

Chap 2 The greedy Method

Second
2.2

Explain general greedy method

The general greedy method can be applied to variety of problems, most problem contains inputs & requires the constraints. Any subset that satisfies these constraints is called feasible solution. The feasible soln is used to either maximizes or minimizes the objective function. A feasible solution that does this is called an optimal solution. It considering one input at a time.

The algorithm for the greedy method is as follow, It is a top down approach.

1. Algorithm Greedy(a,n)
2. // a[1:n] contains the n inputs.
3. {
4. solution := \emptyset ; // Initialize the solution
5. for i=1 to n do
6. {
7. $x := \text{select}(a)$;
8. if feasible(solution, x) then
9. solution := union(solution, x);
10. }
11. return solution;

12. }
The void select so not to lost a

method is at : select(x)

Explain knapsack problem & job sequencing with deadline in details.

Knapsack problems are solved by using the general greedy method. In knapsack problem we have given n objects & a knapsack or bag. Where object i having the weight w_i & the bag having the capacity of m . Objective of this method is to find out maximum profit with the capacity of that bag.

If a fraction x_i ; $0 \leq x_i \leq 1$ then profit is calculated by $p_i x_i$; the objective is to find maximum of total profit.

According to all above data the knapsack problem can be solved as,

maximum $\sum p_i x_i$ are subject to

$i \in \{1, 2, \dots, n\}$

$\sum w_i x_i \leq m$ &

$0 \leq x_i \leq 1$ $1 \leq i \leq n$

Where weight & profit are positive numbers.

A feasible solution or filling is any set (x_1, x_2, \dots, x_n) are satisfy above equation,

example: In a problem

$$n=3$$

$$m=20$$

$$(P_1, P_2, P_3) = (25, 24, 15) \rightarrow P_i$$

$$(w_1, w_2, w_3) = (10, 18, 15) \rightarrow w_i$$

We have to find feasible solution i.e. $(x_1, x_2, x_3) \in \Theta$

$$1) \frac{10}{10}, \frac{10}{18}, \frac{0}{15} = (1, \frac{5}{9}, 0)$$

$$2) \frac{10}{10}, 0, \frac{10}{15} = (1, 0, \frac{2}{3})$$

$$3) 0, \frac{18}{18}, \frac{2}{15} = (0, 1, \frac{2}{15})$$

$$4) \frac{2}{10}, \frac{18}{18}, \frac{0}{15} = (\frac{1}{5}, 1, 0)$$

$$5) 0, \frac{5}{18}, \frac{15}{15} \text{ cannot be feasible}$$

$$6) \frac{5}{10}, 0, \frac{15}{15} = (\frac{1}{2}, 0, 1)$$

Now according to above filling we find the weight & profit as follow.

$$\sum p_i x_i$$

$$\sum w_i x_i$$

$$① 25x1 + 24x\frac{5}{9} = 25 + \frac{120}{9} \\ = 38.33$$

$$① 1x10 + 0x18 + 15x\frac{2}{3} \\ = 20$$

$$② 25x1 + 24x0 + \frac{2}{3}x15 \\ = 35.00$$

$$② 1x10 + 0x18 + \frac{2}{3}x15 \\ = 20$$

$$③ 25x0 + 1x24 + \frac{2}{15}x15 \\ = 28.00$$

$$③ 1x10 + \frac{5}{9}x18 + 0x15 \\ = 20$$

$$④ \frac{2}{5}x25 + 24x1 = 34.00$$

$$④ \frac{1}{5}x10 + 1x18 + 0x15 \\ = 20$$

$$\textcircled{5} \quad \frac{5}{18} \times 24 + 1 \times 15 = \underline{\underline{21.33}} \quad \text{initial solution} \quad 0 \times 10 + \frac{5}{18} \times 18 + 1 \times 15 = \underline{\underline{20}}$$

$$\textcircled{6} \quad 12 \cdot 5 + 15 = \underline{\underline{27.5}} \quad (\text{check}) \quad \frac{1}{2} \times 10 + 0 \times 18 + 1 \times 15 = \underline{\underline{20}}$$

From all over data the 38.33 is optimal solution hence it is maximum profit.

Algorithm for greedy strategies for the knapsack problem

Algorithm Greedy knapsack (m, n)

// $p[1:n]$ and $w[1:n]$ contains the profit &

// weight respectively of n objects ordered

// such that $p[i]/w[i] \geq p[i+1]/w[i+1]$

// m is the knapsack size &

~~triples~~ // $x[1:n]$ is the solution vector.

{

for $i := 1$ to n do $x[i] := 0.0$; // initialize x ;

$v := m$

- for $i := 1$ to n do

~~except~~ { if $p[i] > v$ then $x[i] := 1.0$ }

IF ($w[i] > v$) then break;

$x[i] := 1.0$; $v := v - w[i]$;

} for $i := 1$ to n do

~~else if~~ ($i < n$) then $x[i] := v / w[i]$;

}

$21x_0 + 27x_1 + 15x_2$ ③

$21x_0 + 27x_1 + 15x_2$ ④

~~0.0~~

~~0.0~~

$21x_0 + 27x_1 + 15x_2$ ⑤

$0.048 = 1x_0 + 29x_2$ ⑥

~~0.0~~

Job sequencing with deadlines

We are given a set of n jobs. Associated with job i is an integer deadline $d_i \geq 0$ & profit $p_i \geq 0$. For any job i the profit p_i is earned iff the job is completed by its deadlines.

To complete a job, one has to process the job on a machine for one time (unit time). Only one machine is available for processing jobs. The feasible solution for this problem is a subset J of jobs such that each job in this subset can be completed by its deadlines. The value of feasible solution J is the sum of the profit of the job in J or $\sum_{i \in J} p_i$. An optimal solution is a feasible solution with maximum value.

Algorithm:

Algorithm JS(d, j, n) as per sd + pd work

* $d[i] \geq 1$; $1 \leq i \leq n$ are the deadlines, $n \geq 1$

The jobs are ordered such that $p[1] > p[2] \dots > p[n]$. $J[i]$ is the i th job in optimal solution. $1 \leq i \leq k$.

Also at termination $d[j[i]] \leq d[j[i+1]]$, $1 \leq i \leq k$ /*

```
d[0] = j[0] = 0 // Initialize
J[1] = 1; // Include job 1
k=1;
for i=2 to n do
    &
    // Consider jobs in nondecreasing order of
    p[i] find
```

// Position for i & check feasibility of insertion

$\sigma := k$;

while (($d[j[\sigma]] > d[i]$) & ($d[J[\sigma]] \neq \sigma$))

```

do  $\sigma = \sigma - 1$ ;
if ( $(d[J[\sigma]] \leq d[i]) \& (d[i] > \sigma)$ ) then
    // Insert i into J[ ]
    for  $q=k$  to  $(\sigma+1)$  step-1 do  $J[q+1] := J[q]$ ;
     $J[\sigma+1]; i; k=k+1$ ;
}

```

return k ;

Example:

$$n=4 \text{ // No of jobs}$$

$$(P_1, P_2, P_3, P_4) = (100, 10, 15, 27) \text{ // Profit}$$

$$(d_1, d_2, d_3, d_4) = (2, 1, 2, 1) \text{ // Deadlines}$$

Now by the given formula,

No	feasible solution	Processing sequence	Value
1	$(1, 2)$	$2, 1$	170
2	$(1, 3)$	$(1, 3) \Rightarrow (3, 1)$	115
3	$(1, 4)$	$(4, 1)$	127
4	$(2, 3)$	$2, 3$	25
5	$(3, 4)$	$(4, 3)$	42
6	(1)	(1)	100
7	(2)	(2)	10
8	(3)	(3)	15
9	(4)	(4)	27

not feasible to P if $d_i \geq d_j$ for $i < j$ and $P_i > P_j$

$((i < j) \& (P_i > P_j))$ or $(d_i \leq d_j \& P_i < P_j)$

In above only jobs 1, 4 are processes & hence the value is 127. Which is the optimal solution.

These jobs must be processed in the ordered job 4 followed by job 1. Thus the processing time of job 4 is zero & that of job 1 is completed at time 2.

Job sequencing with deadlines

Finding maximum profit for given set of profit & deadlines.

Designing a multiprogramming operating system.

① Example:

Solve the problem by job sequencing with deadlines.

No of objects = 4

Profit = (100, 10, 15, 27)

Deadline = (2, 1, 2, 1)

According to profit arrange the objects.

object	Profit	Deadline
--------	--------	----------

01	100	2
----	-----	---

04	27	1
----	----	---

03	15	2
----	----	---

02	10	1
----	----	---

No of jobs to be selected = Maximum deadline.

We have to select = 2 jobs.

job 04 & 02 having same deadlines select 04 with maximum profit = 27

from 01 & 03 select 01 = 100

Total profit = 127, optimal solution (1, 001)

Q2) Solve job sequence with deadline problem using greedy approach for, $n=5$

$$(P_1, P_2, \dots, P_5) = (45, 15, 20, 7, 65)$$

$$(d_1, d_2, \dots, d_5) = (1, 3, 2, 1, 2)$$

object Profit Deadline

O1 45 1

O2 15 3

O3 20 2

O4 7 1

O5 65 2

If we arrange given problem with profit:

O5 65 2

O1 45 1

O3 20 2

O2 15 3

O4 7 1

As we given maximum deadline as 3.

We have to select 3 object.

1) One from deadline 1

2) One from deadline 2

3) One from deadline 3

from deadline 1 = 45 (O1) feasible or not

from deadline 2 = 65 (O5) feasible or not

from deadline 3 = 15 (O2) feasible or not

$\frac{1}{1} + \frac{1}{3} + \frac{1}{2} = 1.25$ feasible or not

Profit = 125, feasible, optimal soln = (1, 1, 0, 0, 1)

find the maximum profit such a way that knapsack problem.

$$n=3, m=25$$

$$(P_1, P_2, P_3) = (10, 20, 80)$$

$$(w_1, w_2, w_3) = (12, 13, 9)$$

Now we have know,

$$m=25, n=3 \text{ maximum weight,}$$

so feasible solution will be,

$$1) (12, 13, 0)$$

$$2) (12, 4, 9)$$

$$3) (3, 13, 9)$$

$$4) (12, 13, 0)$$

$$5) (12, 4, 9)$$

$$6) (3, 13, 9)$$

Now feasible solution will be,

$$(1, 1, 0), (1, 4, 13), (1, 4, 1, 1)$$

Now we know that,

$$\text{Profit} = \sum P_i x_i \text{ with } i \in \{1, 2, 3\}$$

$$\text{Weight} = \sum w_i x_i$$

$$1) 10x1 + 20x1 + 80x0$$

$$1x12 + 1x13 + 0x93$$

$$= 30$$

$$= 25$$

$$2) 10x1 + 20x4/13 + 80x1$$

$$1x12 + 4/13x13 + 1x93$$

$$= 10 + 80 + 6.1$$

$$= 25$$

$$= 46.1$$

(Optimal)

$$3) 10x1/4 + 20x1 + 80x1$$

$$12x1/4 + 13x1 + 0x93$$

$$2.5 + 50$$

$$= 25$$

$$= 52.5$$

Maximum profit 52.5, hence it is optimal solution

Q3) Solve the knapsack problem using $n=4$.

$$(w_1, w_2, w_3, w_4) = (10, 15, 6, 9)$$

$$(P_1, P_2, P_3, P_4) = (2, 5, 8, 1)$$

Capacity of knapsack is $m = 25$.

Object	Profit	Weight	P/w
O ₁	2	10	0.20
O ₂	5	15	0.33
O ₃	8	6	1.33
O ₄	1	9	0.11

If we arrange P/w ratio in descending.

Object	P/w
O ₃	1.33
O ₂	0.33
O ₁	0.20
O ₄	0.11

Now we know that bag containing maximum 125.

By taking profit & weight ratio into consideration

$$O_3 \text{ object} \Rightarrow \text{profit} = 8 \quad \text{weight} = 6$$

$$\text{upto } O_2 \text{ object} \Rightarrow \text{profit} = 8+5 \quad \text{weight} = 6+15 = 21$$

Now, we want 4 items to fulfill the take 4 items up, from O₁

$$8+5+0.2 \times 8 + 0.1 \times 15 = 13.8$$

Total maximum profit = 13.8

Knapsack capacity $m = 25$

optimal solution = (1/10, 1, 1, 0)

④ find optimal solution of knapsack.

$$n=6, C=20$$

$$(P_1, P_2, \dots, P_6) = (12, 5, 15, 7, 6, 18)$$

$$(W_1, W_2, \dots, W_6) = (2, 3, 5, 7, 1, 5)$$

object profit & weight w.r.t P/W

$$O_1 \quad P_1 = 12 \quad 2 \quad 6$$

$$O_2 \quad P_2 = 5 \quad 3 \quad 1.67$$

$$O_3 \quad P_3 = 15 \quad 5 \quad 3$$

$$O_4 \quad P_4 = 7 \quad 7 \quad 1$$

$$O_5 \quad P_5 = 6 \quad 1 \quad 6$$

$$O_6 \quad P_6 = 18 \quad 3.6 \quad 5$$

If we arrange object in sequence of decreasing order then,

	w	Taken Object = 20			a	$: p_j$
O_1	6	2	2		2/2 = 1	
O_5	6	1	1		1	
O_6	3.6	5	5		1	
O_3	3	5	5		1	
O_2	1.67	3	3	a	1	A
O_4	1	7	7	$7-3=4$	4/7	
					$\Sigma = 20$	

$$\text{Profit} = 6 \times 2 + 6 \times 1 + 8 \times 5 + 3 \times 5 + 1.67 \times 3 + 1 \times 4$$

$$= 12 + 6 + 18 + 15 + 5 + 4$$

$$\text{Profit} = 60$$

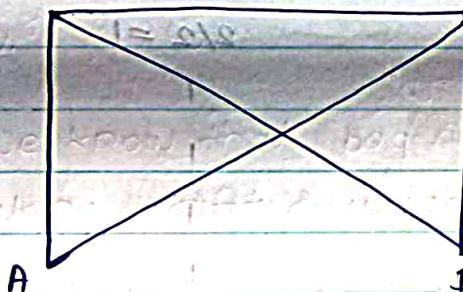
$$\text{optimal solution} = (1, 1, 1, 4/7, 1, 1)$$

grd

Minimum Cost Spanning Tree:

- 1) No. of vertices in graph equals to no. of vertices in tree.
- 2) The edges in tree should be lesser than graph.
- 3) All vertices should be connected.
- 4) No. cycle is found.
- 5) If graph contain n vertices then tree should contain atleast $(n-1)$ edges, should be undirected graph.

e.g.: B



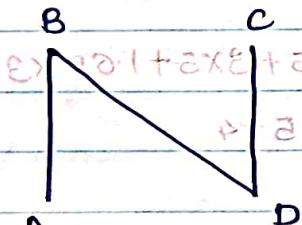
1	1	2	10
1	2	3	20
2	3	2	30
2	3	3	30
3	1	2	10

* some possible spanning tree

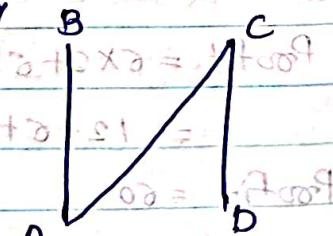
1)



2)



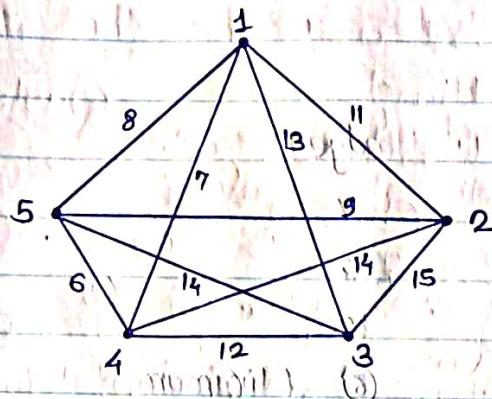
3)



2. Prim's Algorithm :

- 1) Initially node is selected & processes.
- 2) final Adjacency matrix.

e.g.: find Minimal spanning tree of graph.



Now adjacency matrix [5x5]

$$M = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 0 & 11 & 13 & 7 & 8 \\ 2 & 11 & 0 & 15 & 14 & 13 \\ 3 & 9 & 15 & 0 & 12 & 14 \\ 4 & 7 & 14 & 12 & 0 & 6 \\ 5 & 8 & 13 & 14 & 6 & 0 \end{bmatrix}$$

1) Select Node 1

path from

1 - 2

1 - 3

1 - 4

1 - 5

edges

13

9

11

8

⑦ Minimum

2) Select Node 2 :

path from

2 - 1

2 - 3

2 - 4

2 - 5

edge

⑪ Minimum

15

14

13

Select Node 3:

path from edge

3 - 1 6 ⑨ Minimum

3 - 2 15

3 - 4 12

3 - 5 14

Select Node 4:

path from edge

4 - 1 7

4 - 2 14

4 - 3 12

4 - 5 ⑥ Minimum

Select Node 5:

path from edge

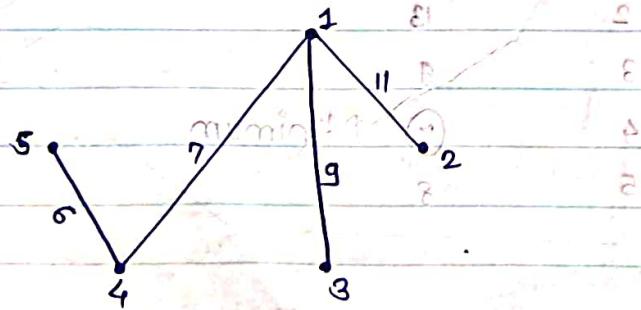
5 - 1 8 ⑧

5 - 2 13

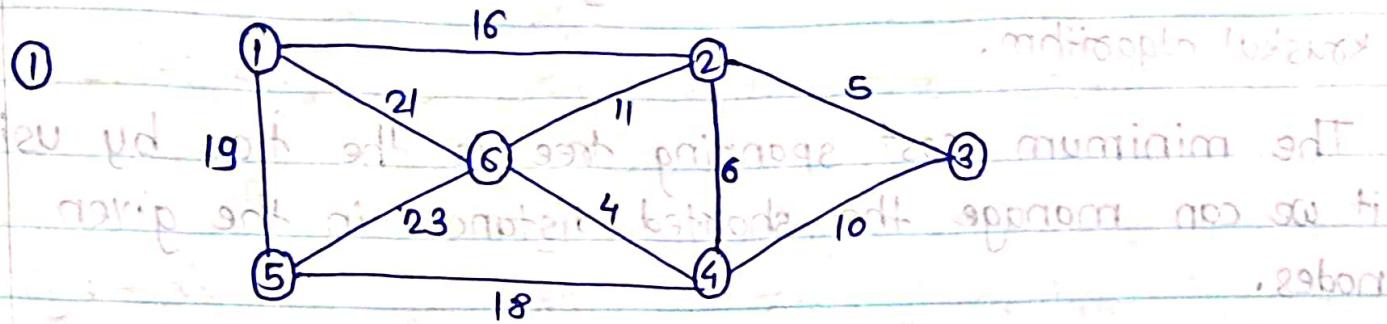
5 - 3 14

5 - 4 6 Selected

spanning Tree

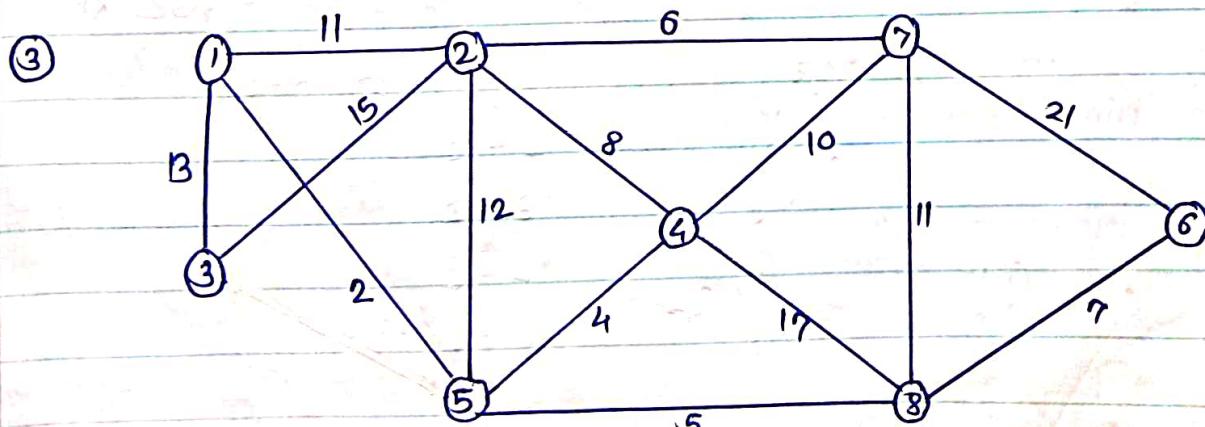
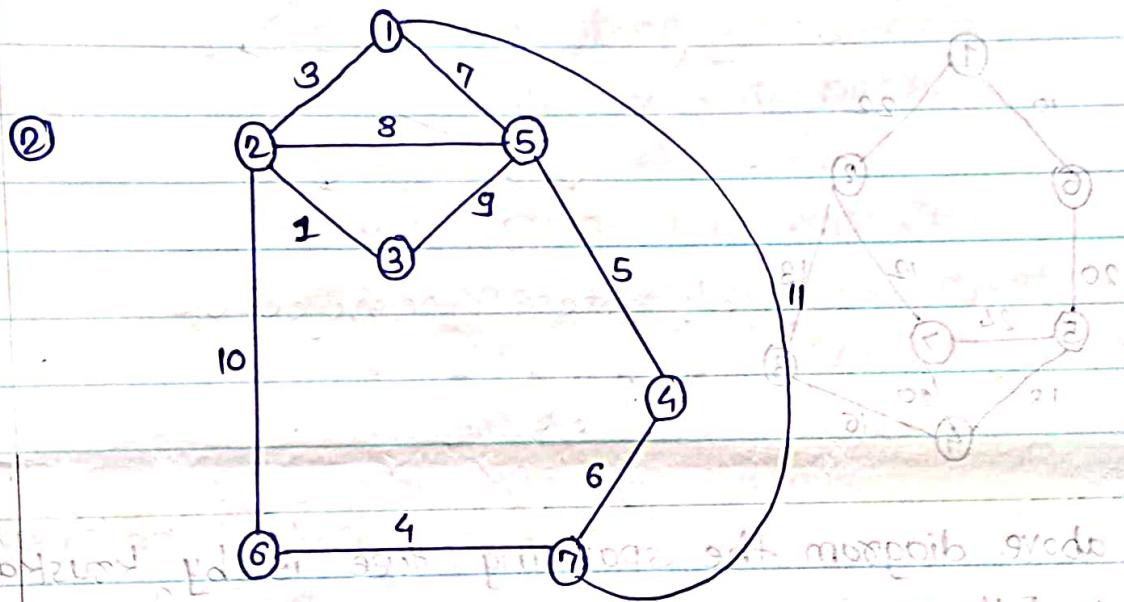


example:



Follow the algorithm to find the minimum spanning tree.

Let us consider the following tree.

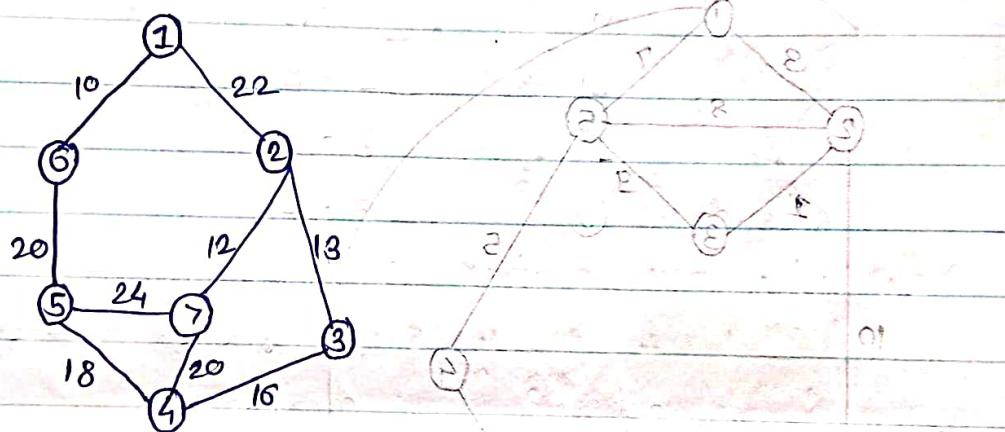


What is minimum cost spanning tree & explain with Kruksal algorithm.

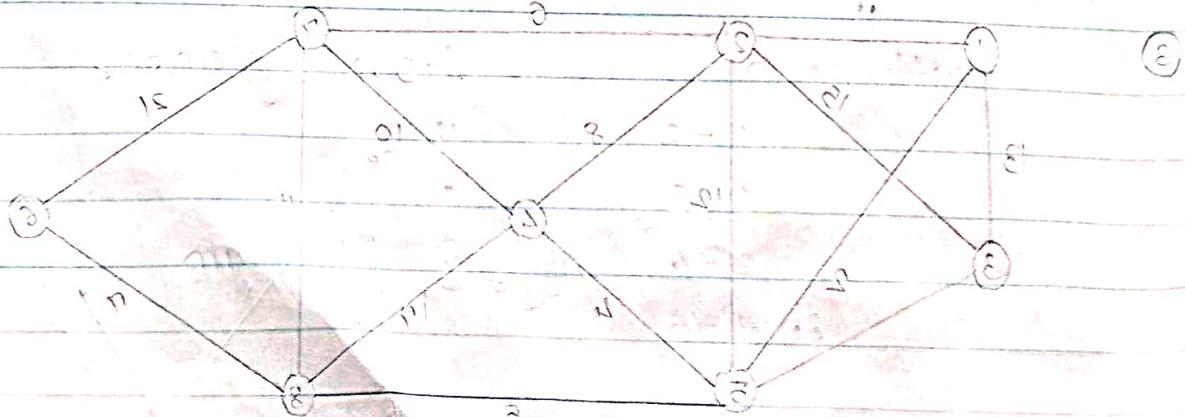
The minimum cost spanning tree is the tree by using it we can manage the shortest distance in the given nodes.

According to the kruskals algorithm the smallest edge is used for the tree.

Let us consider the following tree.



From above diagram the spanning tree is by kruskals algorithm as follows.



Algorithm for kruskals

Kruskal (E , cost, n , t)

{

construct a heap out of the edge constant
using hierarchy

~~for $i=1$ to n do $\text{parent}[i]=-1;$~~

~~// each vertex in different set~~

~~$i=0$; $\text{minwst}=0.0;$~~

~~while ($(i < (n-1)$ & (heap not empty)) do~~

{

~~Delete a minimum wt edge (u,v) & from
the heap & breaphy using adjust;~~

~~$j = \text{find}(u); k = \text{find}(v);$~~

~~if ($j \neq k$) then~~

{

~~$i = i + 1;$~~

~~$t[i,1] = u; t[i,2] = v;$~~

~~$\text{minwst} = \text{minwst} + \text{wst}[u,v];$~~

~~$\text{union}(j,k);$~~

}

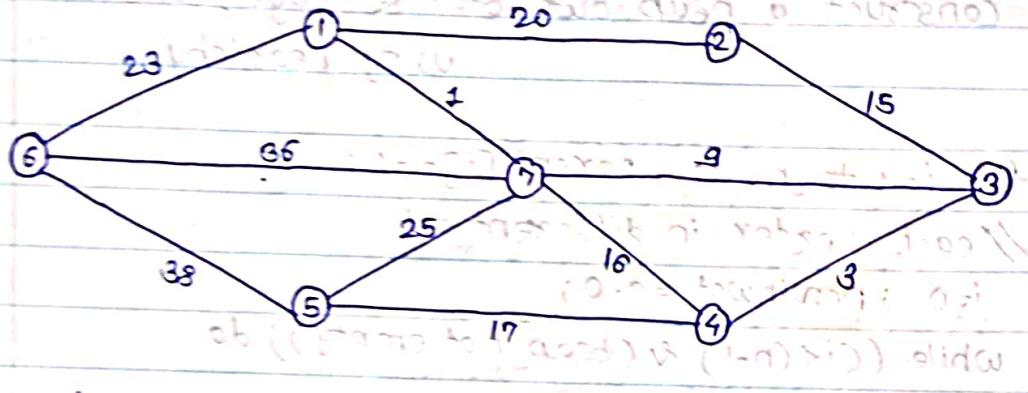
}

~~if ($i \neq n-1$) then write ("No spanning tree");
else return mincost;~~

}

Kruskal's algorithm:

finding the lightest weight & go in increasing order without cycle.

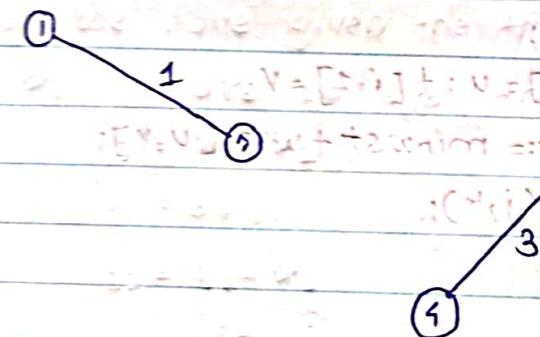


Step ①

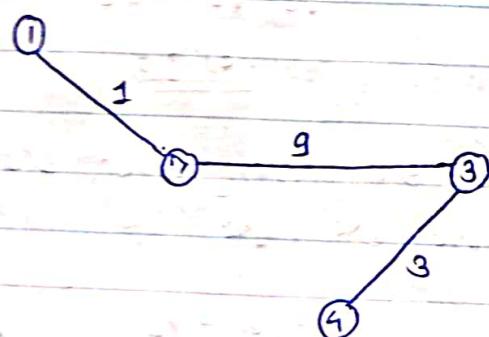
1 (one) sets to minimum & stable
at 1 edge is added & add off

7

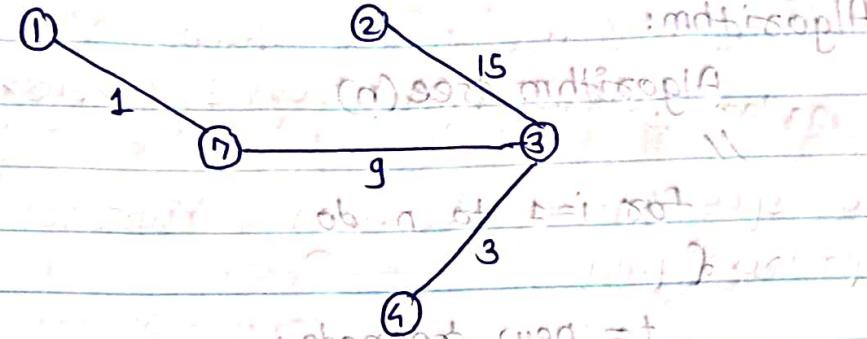
Step ②



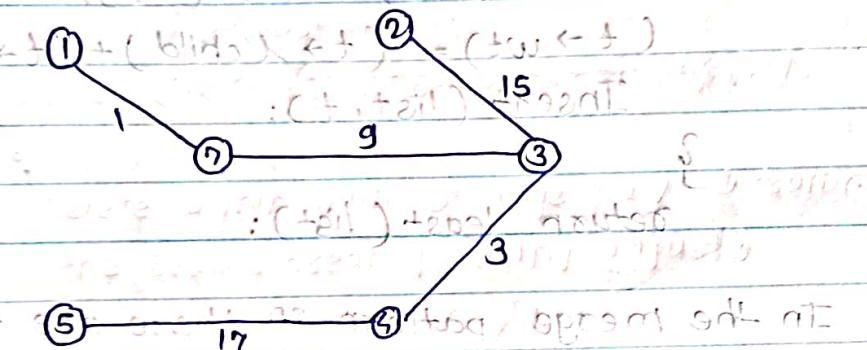
Step ③



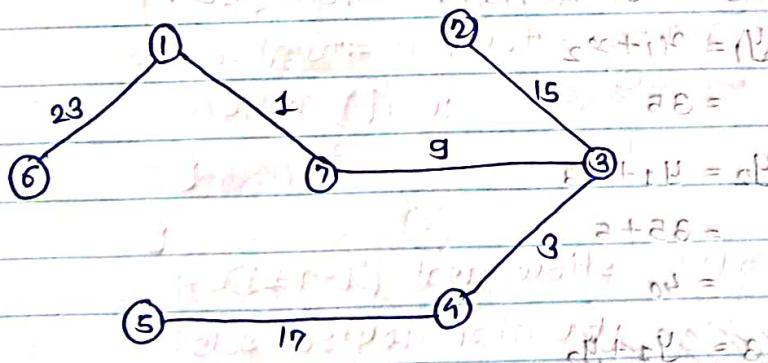
Step 4



Step 5



Now Step 6



$$\text{Minimum Cost} \Rightarrow 1 + 9 + 15 + 17 + 23$$

$$1 + 9 + 15 + 17 + 23 = 68 \text{ odd number over 90 HL}$$

$$1 + 9 + 15 = 25$$

$$15 + 17 = 32$$

$$25 + 32 = 57$$

Explain optimal merge pattern and huffmann's code.

Algorithm:

Algorithm tree(n)

11

for i=1 to n do

{

t = new treenode;

(t → l child) = least(list);

(t → r child) = least(list);

(t → wt) = (t → l child) + (t → r child) + wt);

Insert(list, t);

}

return least(list);

In the merge pattern if there are two inputs then is no any problem for sorting but if merge than 2 nodes are given then we have to merge such problem.

Let we have given the three inputs in such a way that x_1, x_2, x_3 are 20, 15, 5 then the,

$$y_1 = x_1 + x_2$$

$$= 35$$

$$y_2 = y_1 + x_3$$

$$= 35 + 5$$

$$= 40$$

$$y_3 = y_2 + y_1$$

$$= 75$$

~~$x_1 + x_2 + x_3 + y_1 + y_2 + y_3$~~ <= too many

If we have given the merge files as ,

x_1, x_2, x_3, x_4 , then

$$y_1 = x_1 + x_2$$

$$y_2 = y_1 + x_3$$

$y_2 + x_4$ = sorted files

example:

$$(x_1, x_2, x_3, x_4, x_5) = (20, 30, 10, 5, 30)$$

Now by optimal merge pattern

from above example we choose the minimum value such as,

$$y_1 = x_3 + x_4$$

$$= 10 + 5$$

$$y_2 = y_1 + x_1$$

$$= 35$$

$$\text{Similarly } y_3 = x_2 + x_5$$

$$= 30 + 30$$

$$= 60$$

Hence,

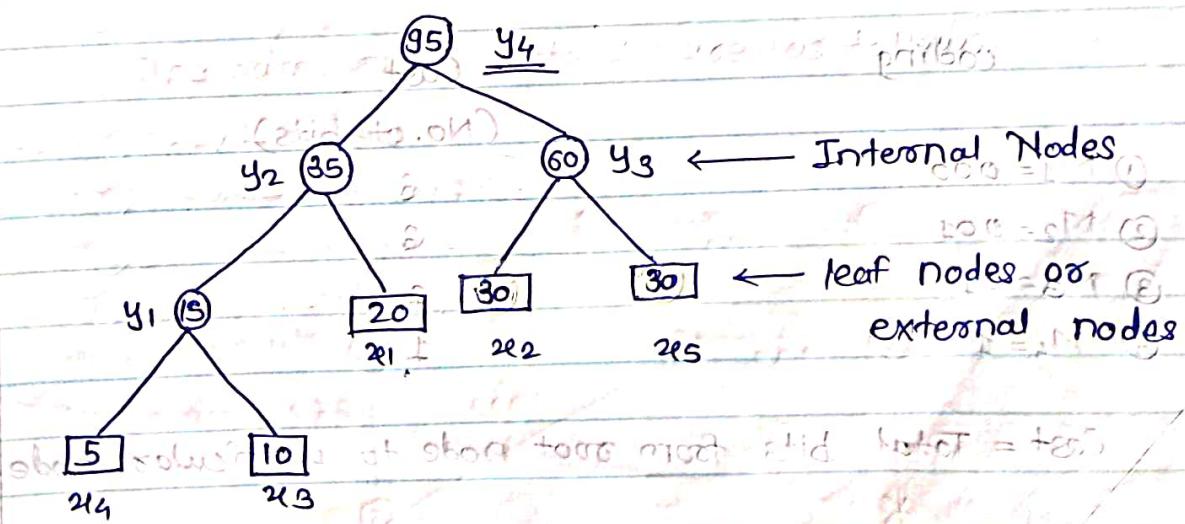
$$y_4 = y_2 + y_3$$

$$= 35 + 60$$

$$y_4 = 95$$

Hence it is the optimal merge pattern of given problem

Tree is as follows



* Huffman's Code:

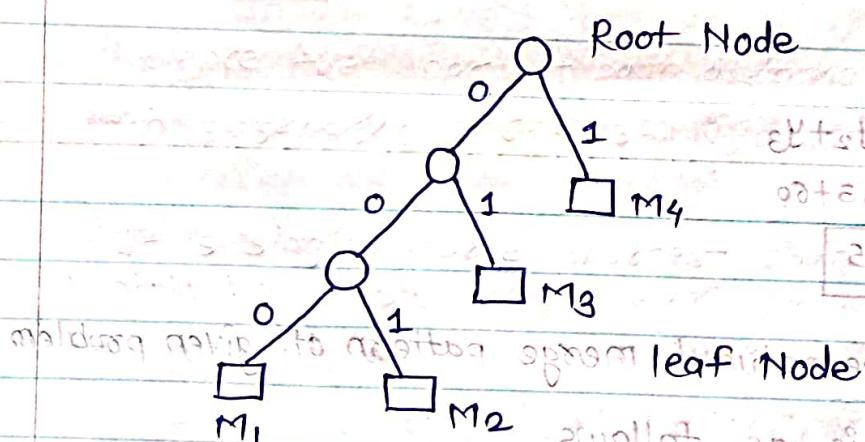
If this code is received at the received side then it depend upon frequency of occurrence. If this code is received at the received side then it will be decoded by decode tree, decode tree is binary tree in which external nodes are represents the messages.

Eg: M_1, M_2, M_3, M_4

In huffman's code the left branch is represented by 0 bit while Right is represented by 1 bit

Huffman's code containing the distance from the nodes of external node.

The distance is represented by the total no. of 0 or 1 bit from each sequencing node.



Coding

- ① $M_1 = 000$
- ② $M_2 = 001$
- ③ $M_3 = 010$
- ④ $M_4 = 1$

Cost
(No. of bits)

3

3

2

1

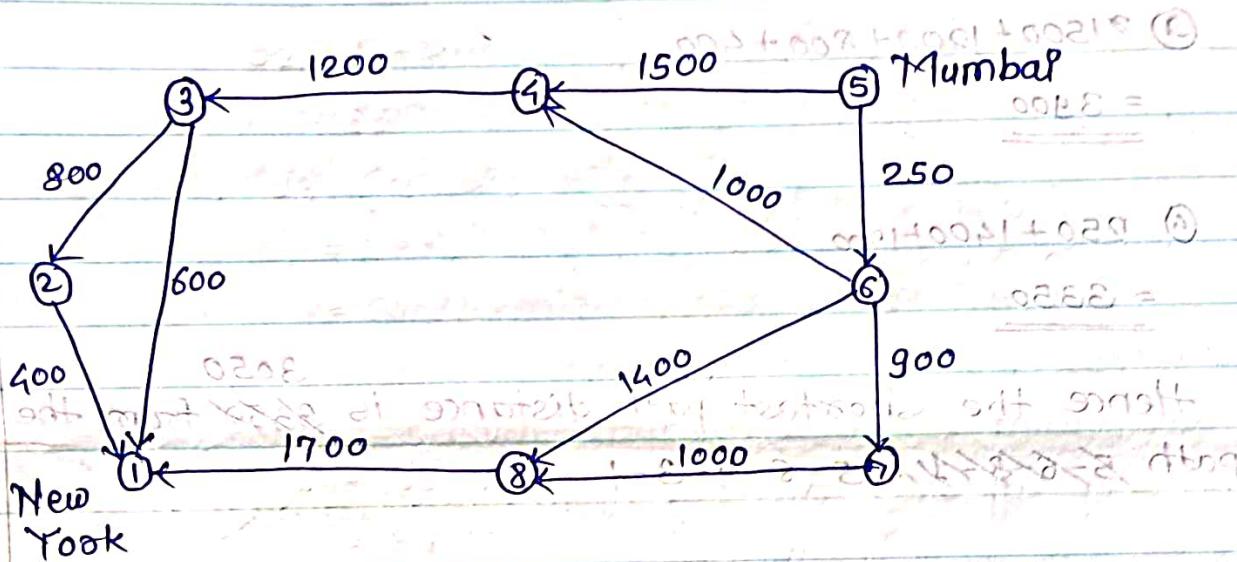
Cost = Total bits from root node to particular node.

Explain single source shortest path.

The single source shortest path is used as its name. It is used for measuring the distances from any two stations.

In the diagram we have given the nodes which represents the name of the city & edge shows the distance between that two cities.

example:



From the above Diagram we have to calculate the shortest path from point 5 to 1 i.e. we have to find distance from Mumbai to New York.

The paths from 5 to 1 are as follows.

- ① 5-4-3-1.
- ② 5-4-3-2-1.
- ③ 5-6-7-8-1.
- ④ 5-6-8-1.
- ⑤ 5-6-4-3-2-1
- ⑥ 5-6-4-3-1

We having only 6 paths from 5 to 1 hence

the values are,

$$\textcircled{1} \quad 1500 + 1200 + 600$$

$$= 3300$$

$$\textcircled{5} \quad 250 + 1000 + 1200 + 800 + 400$$

$$= 3650$$

$$\textcircled{3} \quad 250 + 900 + 1000 + 1700$$

$$= 3850$$

$$\textcircled{6} \quad 250 + 1000 + 1200 + 600$$

$$= 3350$$

$$\textcircled{2} \quad 1500 + 1200 + 800 + 400$$

$$= 3300$$

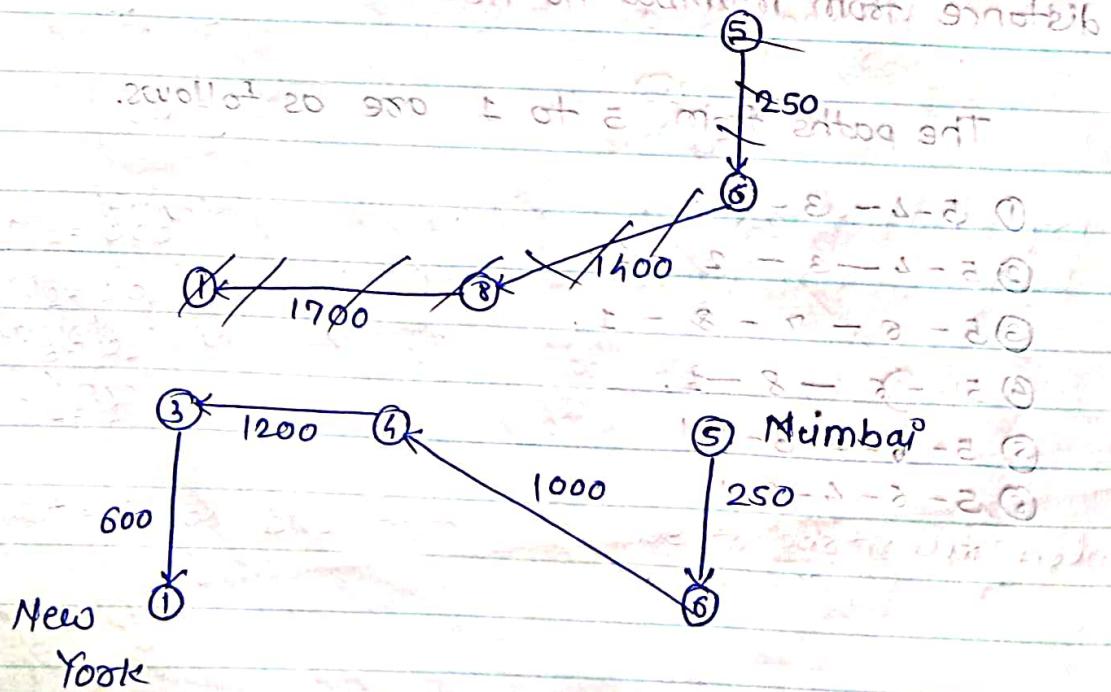
$$\textcircled{7} \quad 250 + 1400 + 1700$$

$$= 3350$$

Hence the shortest path distance is 3350 from the path 5-6-3-1.

It is single source shortest path from mumbai to

New York.



* cost no greater than 0 *

Algorithm shortest path (v , wst , $dist$, n)
{ : minisopt

for $i=1$ to n do; note: minisopt is O(n^2)

{ : minisopt
 minisopt to demand = d ||

// Initialize s demand = 0 ||

$s[i] = \text{false}$;

$dist[i] = wst[v, i]$;

}

$s[v] = \text{true}$;

$dist[v] = 0.0$;

: (i) for num = 2 to $n-1$ do

{ : minisopt O(n^2)

// Determine $n-1$ paths from V

choose u from among those vertices not
in s such that $dist[u]$ is minimum;

set $s[u] := \text{true}$ // put u in s opt

below code: for (each w adjacent to u with $s[w] = \text{false}$) do

// update distances

if ($dist[w] > dist[u] + wst[u, w]$) then

$dist[w] = dist[u] + wst[u, w]$; if min

set $s[w] := \text{true}$ // put w in s opt

below code will do best off spot side minisopt

set $s[w] := \text{true}$ // put w in s opt

dist = dist + 1

if maxspt not greater or below it omit off

from between adjt vertices below spt minisopt is O(n^2)

from between adjt vertices above spt minisopt is O(n^2)

* Optimal Storage on Tapes:

Algorithm :

Algorithm store(n, m)

// n = number of programs

// m = number of tapes

{

j=0;

for i=1 to n do

{

 Write("append program", i,
 "to permutation for tape", j);

 j = (j+1) % mod m;

^{V next} ^{if} ^{last} ^{is-a} ^{minimized} ¹¹

^{total} ^{length} ^{of} ^{programs} ^{most} ⁿ ^{tapes}

 minimum ³ ^{entlib} ^{test} ^{and} ² ⁱⁿ ^{blocks}

If there are n programs that are to be stored on a computer tape of length L, associated with each program i is length l_i , $i \in [n]$.

The all programs could be stored on tape iff sum of length of the programs is at most L.

We assume that however a program is to be retrieved from this tape the tape is initially positioned at the front. Hence if programs are stored in the order.

$I = l_1, l_2, \dots, l_n$.

The time t_j required to retrieve the program i_j . If all programs are retrieved equally then excepted mean retrieval time MRT.

In MRT the taps are stored depend upon permutation.

While minimizing MRT we should minimize the $d(I)$.

Example:

let $n=3$ & $(l_1, l_2, l_3) = (5, 10, 3)$

Now $n=3$

possible orderings $= n! = 6$

possible orderings I will be,

$(1, 2, 3), (1, 3, 2), (2, 1, 3), (2, 3, 1)$

$(3, 1, 2), (3, 2, 1)$

orderings

$d(I)$

$$(1, 2, 3) \quad 5+5+10+5+3+10 = 38$$

$$(1, 3, 2) \quad 5+5+3+5+10+3 = 31$$

$$(2, 1, 3) \quad 10+10+5+10+3+5 = 43$$

$$(2, 3, 1) \quad 10+10+3+10+5+3 = 41$$

$$(3, 1, 2) \quad 3+3+10+8+5+10 = 34$$

$$(3, 2, 1) \quad 3+3+5+3+10+5 = 29$$