11/2/15

UNIT-3

GREEDY METHOD

→ General Method : For a given n inputs of any problem, obtain a subset that satisfies some constraints. Any subset that satisfies these constraints is called a feasible solution.

*. A feasible solution that either maximises or minimises given objective function is called Optimal solution.

→ Algorithm:-

```
Algorithm Greedy(a,n)
{
    solution := ϕ;
    for i:= 1 to n do
    {
        x: =select(a);
        if Feasible(solution,x) then
            solution: = union(Solution,x);
    }
    return solution;
}
```

KNAPSACK PROBLEM :-

For a given n objects and a knapsack bag, object i has a weight w_i and capacity of knapsack is m. If a fraction x_i between $0 \le x_i \le 1$, when object i is placed into the knapsack then the profit of $p_i x_i$ is earned. The objective is to obtain filling of knapsack that maximises the total profit gain. The problem can be stated as

$$\text{Maximize} \sum_{1 \le i \le n} p_i x_i \quad \text{— ①}$$

$$\text{Subject to} \sum_{1 \le i \le n} w_i x_i \le m \quad \text{— ②}$$

and $0 \le x_i \le 1, \ 1 \le i \le n$ — ③

The profits and weights are positive number. Eqn ② will give the feasible solution and eqn ① is an optimal solution which contains maximum value.

For ~~n=~~

Find solution of knapsack problem where $n=3$; $m=20$;

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

S.NO	$\sum w_i x_i$	$\sum P_i x_i$
1) Considering the objects $x_1 \quad x_2 \quad x_3$ $1/2 \quad 1/3 \quad 1/4$	$18 \times 1/2 + 15 \times 1/3 + 10 \times 1/4 = 16.5$	$25 \times \frac{1}{2} + 24 \times \frac{1}{3} + 15 \times \frac{1}{4} = 24.25$
2) Consider the maximum profit 25 24 15 18 15 10 $x_1 \quad x_2 \quad x_3$ $1 \quad 2/15 \quad 0$	$18 \times 1 + \frac{2 \times 15}{15} + 10 \times 0 = \underline{20}$	$25 \times 1 + 15 \times 0 = 28.2$
3) Considering the ~~maxi~~ mini-mum ~~at~~ profit $x_1 \quad x_2 \quad x_3$ $0 \quad 2/3 \quad 1$ 25, 24, 15 - P 18, 15, 10 - w	$18 \times 0 + 15 \times 2/3 + 1 \times 10 = \underline{20}$	$25 \times 0 + 24 \times 2/3 + 15 \times 1 = 31$
4) Greedy approach $\frac{P_i}{w_i}$ ratio $\frac{P_1}{w_1} = \frac{25}{18} = 1.38$ $\quad \frac{P_3}{w_3} = \frac{15}{10} = 1.5$ $\frac{P_2}{w_2} = \frac{24}{15} = 1.6$ $x_1 \quad x_2 \quad x_3$ $0 \quad 1 \quad 1/2$	$18 \times 0 + 15 \times 1 + 10 \times 1/2$ $= 20$	$25 \times 0 + 24 \times 1 + 15 \times 1/2 = 31.5$

For the above problem, the optimal value is 31.5.

MID **

2) $n=7, m=15$, P_1 to $P_7 = 10, 5, 15, 7, 6, 18, 3$ $\quad w_1$ to $w_7 = 2, 3, 5, 7, 1, 4, 1$

→ S.NO | $\Sigma W_i x_i$ | $\Sigma P_i x_i$

1) Considering the objects

x_1 x_2 x_3 x_4 x_5 x_6 x_7

$1/2$ $1/3$ $1/4$ $1/5$ $1/6$ $1/7$ $1/8$

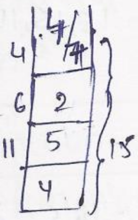
$2\times\frac{1}{2}+\frac{3}{3}+\frac{5}{4}+\frac{7}{5}+\frac{1}{6}+\frac{4}{7}+\frac{1}{8}$

$=5.51$

$\frac{10}{2}+\frac{5}{3}+\frac{15}{4}+\frac{7}{5}+\frac{6}{6}+\frac{18}{7}+\frac{3}{8}$

$=15.76$

2) Considering the maximum profit



x_1 x_2 x_3 x_4 x_5 x_6 x_7

1 0 1 $7/4$ 0 1 0

$2+5+\cancel{7/4}\,\overset{7\times\frac{4}{7}}{}+4$

$=\cancel{23.25}$

$=15$

$10+15+7\cdot\frac{4}{7}+18$

$=\cancel{55.25}$ 47

3) Considering the minimum profit

x_1 x_2 x_3 x_4 x_5 x_6 x_7

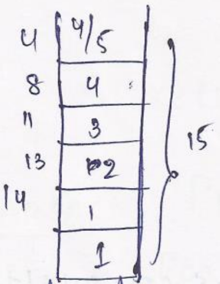
1 1 $4/5$ 0 1 1 1

$2+3+\cancel{5}\,4/5+\cancel{7}\,\cancel{4}+1$

$=\cancel{12}\;15$

$10+\cancel{15}+\overset{3}{\cancel{15}}\times\frac{4}{5}+6\times1+18\times1+3\times1$

$=\cancel{55}\;54$



4) Greedy Approach

$\frac{P_i}{W_i}$ (1) $\frac{10}{2}=5$

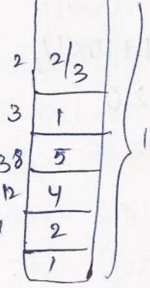
(2) $5/3=1.6$, (3) $\frac{15}{5}=3$

(4) $7/7=1$ (5) $6/1=6$

(6) $18/4=4.5$ (7) $3/1=3$

x_1 x_2 x_3 x_4 x_5 x_6 x_7

1 $2/3$ 1 0 1 1 1



$2+3\cdot2/3+5+1+4+1$

$=15$

$10+5\cdot\frac{2}{3}+15+6+18+3$

$=55.33$

$\begin{array}{r}18\\15\\10\\ \hline 43\end{array}$

Algorithm :-

```
Algorithm GreedyKnapsack(m n)
{
    for i := 1 to n do
        x[i] := 0.0;
        U := m;  U - capacity
    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];
}
```

3) $m = 7$. $n = 3$, P_1 to P_3 = 6, 5, 4, w_1 to w_3 = 5, 4, 3.

S.NO	$\Sigma w_i x_i$	$\Sigma P_i x_i$
1) Considering the objects x_1 x_2 x_3 $1/2$ $1/3$ $1/4$	$\cancel{\frac{6}{2} + \frac{5}{3} + \frac{4}{4}}$ $\cancel{4 + \frac{5}{3} = \frac{17}{3} = 5.66}$ $5/2 + 4/3 + 3/4$ $= \frac{30+16+9}{12} = \frac{55}{12} = 4.5$	$6/2 + 5/3 + 4/4$ $= 5.66$
2) Considering the maximum profits x_1 x_2 x_3 1 $1/2$ 0 2/4, 5 ; 7	$5 + 4/2 = 7$.	$\cancel{5+}$ $6 + 5/2 =$ $= 29/4 = 8.5$
3) Considering minimum profits x_1 x_2 x_3 0 1 1. 4, 3 ; 7	$4\times1 + 3\times1$ $= 7$	$5 + 4 = 9$

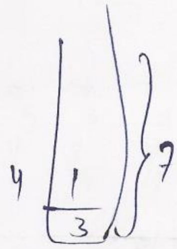
4) Greedy Approach $\sum w_i x_i$ $\sum P_i x_i$

$\frac{P_i}{w_i}$

1) $6/5 = 1.2$

2) $5/4 = 1.25$

3) $4/3 = 1.3$



$4\times1 + 3\times1 = 7$.

$5\times1 + 4\times1 = 9$

x_1	x_2	x_3
0	1	1

The optimal value is $\underline{9}$.

Note:- All optimal solutions will fill the knapsack bag exactly.

*. If $P_1/w_1 \geq P_2/w_2 \geq \cdots P_n/w_n$ then greedy knapsack generates an optimal solutions to the given instance of the knapsack problem.

→ JOB SEQUENCING WITH DEAD LINES:

For a set of n jobs, each job i is associated with an integer dead line $d_i \geq 0$ and $P_i > 0$. For any job i the profit P_i is earned or gained if and only iff job is completed by its dead line.

A feasible solution for this problem is a subset J of jobs J, such that each job in the subset can be completed by its dead line. The value of a feasible solution J is the sum of profits of the jobs in J $\sum_{i \in J} P_i$.

An optimal solution is a feasible with maximal value.

1) $n = 4$ (jobs) P_1 to $P_4 = 100, 10, 15, 27$.

d_1 to $D_4 = 2, 1, 2, 1 \rightarrow d \rightarrow$ dead line

1 2 3 4
100, 10, 15, 27 d_1 to d_4 = 2, 1, 2, 1

Method 1 :-

S.NO	Feasible solution	Processing Sequence	Value
1.	1, 2	2, 1.	100+10=110
2.	1, 3	1, 3 or 3, 1	100+15=115
3.	1, 4	4, 1	100+27 = 127
4.	2, 3	2, 3	10+15 = 25
5.	2, 4	2, 4 or 4, 2	10+27 = 37
6.	3, 4	4, 3	15+27 = 42
7.	1	1	100
8.	2	2	10
9.	3	3	15
10	4	4	27.

For, above problem The optimal value is 127.
Feasible solution is 1, 4
Prossessing Sequence is 4, 1.

~~For~~ ~~a~~ The time complexity of this method is $O(n^2)$. To overcome above complexity Method 2 is followed.

Method 2 :-

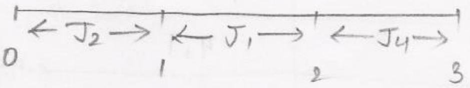
2) n=5, P_1 to P_5 = $\underbrace{20, 15, 10, 5, 1}_{\text{descending order}}$ d_1 to d_4 = 2, 2, 1, 3, 3

0 1 2 3

J	assigned slots	Job considered	action	profit
ϕ	none	1	[1, 2]	0 .
{1}	[1, 2]	2	[0, 1]	20
{1,2}	[0,1], [1,2]	3	rejected no slots	20+15=35
{1, 2,}	[0,1], [1,2]	4	[2,3]	35

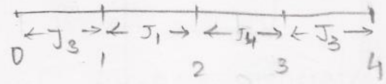
{1,2,4}	[0,1][1,2][2,3]	5	rejected no slots	35+5=40
{1,2,4}	[0,1][1,2][2,3]			=40

Optimal value is 40.



3. $n = 7$.
P_1 to P_7 = 3, 5, 20, 18, 1, 6, 30
d_1 to d_7 = 1, 3, 4, 3, 2, 1, 2.

→ P_1 to P_7 = 30, 20, 18, 6, 5, 3, 1
d_1 to d_7 = P_7 P_3 P_4 P_6 P_2 P_1 P_5
2 4 2 1 3 1 2



maximum dead line = 4.

J	assigned slots	Job considered	action	profit
$\emptyset$	none	7	[1,2]	0
{7}	[1,2]	3	[3,4]	30
{3,7}	[1,2][3,4]	4	[2,3]	30+20=50
{3,4,7}	[2,3][1,2][3,4]	6	rejected slots [0,1]	50+18=68
{3,4,7}	[2,3][1,2][3,4][0,1]	2	rejected slot	68+6=74
{3,4,7,2}	[2,3][1,2][3,4][0,1]	1	rejected slots	68+6=74
{3,4,7}	[2,3][1,2][3,4][0,1]	5	~~[0,1]~~	74
{7,4,3}	[2,3][1,2][3,4][0,1]			74

```
Algorithm JS (d, j, n)
{
    d[0] := J[0] := 0;
     J[1] := 1;
     k := 1;
   for i := 2 to n do
    {
      r := k;
    while ((d[J(0]] > d[i]) and (d[J[r]] ≠ r)) do
      r := r-1;
     if ((d[J[r]] ≤ d[i]) and (d[i] > r)) then
      {
          for q := k to (r+1) step -1 do
           J [q+1] := J[q];
             J [r+1] := i;
             k := k+1;
      }   }
        return k;
}
```

PRIMS ALGORITHM :-

```
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
    if ( cost[i, l] < cost[i, k]) then
      near [i] := l ;
    else near[i] := k ;
   near [k] := near [l] := 0 ;
    for i := 2 to n-1 do
    {
```

```
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
   if ((near [k] ≠ 0) and (cost (k, near [k]) > cost [k, j]))
   then near [k] := j;
  }
 return mincost;
}
```

Time Complexity is $O(n^2)$

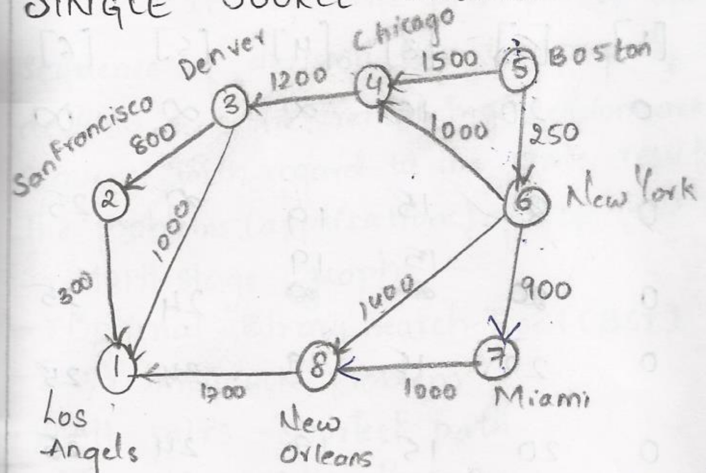
MID ** KRUSHKAL'S ALGORITHM :- All weights are tabulated in icreasing order and drawn the graph for Krushkali

```
Algorithm Krushkal (E, cost, n, t)
{
   construct a heapout of the edges
   cost 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
         from the heap and verify
            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;
}

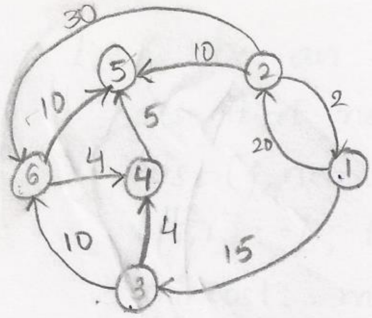
Time Complexity is $O(|E| \log |E|)$

SINGLE SOURCE SHORTEST PATH :



Iteration	S	Vertex Selected	LA [1]	SF [2]	DEN [3]	CHI [4]	BOST [5]	NY [6]	MA [7]	NO [8]
–	–	–	∞	∞	∞	1500	0	250	∞	∞
1	5	6	∞	∞	∞	1250	0	250	1150	1650
2	{5,6}	7	∞	∞	∞	1250	0	250	1150	1650
3	{5,6,7}	4	∞	∞	2450	1250	0	250	1150	1650
4	{5,6,7,4}	8	3350	∞	2450	1250	0	250	1150	1650
5	{6,5,7,4,8}	3	3350	3250	2450	1250	0	250	1150	1650
6	{5,6,7,4,8,3}	2	3350	3250	2450	1250	0	250		
7	{5,6,7,8,3,2}	1	3350							

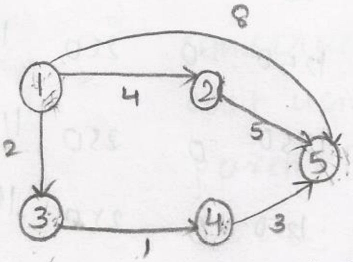
2)



Source 1
destination 6

Iteration	S	Vertex Selected	[1]	[2]	[3]	[4]	[5]	[6]
-	-	-	0	20	15	∞	∞	∞
1	1	3	0	20	15	19	∞	25
2	{1,3}	4	0	20	15	19	24	25
3	{1,3,4}	2	0	20	15	19	24	25
4	{1,3,4,2}	5	0	20	15	19	24	25
5	{1,3,4,2,5}	6	0	20	15	19	24	25

3)



Iteration	S	vertex selected	[1]	[2]	[3]	[4]	[5]
-	-	-	0	4	2	∞	8
1	1	3	0	4	2	3	8
2	{1,3}	4	0	4	2	3	6
3	{1,3,4}	2	0	4	2	3	6
4	{1,3,4,2}	5	0	4	2	3	6