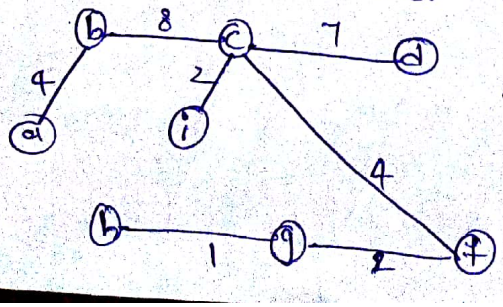
Step:8 In this step, the next minimum edge cost makes from i to h It makes a closed cycle. Hence not selected.

Step:9 Identify minimum cost edge from the graph



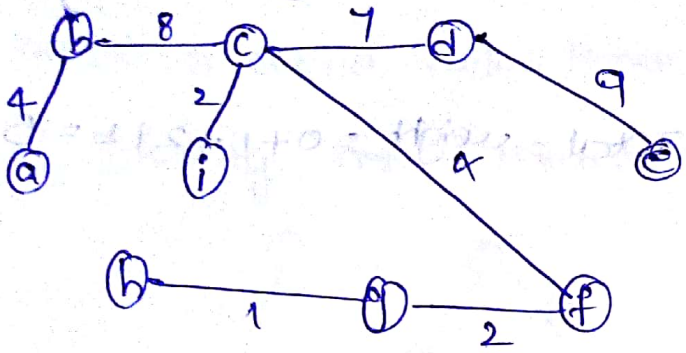
Total weight = 0+1+2+2+4+4+7 = 20

Step:10 Identify minimum cost edge from the graph



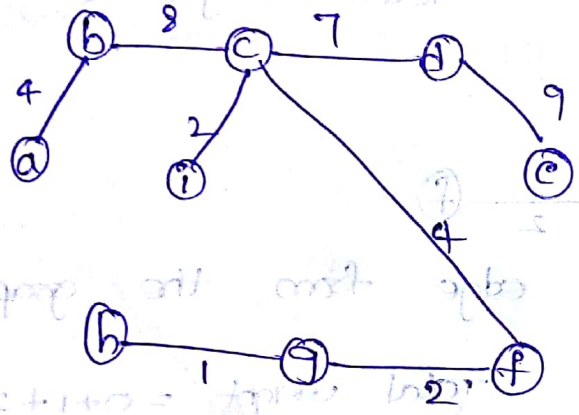
Total weight = 0+1+2+2+4+4+7+8 = 28

Step:11. Identify the minimum cost edge from the graph.



Total weight = 0+1+2+2+4+4+7+8+9 = 37

The minimum spanning tree is 37.



Note:- The time complexity greedy kruskals algorithm is O(Elogn) where E represents no. of edges

The time complexity nlogn.

Algorithm:-

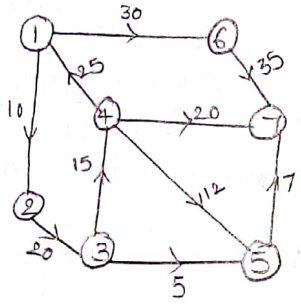
```
Algorithm Kruskal(E, cost, n, t)
{
 construct a heap out of the edge costs using Heapify;
 for i:=1 to n do parent[i]:=-1;
  i:=0; mincost:=0.0;
 While ((i<n-1) and (heap not empty)) do
 {
 Delete a minimum cost edge (u,v) from the heap and reheapify
using Adjust;
    j:= Find(u); k:= Find(v);
    if (j≠k) then
    {
      i:= i+1;
      t[i,1]:=u; t[i,2]:=v;
    mincost:=mincost + cost[u,v];
     Union(j,k);
    }}
    if (i≠n-1) then write("No spanning tree");
    else return mincost;
}
```

8/1/19

SINGLE SOURCE SHORTEST PATH PROBLEM (Dijkstra's Algorithm):

* In this problem the given graph is a weighted and directed graph.
* Dijkstra's Algorithm is used to represent the distance b/w two cities (or) any two edges.
* In single source shortest path problem the shortest distance from a single vertex is called source, and the last vertex is called Destination.
* Finally, we can written the shortest path and minimum distance

eg:-1



Consider source vertex S(1) = 1

Now select source vertex is 1

From source vertex we can find possible distances the distance between

$D\{1,2\} = 10$

$D\{1,3\} = \infty$

$D\{1,4\} = \infty$

$D\{1,5\} = \infty$

$D\{1,6\} = 30$

$D\{1,7\} = \infty$

Now consider the optimal path is $\{1 \to 2\}$ and minimum distance is 10.

Now select S(2) = 2

$D\{1,2,3\} = 10+20 = 30$

$D\{1,2,4\} = 10+\infty = \infty$

$D\{1,2,5\} = \infty$

$D\{1,2,6\} = \infty$

$D\{1,2,7\} = \infty$

Now shortest path is {1→2→3} and its distance is 30.

Now select S(3) = 3.

D{1, 2, 3, 4} = 10 + 20 + 15 = 45

D{1, 2, 3, 5} = 10 + 20 + 5 = 35

D{1, 2, 3, 6} = ∞

D{1, 2, 3, 7} = ∞.

Now shortest path is {1→2→3→5} and its distance is 35.

Now select S(5) = 5

D{1, 2, 3, 5, 4} = ∞

D{1, 2, 3, 5, 6} = ∞

D{1, 2, 3, 5, 7} = 10 + 20 + 5 + 7 = 42

Hence shortest path is {1→2→3→5→7} and its distance is 42

Hence the single source shortest path is

$\therefore 1 \xrightarrow{10} 2 \xrightarrow{20} 3 \xrightarrow{5} 5 \xrightarrow{7} 7$

and the minimum distance is 42.

Single source shortest path from each vertex is summarized below.

1 → 2 = 10

1 → 6 = 30

1 → 2 → 3 = 10 + 20 = 30

1 → 6 → 7 = 30 + 35 = 65

1 → 2 → 3 → 4 = 10 + 20 + 15 = 45

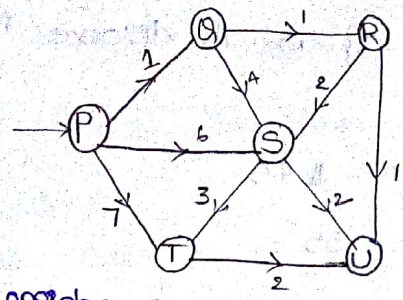
1 → 2 → 3 → 5 = 10 + 20 + 5 = 35

1 → 2 → 3 → 4 → 5 = 10 + 20 + 15 + 12 = 57

1 → 2 → 3 → 4 → 7 = 10 + 20 + 15 + 20 = 65

1 → 2 → 3 → 5 → 7 = 10 + 20 + 5 + 7 = 42.

eg:2



Consider source vertex $S(p) = p$

Now select source vertex is P.

From source vertex we can find possible distances the distance between

$D\{P,Q\} = 1$

$D\{P,R\} = \infty$

$D\{P,S\} = 6$

$D\{P,T\} = 7$

$D\{P,U\} = \infty$

Now consider the optimal path $\{P \to Q\}$ and minimum distance is 1

Now select $S(Q) = Q$

$D\{P,Q,R\} = 1+1 = 2$

$D\{P,Q,S\} = 1+4 = 5$

$D\{P,Q,T\} = \infty$

$D\{P,Q,U\} = \infty$

Now shortest path is $\{P \to Q \to R\}$ and distance is 2.

Now select $S(R) = R$

$D\{P,Q,R,S\} = 1+1+2 = 4$

$D\{P,Q,R,T\} = \infty$

$D\{P,Q,R,U\} = 1+1+1 = 3.$

Hence shortest path is $\{P \to Q \to R \to U\}$ and distance is 3

Hence single source shortest path is

$P \xrightarrow{1} Q \xrightarrow{1} R \xrightarrow{1} U$ and minimum distance is 3.

Single source shortest path from each vertex is summarized below.

P → Q = 1

P → T = 7

P → S = 6

P → Q → R = 1+1 = 2

P → Q → S = 5

P → T → U = 7+2 = 9

P → Q → R → S = 1+1+2 = 4

P → Q → S → T = 1+4+3 = 8

P → S → U = 6+2 = 8

P → S → T = 6+3 = 9

P → Q → S → U = 1+4+2 = 7

P → Q → R → U = 1+1+1 = 3

Algorithm for shortest path:-

```
Algorithm ShortestPaths (v, cost, dist, n)
{
  for i:= 1 to n do
  {
    S[i]:= false; dist[i]:=cost[v,i];
  }
  S[v]:= true; dist[v]:=0.0;
  for num:= 2 to n do
  {
  choose u from among those vertices not in S such
that dist[u] is minimum;
    S[u]:=true;
  for(each w adjacent to u with S[w]:= false) do
    if (dist[w] >(dist[u] + cost[u,w])) then
        dist[w]:= dist[u] + cost[u,w];
  }
}
```

10/1/19

Note:-

Time complexity for Greedy single source shortest path is $O(n^2)$

OPTIMAL MERGE PATTERNS:-

* In merging already the given list of elements are in sorted order.

eg:- list A contains 4 elements 3 5 7 9

list B contains 4 elements 2 4 8 11

Now merge A and B

A	B	C
3	2	2
5	4	3
7	8	4
9	11	5
		7
		8
		9
		10.

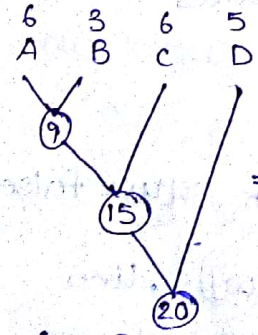
Now

* the size of C list is 4+4 = 8

* The In two-way merging merge two lists for each and every time

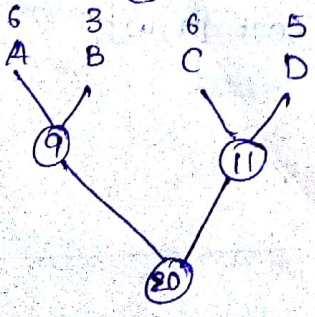
eg:- the given lists are A, B, C, D and its corresponding sizes are 6, 3, 6, 5

(1)

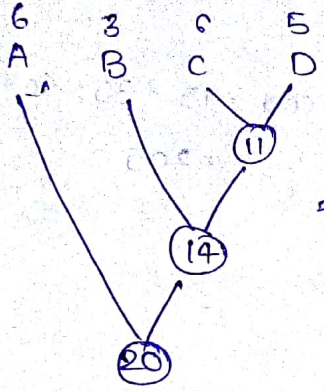


$\Rightarrow 9+15+20 = 44$

(2)



$\Rightarrow 9+11+20 = 40$



$\Rightarrow 11+14+20$

$= 45$

* Hence the minimum cost optimal solution is 40.

* To generate optimal solution the greedy Method Introduce the dynamic problem is called "optimal merge patterns".

Eg:- To find total cost using optimal merge pattern problem for the following lists: x_1 x_2 x_3 x_4 x_5

20 30 10 5 30

Given lists : x_1 x_2 x_3 x_4 x_5

20 30 10 5 30

⇒ Arrange the above sizes are in increasing order

5 10 20 30 30

⇒ Apply merging (Adding) 5 and 10

(15) 20 30 30

⇒ Apply merging (Adding) 15 and 20

(35) 30 30

⇒ Arrange the above sizes are in increasing order

30 30 35

⇒ Apply merging in 30 30

(60) 35

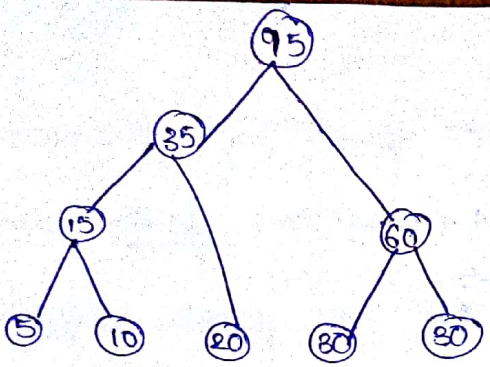
⇒ Arrange the above sizes are in increasing order

35 60

⇒ Apply merging in 35 60

(95)

total cost is $15+35+60+95 = 205$



total cost = 15 + 35 + 60 + 95
= 205

$\sum_{i=1}^{n} d_i x_i$

$= 3\times5 + 3\times10 + 2\times20 + 2\times30 + 2\times30$

$= 15 + 30 + 40 + 60 + 60$

$= 205$

Hence optimum (or) minimum

Eg: Find the optimal solution for the following sizes

28, 32, 12, 5, 84, 53, 91, 35, 3, 11

Given lists

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
28	32	12	5	84	53	91	35	3	11

Arrange the above sizes are in ascending order (or) increasing order

3 5 11 12 28 32 35 53 84 91

Apply merging 3 and 5

(8) 11 12 28 32 35 53 84 91

Apply merging 8 and 11

(19) 12 28 32 35 53 84 91

Arrange the above sizes are in increasing order

12 19 28 32 35 53 84 91

Apply merging in 12 and 19

(31) 28 32 35 53 84 91

Arrange the above sizes are in increasing order

28 31 32 35 53 84 91

Apply merging 28 and 31

(59) 32 35 53 84 91

Arrange the above sizes are in increasing order

32 35 53 59 84 91

Apply merging 32 and 35

(67) 53 59 84 91

Arrange the above sizes are in increasing order

53 59 67 84 91

Apply merging 53 and 59

(112) 67 84 91

Arrange the above sizes are in increasing order

67 84 91 112

Apply merging 67 and 84

(151) 91 112

Arrange the above sizes are in increasing order

91 112 151

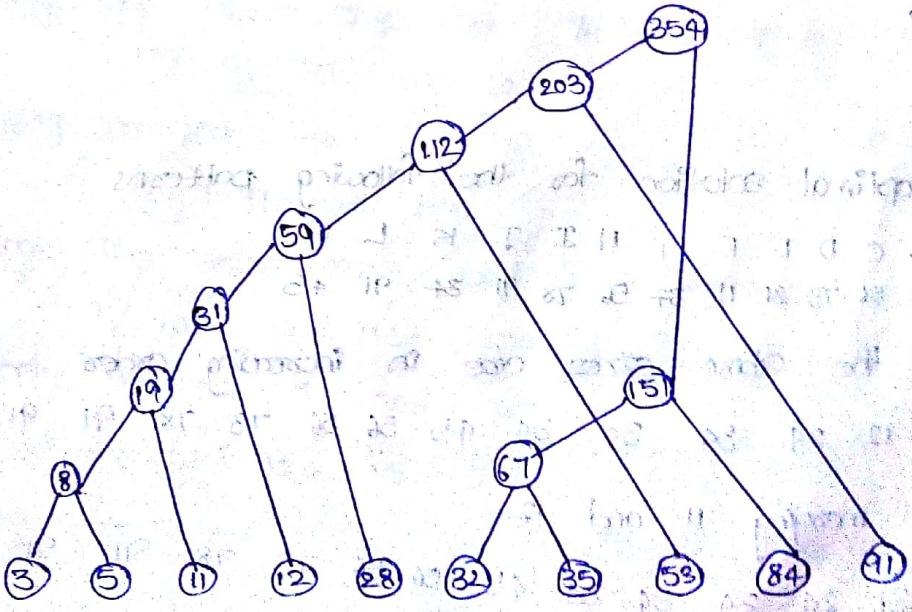
Apply merging 91 and 112

(203) 151

Arrange the above sizes are in increasing order

(354)

Total cost is $8+19+31+59+67+112+151+203+354 = 1004$



Total cost $= 8+19+31+59+67+112+151+203+354$

$= 1004$

$\sum_{i=1}^{n} d_i x_i = 3\times7+5\times7+11\times6+12\times5+28\times4+32\times3+35\times3+53\times3+84\times2+91\times2$

$= 1004$

Algorithm for Optimal Merge patterns:

```
treenode = record {
treenode *lchild ; treenode *rchild;
integer weight;
};
Algorithm Tree(n)
//list is a global list of n single single node
//binary trees as described above
{
for i:= 1 to n-1 do
{
pt:= new treenode ; // Get a new tree node
(pt->lchild) := least (list); //merge two trees with
(pt -> rchild) := least (list); //smallest lengths
(pt -> weight) := (pt -> lchild) -> weight) + ((pt -> rchild)
                                              -> weight);
Insert (list, pt);
}
return least(list); //tree left in list is the merge tree
}
```

*Mid

Find optimal solution for the following patterns

A	B	C	D	E	F	G	H	I	J	K	L	M
12	34	56	73	24	11	34	56	78	91	34	91	45

Arrange the above sizes one in increasing order

11 12 24 34 34 34 45 56 56 73 78 91 91

Apply merging 11 and 12

(23) 24 34 34 34 45 56 56 73 78 91 91

Apply merging 23 and 24

(47) 34 34 34 45 56 56 73 78 91 91

Arrange the above sizes one in ascending order

34 34 34 45 47 56 56 73 78 91 91

Apply merging 34 and 34

(68) 34 45 47 56 56 73 78 91 91

Arrange the above sizes are in increasing order

34 45 47 56 56 68 73 78 91 91

Apply merging 34 and 45

(79) 47 56 56 68 73 78 91 91

Arrange the above sizes are in increasing order

47 56 56 68 73 78 79 91 91

Apply merging 47 and 56

(103) 56 68 73 78 79 91 91

Arrange the above sizes are in increasing order

56 68 73 78 79 91 91 103

Apply merging 56 and 68

(124) 73 78 79 91 91 103

Arrange the above sizes are in increasing order

73 78 79 91 91 103 124.

Apply merging 73 and 78

(151) 79 91 91 103 124

Arrange the above sizes are in increasing order

79 91 91 103 124 151

Apply merging 79 and 91

(170) 91 103 124 151

Arrange the above sizes are in increasing order

91 103 124 151 170

Apply merging 91 and 103

(194) 124 151 170

Arrange the above sizes are in increasing order

124 151 170 194

Apply merging 124 and 151

(275) 170 194

Arrange the ranges are in increasing order

170 194 275

Apply merging 170 and 194

(364) 275

Arrange the above sizes are in increasing order:

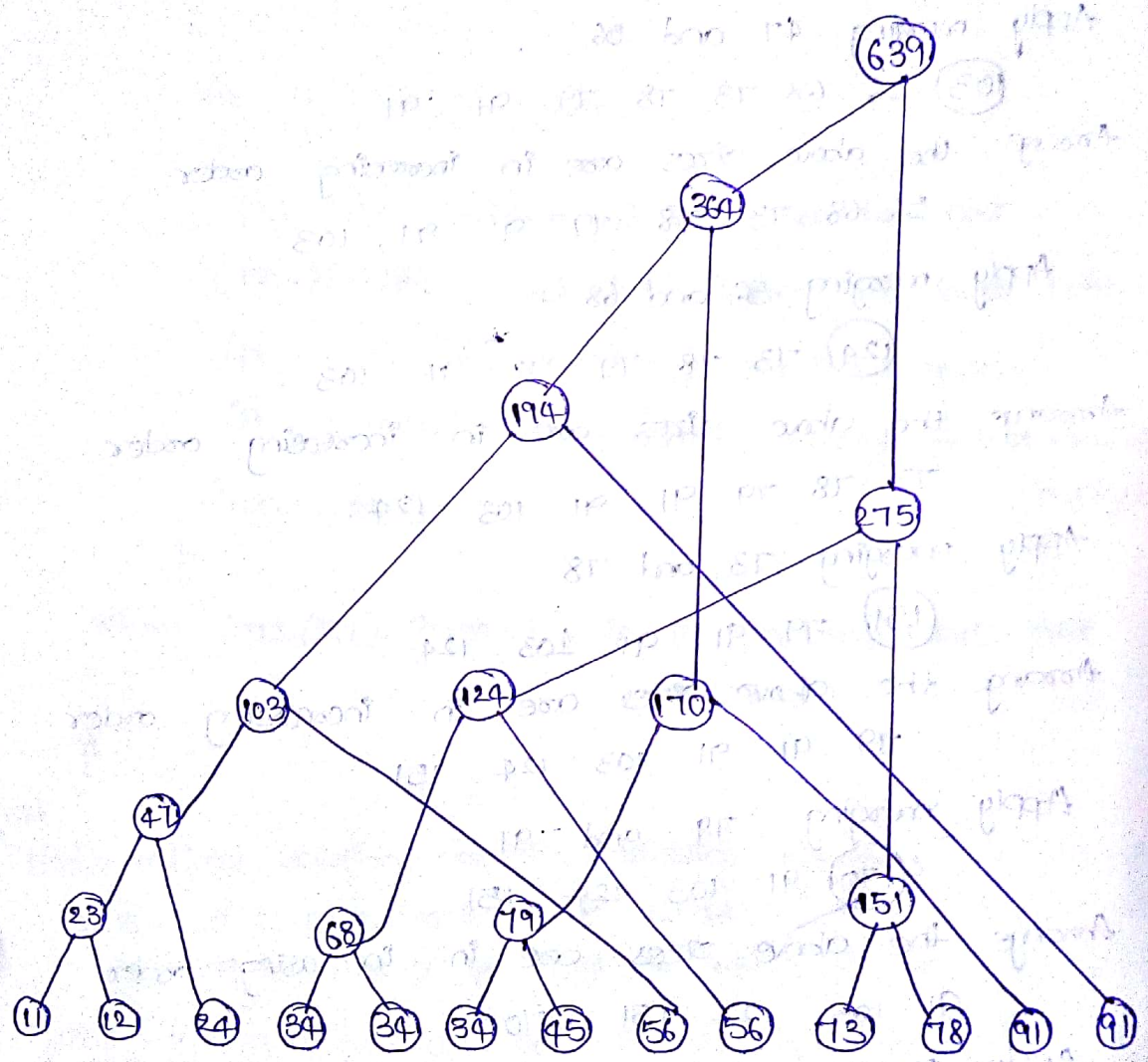
275 364

Apply merging 275 and 364

(639)

Total cost = 23+47+68+79+103+124+151+170+194+275+364+639.

= 2237



Total cost = 23 + 47 + 68 + 79 + 103 + 124 + 151 + 170 + 194 + 275 + 364 + 639

= 2237

$\sum_{i=1}^{n} d_i x_i$ = 11x6 + 12x6 + 24x5 + 34x4 + 34x4 + 34x4 + 45x4 + 56x4 + 56x3 + 73x3 + 78x3 + 91x3 + 91x3

= 2237

Note:-

The time complexity for Greedy optimal merge pattern is $O(n^2)$.

Differences between Greedy Method and Divide and conquer.

Greedy Method	Divide and Conquer.
* Greedy method does not give best solution always. Eg:- Job sequence with deadlines and Knapsack problem. * Greedy algorithm neither postpone nor recursive decisions. * A solution is made up of a no. of steps * In Greedy method we can find out possible no. of feasible solutions and obtain optimal solutions.	* But the divide and conquer gives the best optimal solution only Eg:- quicksort * Divide and conquer algorithms merge the results of the very same algorithm applied to subset of the data. * Breaks a problem into smaller sub problems. and finally merge the solutions and build return. * In Divide & conquer method the given problem is very small then return solution directly, otherwise the problem can be divided into possible no. of sub problems finally return single solution.