Date 31-10-22
Page

Graphs

Minimum Cost Spanning Tree
Kruskal's
Prim's

Greedy → jis time pe best lag rha hai vo method opted

Single Source Shortest Path
(Positive weights) Dijkstra $O(E + \log v)$
(-ve weights) Bellman Ford $O(VE)$

All Pair Shortest Path
Floyd Warshall $O(n^3)$] Dynamic →

2 types of problem

Greedy Dynamic (Backtracking) Optimization Problem → Minimization or Maximization
Decision Problem → either 0 or 1

Greedy → Feasible (Sol which fulfill criterion)
→ Optimal (Best sol)

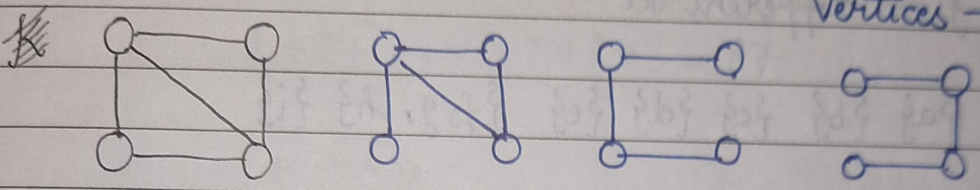
Dynamic → all sol are optimal

Min^m Cost Spanning Tree

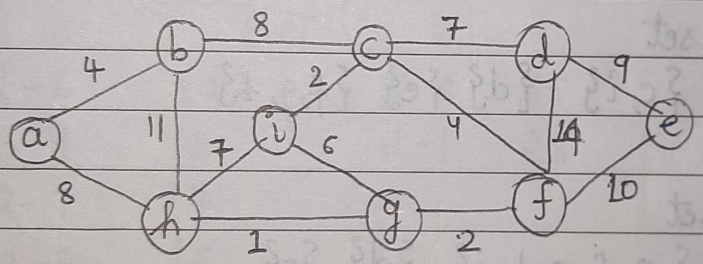
Greedy method → makes a choice that looks better at a moment.
Any subset that satisfies the constraint of a problem is called Feasible solution.
Now we need to find a feasible solution that either maximise or minimise given objective fn
that solution is called Optimal solution

Spanning Tree → Let $G=\{V, E\}$ be an undirected graph. A subgraph $t=\{V, E'\}$ is a spanning tree if & only if t does not contain any loop or cycle.

Min^m Cost Spanning Tree is that spanning tree whose sum of cost of edges is minimum.



Kruskal's Algorithm



Step 1: Sort all the edges in Ascending order

h-g	1
g-f	2
c-i	2
a-b	4
c-f	4
i-g	6
c-d	7
i-h	7
a-h	8
b-c	8
d-e	9
e-f	10
b-h	11
d-f	14

Step 2: Individual sets ~~formed~~ of each vertices

{a} {b} {c} {d} {e} {f} {g} {h} {i}

h-g different set so union of both

{a} {b} {c} {d} {e} {f} {h, g} {i}

g-f → different set

{a} {b} {c} {d} {e} {f, g, h} {i}

c-i diff set

{a} {b} {c, i} {d} {e} {f, g, h}

a-b diff set

{a, b} {c, i} {d} {e} {f, g, h}

c-f diff set

{a, b} {c, f, g, h, i} {d} {e}

i-g same set x

c-d diff set

~~{a}~~ {a, b} {c, d, f, g, h, i} {e}

i-h → x

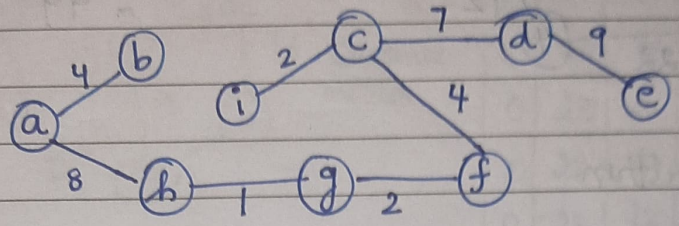
a-h → ✓

{a, b, c, d, f, g, h, i} {e}

b-c → x

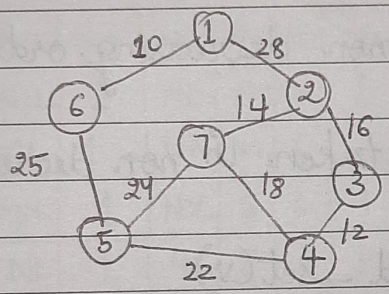
d-e ✓

{a, b, c, d, e, f, g, h, i}



$1 + 2 \overset{+2}{+} 4 + 4 + 7 + 8 + 9 = 37$

8

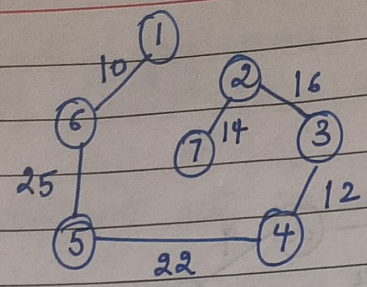


Step 1:

1-6	10	5-6	25
3-4	12	2-1	28
2-7	14		
2-3	16		
7-4	18		
5-4	22		
7-5	24		

Step 2: {1} {2} {3} {4} {5} {6} {7}.

1-6 ✓ {1,6} {2} {3} {4} {5} {7}
3-4 ✓ {1,6} {2} {3,4} {5} {7}
2-7 ✓ {1,6} {2,7} {3,4} {5}
2-3 ✓ {1,6} {2,7,3,4} {5}
7-4 → X
5-4 ✓ {1,6} {2,7,3,4,5}
7-5 X
5-6 ✓ {1,2,3,4,5,6,7}
2-1 X



$10+16+14+12+22+25$
$= 99$

Kruskal Algorithm

1. $A \leftarrow \phi$
2. for each vertex $V \in V[G]$
3. do Make-Set(V)
4. Sort all the edges in non decreasing order by weight w
5. for each edge $(u, v) \in E$ taken in non decreasing order by weight
6. do if find-set $(u) \neq$ find-set (v)
7. $A \leftarrow A \cup (u, v)$
8. Union (u, v)
9. return A

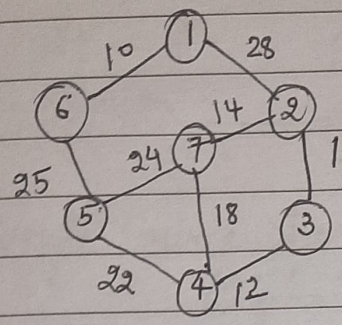
Prims (G, w, r)

1. for each $u \in V[G]$
2. do key $[u] \leftarrow \infty$
3. $\pi[u] \leftarrow$ NIL
4. key $[r] \leftarrow 0$
5. $Q \leftarrow V[G]$
6. while $Q \neq \phi$
7. do Extract Min Q
8. for each edge $V \in$ ~~edge u~~ adjacent u
9. do if $V \in Q$ and $w[u, v] <$ key $[v]$
10. $\pi[v] \leftarrow u$
11. key $[v] \leftarrow w[u, v]$

$\pi \rightarrow$ Parent

$r \rightarrow$ Source vertex

Kruskal (forest) Prims (tree)



Distance matrix

Q =	1	2	3	4	5	6	7
	0	∞	∞	∞	∞	∞	∞
	0	28	∞	∞	∞	10	∞
	0	28	∞	∞	25	10	∞
	0	28	∞	22	25	10	24
	0	28	12	22	25	10	18
	0	16	12	22	25	10	18
	0	16	12	22	25	10	14

Parent matrix

π	1	2	3	4	5	6	7
	NIL	NIL	NIL	NIL	NIL	NIL	NIL
	NIL	1	NIL	NIL	NIL	1	NIL
	NIL	1	NIL	NIL	6	1	NIL
	NIL	1	NIL	5	6	1	5
	NIL	1	4	5	6	1	4
	NIL	3	4	5	6	1	4
	NIL	3	4	5	6	1	2

Extract 1 key val $\overset{min^m}{\wedge}$ initially

adjacent of 1 → 2 & 6

2 is not extract

Wt (1,2) < key [2]

28 < ∞

6 is not extract → Extract '6'

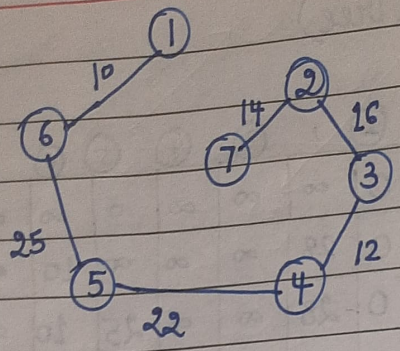
Extract '1'

adj(1) ∈ 2, 6

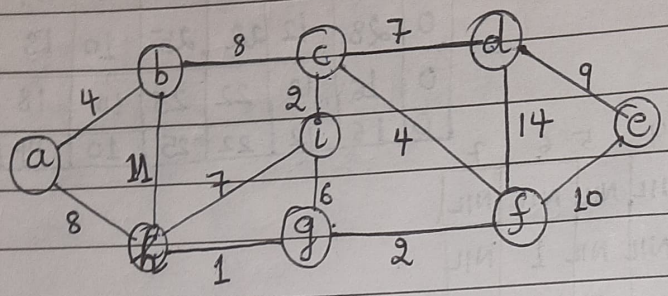
adj (6) ∈ 5

Extract '5' ∈ (4,7)

Min^m Cost 16 + 22 + 12 + 25 + 10 + 14 = 99



Q



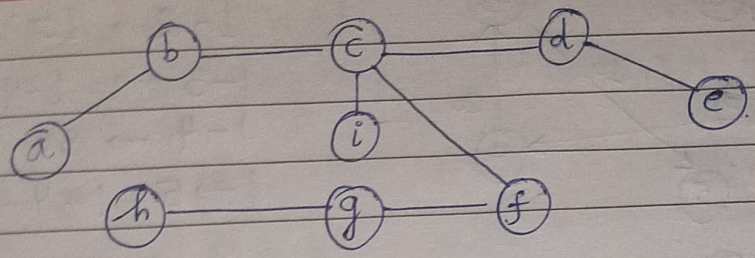
Q

a	b	c	d	e	f	g	h	i
0	∞	∞	∞	∞	∞	∞	∞	∞
0	4	∞	∞	∞	∞	∞	8	∞
0	4	8	∞	∞	∞	∞	8	∞
0	4	8	7	∞	4	∞	8	2
0	4	8	7	∞	4	6	7	2
0	4	8	7	10	4	2	7	2
0	4	8	7	10	4	2	1	2
0	4	8	7	9	4	2	1	2

Extract 'a'
adj(a) ∈ (b, h).
Extract 'c'
adj(c) ∈ (d, f, i)
Extract 'i'
Extract i
adj(i) ∈ (c, h, g)

π

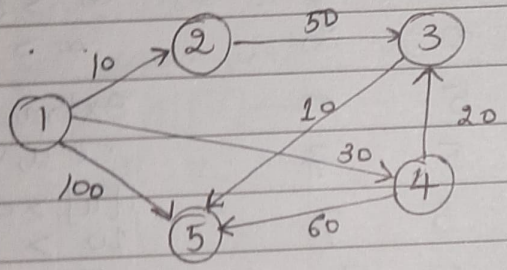
a	b	c	d	e	f	g	h	i
NIL	NIL	NIL	NIL	NIL	NIL	NIL	NIL	NIL
NIL	a	NIL	NIL	NIL	NIL	NIL	a	NIL
NIL	a	b	-	-	-	-	a	-
-	a	b	c	-	c	-	a	c
-	a	b	c	-	c	i	i	c
	a	b	c	f	c	f	i	c
	a	b	c	f	c	f	g	c
	a	b	c	d	c	f	g	c



Bellman ford (Edges pr Relax algo).

Single Source Shortest path
Dijkstra
Bellman ford

Dijkstra Algorithm Graph would be directed



Considering 1 as the source vertex
On Vertices we are going to apply 'Relax Algorithm'
Relax(u,v,w) if $d[v] > d[u] + w(u,v)$
$d[v] \leftarrow d[u] + w[u,v]$
$\pi[v] \leftarrow u$

d	1	2	3	4	5
	0	∞	∞	∞	∞
	0	10	∞	30	100
	0	10	60	30	100
	0	10	50	30	90
	0	10	50	30	60

π	1	2	3	4	5
	NIL	NIL	NIL	NIL	NIL
	NIL	1	NIL	1	1
	NIL	1	2	1	1
	NIL	1	4	1	4
	NIL	1	4	1	3

① $\infty > 0 + 10$

Extract '1'
adj(1) ∈ 2, 5, 4

(1,2)
$d[2] > d[1] + w[1,2]$
$\infty > 0 + 10$
$\infty > 10$
$d[2] \leftarrow 10$
$\pi[2] \leftarrow 1$

Extract '2'
adj(2) ∈ 3

$d[3] > d[2] + w[1,2]$
$\infty > 50 + 10$

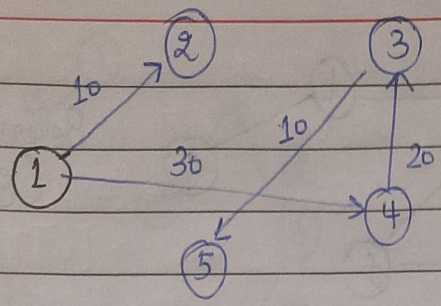
Extract '4'
adj(4) ∈ 3, 5

(4,3)
$d[3] > d[3] + w[4,3]$
~~80~~ > 30 + 20
60 > 50

$d[5] > d[4] + w[4,5]$
$100 > 30 + 60$
$100 > 90$

Extract '3'
adj(3) ∈ 5

$d[5] > d[3] + w[3,5]$
$90 > 50 + 10$

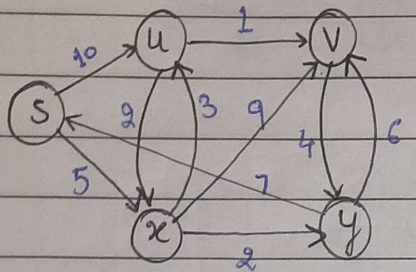


1-2 → 10

1-3 → 1-4-3 = 50

1-4 → 30

1→5 1-4-3-5 = 60



d=	s	u	v	x	y
	0	∞	∞	∞	∞
	0	10	∞	5	∞
	0	8	14	5	7
	0	8	13	5	7
	0	8	9	5	7

π	s	u	v	x	y
	NIL	NIL	NIL	NIL	NIL
		s		s	
		x	x	s	x
	NIL	x	y	s	x
	NIL	x	u	s	x

Extract 's'

$s \in u, x$

(s, u)

$d[u] > d[s] + d[u,s]$

$\infty > 0 + 10$

$d[x] > d[s] + d[x,s]$

$\infty > 0 + 5$

$\infty > 5$

Extract 'x'

$s \in u, v, y$

$d[u] > d[x] + d[x,u]$

$10 > 5 + 3$

$10 > 8$

$d[v] > d[x] + d[v,x]$

$\infty > 5 + 9$

$\infty > 14$

$d[y] > d[x] + d[y,x]$

$\infty > 5 + 2$

$\infty > 7$

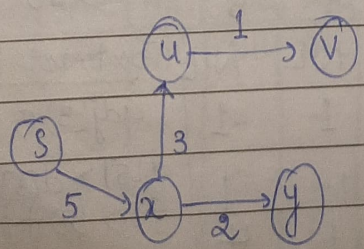
Extract 'y'

$y \in v, s$

$d[v] > d[y] + d[v,y]$

$14 > 7 + 6$

$d[s] > d[y] + d[$



Dijkstra (G, W, S)

```
1. Initialise single source (G, w, s)
2.    S ← ϕ
3.    Q ← V[G]
4.    while Q ≠ ϕ
5.    do u ← Extract_min(Q)
6.     S ← S ∪ {u}.
7.     for each vertex v ∈ adj(u)
8.     do relax (u, v, w)
```

Initialise Single Source (G, w, s)

```
1.  for each vertex v ∈ V[G]
2.   do d[v] ← ∞
3.      π[v] ← NIL
4.      d[s] ← 0
```

Relax (u, v, w)

```
1.  if d[v] > d[u] + w[u, v].
2.  then d[v] ← d[u] + w[u, v]
3.   π[v] ← u
```

Bellman Ford

Bellman Ford (G, w, s)

```
1.  Initialise single source (G, w, s)
2.  for i ← 1 to |V[G]| − 1          ]
3.  for each edge (u, v) ∈ G         ] edge Relax
4.   do Relax (u, v, w)             ]
5.  for each edge (u, v) ∈ G
6.   if d[v] > d[u] + w[u, v]
7.  return false  Print -ve weight cycle exists
8.  return True
```

VE + V
= O(VE)

V

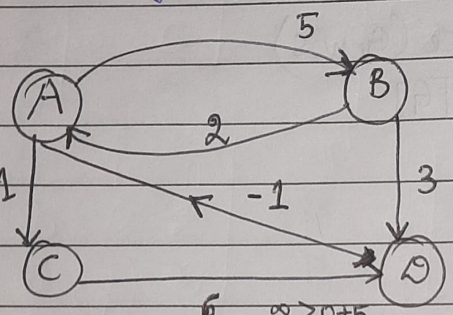
E

Initialise singleSource (G, W, s)

1. for each vertex $v \in V[G]$
2. do $d[v] \leftarrow \infty$
3. $\pi[v] \leftarrow NIL$
4. $d[s] \leftarrow 0$

Relax (u, v, w)

1. if $d[v] > d[u] + w[u,v]$
2. $d[v] \leftarrow d[u] + w[u,v]$
3. $\pi[v] \leftarrow u$



Condition false over each iteration no -ve cycle

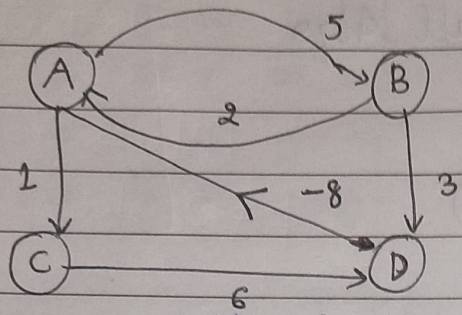
Loop ~~from~~ to no of edges - 1

→ To detect -ve wt cycle

	Weight	i=1	i=2	i=3	i=4
A-B	5	$A \overset{0}{=} B \overset{5}{\leq} \infty$	$A \overset{0}{=} B \overset{5}{\leq} 5$		A - B
A-C	1	$A \overset{0}{=} C \overset{1}{\leq} \infty$	$A \overset{0}{=} C \overset{1}{\leq} 1$		A - C
B-A	2	$B \overset{5}{-} A \overset{2}{\leq} 0$ No change	$B \overset{5}{-} A \overset{2}{\leq} 0$		B - A
B-D	3	$B \overset{5}{-} D \overset{3}{\leq} \infty$	$B \overset{5}{-} D \overset{3}{\leq} 7$		B - D
C-D	6	$C \overset{1}{-} D \overset{6}{\leq} 8$	$C \overset{1}{-} D \overset{6}{\leq} 7$		C - D
D-A	-1	$D \overset{7}{-} A \overset{-1}{\leq} 0$	$D \overset{7}{-} A \overset{-1}{\leq} 0$		D - A

d	A	B	C	D
	0	∞	∞	∞
Q	5	1	8 7	
	0	5	1	7
	0	5	1	7

π	A	B	C	D
	NIL	NIL	NIL	NIL
	NIL	A	A	B
	NIL	A	A	C
	NIL	A	A	C



	Weight	i=1	i=2	i=3
A-B	5	~~A~~-B	A-B	A-B
A-C	1	A-C	A-C	
B-A	2	B-A	B-A	
B-D	3	B-D	B-D	
C-D	6	C-D	C-D	
D-A	-8	D-A	D-A	

d

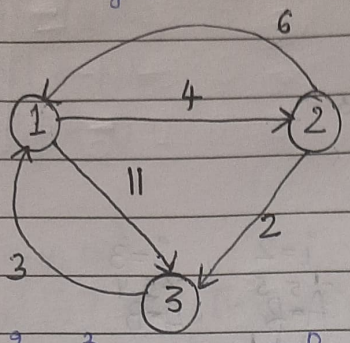
A	B	C	D
0	∞	∞	∞
~~∅~~-1	5	1	~~∞~~7
-2	4	0	6
-3	3	-1	5

π

A	B	C	D
NIL	NIL	NIL	NIL
D	A	A	~~B~~/C
D	A	A	C
D	A	A	C

All Pair Shortest Path

→ Floyd Warshall Algo



$D^0 =$

	1	2	3
1	0	4	11
2	6	0	2
3	3	∞	0

Distance

$\pi^0 =$

	1	2	3
1	NIL	1	1
2	2	NIL	2
3	3	3 NIL	NIL

Parent matrix

$D^1 =$

	1	2	3
1	0	4	11
2	6	0	2
3	3	(7)	0

$\pi^1 =$

	1	2	3
1	NIL	1	1
2	2	NIL	2
3	3	(1)	NIL

$d_{ij}^{k} = \min\{ d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1} \}$

k → Intermediate vertex

(2, 3) via 1

$D^2 =$

	1	2	3
1	0	4	6
2	6	0	2
3	3	7	0

$\pi^2 =$

	1	2	3
1	NIL	1	2
2	2	NIL	2
3	3	1	NIL

$2, d_{21}^{0} + d_{13}^{0}$

$6 + 11$

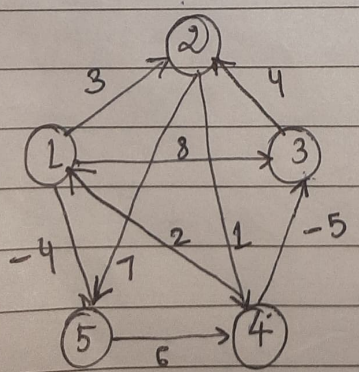
$\min(2, 17)$

$D^3 =$

	1	2	3
1	0	4	6
2	(5)	0	2
3	3	7	0

$\pi^3 =$

	1	2	3
1	NIL	1	2
2	3	NIL	2
3	3	1	NIL



$D^0 =$

	1	2	3	4	5
1	0	3	8	∞	-4
2	∞	0	∞	1	7
3	∞	4	0	∞	∞
4	2	∞	-5	0	∞
5	∞	∞	∞	6	0

$\pi^0 =$

	1	2	3	4	5
1	NIL	1	1	NIL	1
2	NIL	NIL	NIL	2	2
3	N	3	N	N	N
4	4	N	4	N	N
5	N	N	N	5	N

$D^1 =$

	1	2	3	4	5
1	0	3	8	∞	-4
2	∞	0	∞	1	7
3	∞	4	0	∞	11
4	2	-1	-5	0	-2
5	∞	7	1	6	0

$\pi^1 =$

	1	2	3	4	5
1	N	1	1	N	1
2	N	N	N	2	2
3	N	3	N	N	N
4	4	(1)	4	N	(1)
5	N	N	N	5	N

$D^1 =$

	1	2	3	4	5
1	0	3	8	+∞	-4
2	∞	0	∞	1	7
3	∞	4	0	∞	∞
4	2	5	-5	0	-2
5	∞	∞	∞	6	0

$D^2 =$

	1	2	3	4	5
1	0	3	8	(4)	-4
2	∞	0	∞	1	7
3	∞	4	0	(5)	(11)
4	2	5	-5	0	-2
5	∞	∞	∞	6	0

$\pi^2 =$

	1	2	3	4	5
1	N	1	1	2	1
2	N	N	N	2	2
3	N	3	N	2	2
4	4	1	4	N	1
5	N	N	N	5	N

$D^3 =$

	1	2	3	4	5
1	0	3	8	4	-4
2	∞	0	∞	1	7
3	∞	4	0	5	11
4	2	-1	-5	0	-2
5	∞	∞	∞	6	0

$\pi^3 =$

	1	2	3	4	5
1	N	1	1	2	1
2	N	N	N	2	2
3	N	3	N	2	2
4	4	3	4	N	1
5	N	N	N	5	N

Longest Common Subsequence

X = BC BDCB
Y = BC BDAB

Common substring
BCBD → Length → 4

X = BACDB
Y = BDCB

Common Subsequence
↓
consecutive index
not mandatory

Date
Page

		0	1	2	3	4	5
	X/Y	B	A	C	D	B	
	0	0	0	0	0	0	
1	B	0	1↑↖	1←	1←	1←	1↖
2	D	0	1↑	1↑	1↑	2↑↖	2←
3	C	0	1↑	1↑	2↑	2↑↑	2↑
4	B	0	1↑	1↑	2↑	2↑	3↑

BDB

3 → Length of common subsequence

$$C_{ij} = \begin{cases} 0 & \text{if } i \& j = 0 \\ c[i-1, j-1] + 1 & \text{if } x_i = y_j \\ \max\{c[i, j-1], c[i-1, j]\} & \text{if } i, j > 0 \text{ and } x_i = y_j \end{cases}$$

diagonal

X = MNOPABCD

Y = NPCDB.

		M	N	O	P	A	B	C	D
	0	0	0	0	0	0	0	0	0
N	0	0←	1↑↖	1←	1←	1←	1←	1←	1←
P	0	0←	1↑	1↑	2↑↖	2←	2←	2←	2←
C	0	0←	1↑	1↑	2↑	2↑	2↑	3↑↖	3←
D	0	0←	1↑	1↑	2↑	2↑	2↑	3↑	4↑↖
B	0	0←	1↑	1↑	2↑	2↑	3↑↖	3↑	4↑

NPCD

Complex

Complexity O(mn)

```
LCS(X, Y)
1.  m ← Length(X)
2.  n ← Length(Y)
3.  for i ← 0 to m
4.  do c[i, 0] ← 0
5.  for j ← 0 to n
6.  do c[0, j] ← 0
7.  for i ← 1 to m
8.  for j ← 1 to n
9.  do if xi = yj
10. c[i, j] ← c[i-1, j-1] + 1
11. B[i, j] = "↖"
12. else if c[i-1, j] > c[i, j-1]
13. c[i, j] ← c[i-1, j]
14. B[i-j] = "↑"
15. else
16.
```

Date 11 Nov 2022
Page

Knapsack Problem
↳ Fractional Knapsack [Greedy Strategy]
↳ 0/1 Knapsack → Fractions are not allowed.
↓
[Dynamic Programming]

} Bag capacity does not exceed and max profit should be earned

We have given n objects and a knapsack with capacity m. Each object i has some weight w_i and profit p_i. If fraction x_i $[0 \le x_i \le 1]$ is placed in knapsack then some profit $p_i x_i$ is earned.
The main objective is to fill the knapsack in such a way that maximise the total profit earned.

$$\text{maximises} \sum_{i=1}^{n} p_i x_i$$

$$\text{subject to} \sum_{i=1}^{n} w_i x_i \le m$$

where $0 \le x_i \le 1$

$n = 3 \quad m = 20$ Greedy → sorting

$P_1 \; P_2 \; P_3 = 25 \; 24 \; 15$

$W_1 \; W_2 \; W_3 = 18, \; 15, \; 10$

Calculate sol vector, Max Profit earned.
(which item select or reject)

Using Greedy

$\frac{P_1}{W_1} = \frac{25}{18} = 1.3 \qquad \frac{P_2}{W_2} = \frac{24}{15} = 1.6 \qquad \frac{P_3}{W_3} = \frac{15}{10} = 1.5$

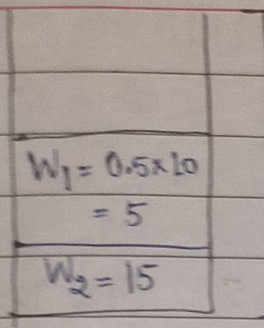
Step 2: Apply sorting

$$\frac{P_2}{W_2} > \frac{P_3}{W_3} > \frac{P_1}{W_1}$$

$1.6 > 1.5 > 1.3$

$i = \quad ① \quad ② \quad ③$

Date
Page



$\frac{P_i}{W_i}$ → greater first added to bag

$\frac{\text{Left out space of bag}}{\text{Wt of object } i} : \frac{5}{10} = 0.5$

Solution vector = $\{x_1, x_2, x_3\}$

$\{0, 1, 1/2\}$

Max Profit → $P_1x_1 + P_2x_2 + P_3x_3$

$25 \times 0 + 24 \times 1 + 15 \times 1/2 = 31.5$

1. $n = 7 \quad m = 15$

$P_1\ P_2\ P_3\ P_4\ P_5\ P_6\ P_7 = \{10, 5, 15, 7, 6, 18, 3\}$

$W_1\ W_2\ W_3\ W_4\ W_5\ W_6\ W_7 = \{2, 3, 5, 7, 1, 4, 1\}$

Cal. sol vector, Max Profit

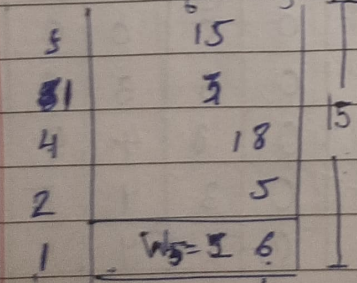
$\frac{P_1}{W_1} = 5 \quad \frac{P_2}{W_2} = \frac{5}{3} = 1.6 \quad \frac{P_3}{W_3} = \frac{15}{5} = 3 \quad \frac{P_4}{W_4} = 1 \quad \frac{P_5}{W_5} = \frac{6}{1} = 6$

$\frac{P_6}{W_6} = \frac{18}{4} = 4.5 \quad \frac{P_7}{W_7} = 3.$

Apply sorting

$\frac{P_5}{W_5} > \frac{P_1}{W_1} > \frac{P_6}{W_6} > \frac{P_3}{W_3} > \frac{P_7}{W_7} > \frac{P_2}{W_2} > \frac{P_4}{W_4}$

$6 \quad 5 \quad 4.5 \quad 3 \quad 3 \quad 1.6 \quad 1$



Solvector = $\{x_1\ x_2\ x_3\ x_4\ x_5\ x_6\ x_7\}$

$\{1\ 2/3\ 1\ 0\ 1\ 1\ 1\}$

Max Profit = 55.33

2 Kg

$\frac{2}{3} \times 15 \Rightarrow 3.33$

Date
Page

Apply Greedy Strategy to 0/1 knapsack

$n=5 \quad W=60$

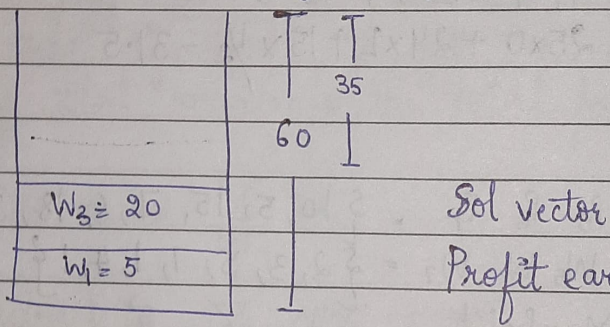
$\{W_1 \text{ —— } W_5\} = \{5, 10, 20, 30, 40\}$

$\{P_1 \text{ —— } P_5\} = \{30, 20, 100, 90, 160\}$

$\frac{P_1}{W_1} = \frac{30}{5} = 6 \quad \frac{P_2}{W_2} = \frac{20}{10} = 2 \quad \frac{P_3}{W_3} = \frac{100}{20} = 5 \quad \frac{P_4}{W_4} = \frac{90}{30} = 3$

$\frac{P_5}{W_5} = \frac{160}{40} = 4.$

$\frac{P_1}{W_1} > \frac{P_3}{W_3} > \frac{P_5}{W_5} > \frac{P_4}{W_4} > \frac{P_2}{W_2}$

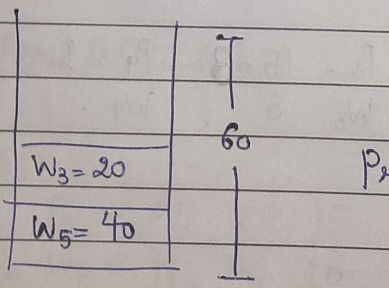


cannot be ered

Sol vector = {1, 0, 1, 0, 0}

Profit earned = 130

hest it is 0



Profit = 100 + 160

= 260

$n=4 \quad m=5.$

$W_1 \; W_2 \; W_3 = \{2, 3, 4, 5\}$

$P_1 \; P_2 \; P_3 = \{3, 4, 5, 6\}$.

t item

2 items $W_1 = 2$ 2kg ko 1kg me nhi daal sakte ✗

Profit Matrix

i \ W	0	1	2	3	4	5
0	0	0	0	0	0	0
✓ 1	0	0	3	3	3	3
✓ 2	0	0	3	4	4	7
3	0	0	3	4	5	7
✗ 4	0	0	3	4	5	7

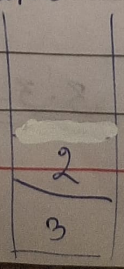
→ If values are same exclude item 4

Exclude item 4

Exclude item 3

Include 2

Include 1



Greedy Knapsack(m, n)

① Compute $\frac{P_i}{W_i}$ such that $\frac{P_i}{W_i} > \frac{P_{i+1}}{W_{i+1}}$

② for i ← 1 to n do x[i] = {0.0}

③ for i ← 1 to n do

④ if W[i] > m then break

⑤ x[i] = 1.0 , U = U − W[i]

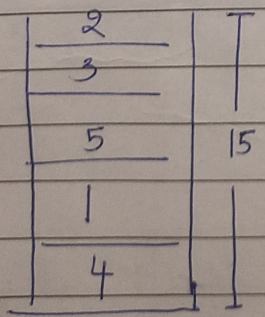
⑥ if i ≤ n then x[i] = U/W[i]

n = 7 m = 15

W_1 —— W_7 = {2, 3, 5, 7, 1, 4, 1}

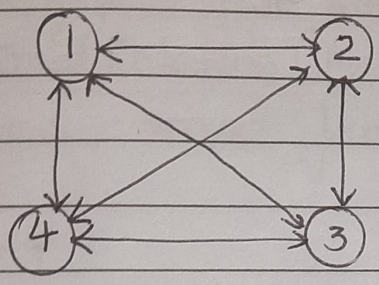
P_1 —— P_7 = {10, 5, 15, 7, 6, 18, 3}.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	10	10	10	10	10	10	10	10	10	10	10	10	10	10
2	0	0	10	10	10	15	15	15	15	15	15	15	15	15	15	15
3	0	0	10	10	10	15	15	25	25	25	30	30	30	30	30	30
4	0	0	10	10	10	15	15	25	25	25	30	30	30	30	32	32
5	0	6	10	16	16	16	21	25	31	31	31	36	36	36	36	38
6	0	6	10	16	18	24	28	34	34	34	40	43	43	49	49	54
7	0	6	10	16	19	24	28	34	37	37	39	43	49	52	52	54



Travelling Sales Person Problem.

Given a set of cities & distance b/w each pair of cities. The problem is to find the shortest path that visits every city exactly once & return to its starting position.



	1	2	3	4
1	0	(10)	15	20
2	5	0	(9)	10
3	6	13	0	(12)
4	8	8	9	0

Brute Force → n! combinations.

Greedy 1→2 min^m

1→2→3→4→1

Dynamic Programming, cost

$g(2, \phi) = C_{21} = 5$

$g(3, \phi) = C_{31} = 6$

$g(4, \phi) = C_{41} = 8$

2→1 via 3 or 4

$g\{2, \{3\}\} = C_{23} + C_{31} = 9+6 = 15$

$g\{2, \{4\}\} = C_{24} + C_{41} = 10+8 = 18$

3→1 via 2 or 4

$g\{3, \{2\}\} = C_{32} + C_{21} = 13+5 = 18$

$g\{3, \{4\}\} = C_{34} + C_{41} = 12+8 = 20.$

$g\{4, \{2\}\} = C_{42} + C_{21} = 8 + 5 = 13$

$g\{4, \{3\}\} = C_{43} + C_{31} = 9 + 6 = 15$

$g\{2, \{3, 4\}\} = C_{23} + C_{34} + C_{41} = 9 + 12 + 8 = 29$

$g\{2, \{4, 3\}\} = C_{24} + C_{43} + C_{31} = 25$

$g\{3, \{2, 4\}\} = 31$

$g\{3, \{4, 2\}\} = 25 \rightarrow$ min

$g\{4, \{2, 3\}\} = 23 \rightarrow$ min

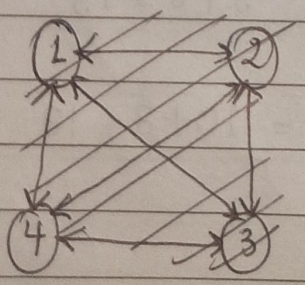
$g\{4, \{3, 2\}\} = 27$

$$\min \begin{cases} C_{12} + g\{2, \{4, 3\}\} \\ C_{13} + g\{3, \{4, 2\}\} \\ C_{14} + g\{4, \{2, 3\}\} \end{cases} = \begin{cases} 10 + 25 \\ 15 + 25 \\ 20 + 23 \end{cases} \begin{matrix} 35 \rightarrow \min \\ 40 \\ 43 \end{matrix}$$

Min^m Path

$1 \rightarrow 2 \rightarrow 4 \rightarrow 3 \rightarrow 1$

Q.



	1	2	3	4	5
1	0	20	30	10	11
2	15	0	16	4	2
3	3	5	0	2	4
4	14	16	18	0	3
5	16	4	7	16	0

Date
Page

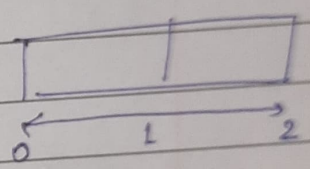
Task Scheduling Problem (Greedy)

We have a set of n jobs & each job is assigned a deadline d_i, $d_i > 0$ for i^{th} job profit P_i is earned if job is completed by its deadline.
To complete a job 1 unit of time is given.
If feasible sol for this problem is a subset of jobs such that each job in this subset is completed by its deadline.
Optimal Solution (Max Profit) is that feasible sol that earns maxm profit.

$n = 4$

$P_1\ P_2\ P_3\ P_4 = \{100, 10, 15, 27\}$

$d_1\ d_2\ d_3\ d_4 = \{2, 1, 2, 1\}$.
0 1 2 3.



Each job is given one unit of time.

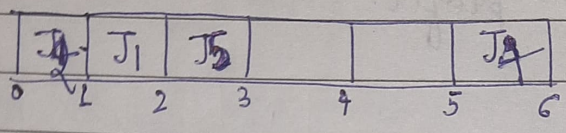
Job	Feasible sol	Processing Sequence	Profit
1	1	1	100
2	2	2	10
3	3	3	15
4	4	4	27
{1,2}	{1,2}	2, 1	110
{1,3}	{1,3}	1,3 or 3,1	115
{1,4}	{4,1}	4, 1	127. ✓
{2,3}	{2,3}	2,3	25
{2,4}	It cannot be processed.		
{3,4}	{3,4}	4,3	42

$n = 5$

P_i	50	20	15	30	45
d_i	2	1	2	6	3.

Step 1: Sort in descending order (Profit)

1	2	3	4	5
50	45	30	20	15
J_1	J_5	J_4	J_2	J_3
2	3	6	1	2



Maxm Profit

~~185~~ 145

*

* Activity Selection Problem. (Greedy)

We have a set of activities [1, 2 . . . . n]

These activities are to be scheduled to use same resource. Each activity 'i' has starting time 'Si' & finishing time 'fi' such that $S_i \leq f_i$

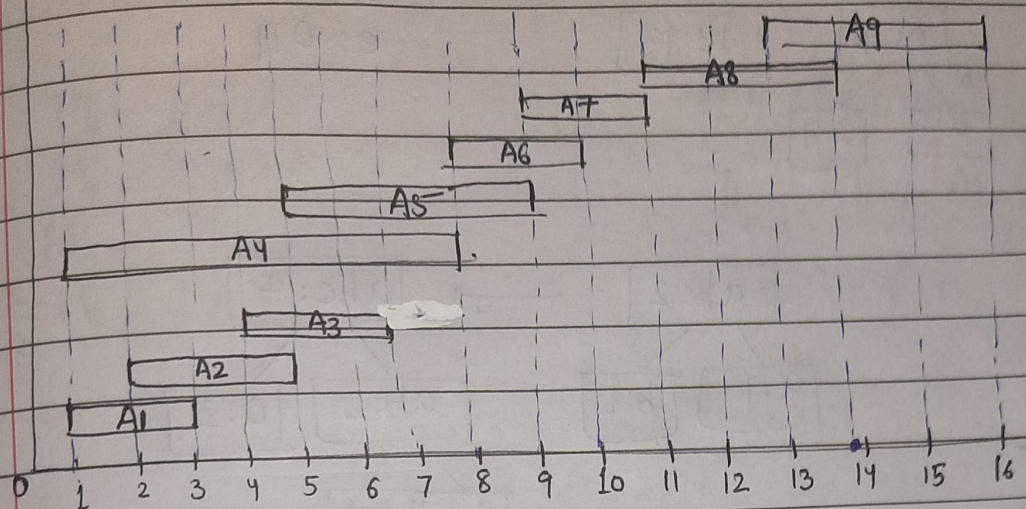
We have to schedule the activity in such a way such that no two activities should overlap.

For eg: $S = \langle 1, 2, 3, 4, 5, 6, 7, 8, 9 \rangle$

$S_i = \langle 1, 2, 4, 1, 5, 8, 9, 11, 13 \rangle$

$f_i = \langle 3, 5, 7, 8, 9, 10, 11, 14, 16 \rangle$

Sort all the activities in increasing order wrt finishing time.



Optimal solution = {A1, A3, A6, A8}

Greedy Activity Selection (s, f)

1. n ← length[s]
2. A ← {a1}
3. i ← 1
4. for m ← 2 to n
5. do if sm ≥ fi
6. then A ← A ∪ {am}
7. i ← m
8. return A

Hoffmann Coding (greedy).

Used in compression.

Code	Freq
a:	5
b:	48
c:	7
d:	17
e:	13

a: 5
c: 7
e: 13
d: 17
b: 48