Assignment 4
Of
Modelling & Simulation Lab (CS1052)

Masters of Technology in Computer Science And Engineering

submitted by
Arghya Bandyopadhyay
RollNo. 20CS4103

submitted to
Dr Nanda Dulal Jana
Assistant Professor
Dept. of CSE

National Institute of Technology, Durgapur

Problem 1

Problem statement 1

Find the optimal solutn to the following transportatn problem in which the cells contains the unit transportatn cost in ruppees.

	W_1	W_2	W_3	W_4	W_5	avail.
F_1	7	6	4	5	9	40
F_2	8	5	6	7	8	30
F_3	6	8	9	6	5	20
F_4	5	7	7	8	6	10
Req.	30	30	15	20	5	

use NWCR and LCM for initial basic feasible solutn.

Problem formulatn –

	W_1	W_2	W_3	W_4	W_5	Supply
F_1	x_{11} 7	x_{12} 6	x_{13} 4	x_{14} 5	x_{15} 9	40
F_2	x_{21} 8	x_{22} 5	x_{23} 6	x_{24} 7	x_{25} 8	30
F_3	x_{31} 6	x_{32} 8	x_{33} 9	x_{34} 6	x_{35} 5	20
~~Bondwal~~ F_4	x_{41} 5	x_{42} 7	x_{43} 7	x_{44} 8	x_{45} 6	10
Dem.	30	30	15	20	5	

The transportation problem is formulated as an LP model as follows —

Minimize (Total T.P cost) —

$$7x_{11} + 6x_{12} + 4x_{13} + 5x_{14} + 9x_{15} \cancel{\le 40}$$
$$+ 8x_{21} + 5x_{22} + 6x_{23} + 7x_{24} + 8x_{25} \cancel{\le 30}$$
$$+ 6x_{31} + 8x_{32} + 9x_{33} + 6x_{34} + 5x_{35} \cancel{\le 20}$$
$$+ 5x_{41} + 7x_{42} + 7x_{43} + 8x_{44} + 6x_{45} \cancel{\le 10}$$

Subject to constraint —

$$7x_{11} + 6x_{12} + 4x_{13} + 5x_{14} + 9x_{15} = 40$$
$$8x_{21} + 5x_{22} + 6x_{23} + 7x_{24} + 8x_{25} = 30$$
$$6x_{31} + 8x_{32} + 9x_{33} + 6x_{34} + 5x_{35} = 20$$
$$5x_{41} + 7x_{42} + 7x_{43} + 8x_{44} + 6x_{45} = 10$$

First we will solve it using north west corner rule. The allocatn matrix is such that.

	W_1	W_2	W_3	W_4	W_5	Sup
F_1	7 (30)	6 (10)	4	5	9	~~40~~ ~~10~~ 0
F_2	8	5 (20)	6 (10)	7	8	~~30~~ ~~10~~ 0
F_3	6	8	9 (5)	6 (15)	5	~~20~~ ~~15~~ 0
F_4	5	7	7	8 (5)	6	10 ~~5~~ 0
Demand	~~30~~ 0	~~30~~ ~~20~~ 0	~~15~~ ~~5~~ 0	~~20~~ ~~15~~ 0	~~5~~ 0	

The step by step descriptn of this solutn is given as :-

and

$$7x_{11} + 8x_{21} + 6x_{31} + 5x_{41} = 30$$

$$6x_{12} + 5x_{22} + 8x_{32} + 7x_{42} = 30$$

$$4x_{13} + 6x_{23} + 9x_{33} + 7x_{43} = 15$$

$$5x_{14} + 7x_{24} + 6x_{34} + 8x_{44} = 20$$

$$9x_{15} + 8x_{25} + 5x_{35} + 6x_{45} = 5$$

and $x_{ij} \geq 0$ for $i = 1, 2, 3$ &

$j = 1, 2, 3, 4$

Soln

Total no of supply const. = 4

Total no of demand const = 5

~~The sym val for $r_1 = 40$ & $r_2 = 30$ compared.~~

Total supply = $40 + 30 + 20 + 10$

$= 100$

Total demand = $30 + 30 + 15 + 20 + 5$

$= 100$

Total supply = Total demand

$\Rightarrow$ Balanced Transp. problem.

North west corner rule

The sim val $F_1 = 40$ & $W_1 = 30$ $\min(F_1, W_1) = 30$ is assigned to $F_1 W_1$, This meets the demand of W_1 & leaves $40 - 30 = 10$ with F_1

The sim value for $F_2 = 30$ & $W_2 = 20$ are compared. The $\min(F_2, W_2) = 20$. This meets the demand of W_2 & leaves $30 - 20 = 10$ units with F_2

The sim values for $F_2 = 10$ and $W_3 = 15$ are compared. The smaller of the two i.e $\min(10, 15) = 10$ is assign to $F_2 W_3$, this exhaust the capacity of F_2 and leaves 15 $-10 = 5$ units with W_3.

The min value for $F_3 = 20$ and $W_3 = 5$ are compared. The min $(20, 5) = 5$ is assigned to $F_3 W_3$ this meets the complete demand of W_3 and leaves $20 - 5 = 15$ units with F_3

The min values for $F_3 = 15$ and $W_4 = 20$ are compared.
The smaller of $15, 20 = 15$ is assigned for $F_3 W_4$, this exhaust the capacity of F_3 and leaves $20 - 15 = 5$ with W_4.

The min value for $F_4 = 10$ & $W_4 = 5$ are compared. The

min (10, 5) = 5 is assign to $F_4 W_4$
This meets the complete demand of W_4 and leaves 10 - 5 = 5 units with F_4

The min value for F_4 = 5 & W_5 = 5 are compared.
The smaller of the two i.e min (5, 5) = 5 is assign to $F_4 W_5$
The IBFS =

	W_1	W_2	W_3	W_4	W_5	supp.
F_1	7 (30)	6 (10)	4	5	9	40
F_2	8	5 (20)	6 (10)	7	8	30
F_3	6	8	9 (5)	6 (15)	5	20
F_4	5	7	7	8 (5)	6 (5)	10
Demand	30	30	15	20	5	

$v_1 = 7 - 0 = 7$

$v_2 = 6 - 0 =$

$v_1 = c_{11} - u_1 = 7 - 0 \Rightarrow v_1 = 7$

$v_2 = c_{12} - u_1 = 6 - 0 \Rightarrow v_2 = 6$

$u_2 = c_{22} - v_2 = 5 - 6 \Rightarrow u_2 = -1$

$v_3 = c_{23} - u_2 = 6 + 1 \Rightarrow v_3 = 7$

$u_3 = c_{33} - v_3 = 9 - 7 \Rightarrow u_3 = 2$

$u_4 = c_{44} - v_4 = 8 - 4 \Rightarrow u_4 = 4$

$v_5 = c_{45} - u_4 = 6 - 4 = v_5 = 2$

	W_1	W_2	W_3	W_4	W_5	S	u_i
F_1	7 (30)	6 (10)	4	5	9	40	$u_1 = 0$
F_2	8	5 (20)	6 (10)	7	8	30	$u_2 = -1$
F_3	6	8	9 (5)	6 (15)	5	20	$u_3 = 2$
	5	7	7	8 (5)	6 (5)	10	$u_4 = 4$
	30	30	15	20	5		
v_j	$v_1 = 7$	$v_2 = 6$	$v_3 = 7$	$v_4 = 4$	$v_5 = 2$		

The minimization T.P. cost =

$7\times30+6\times10+5\times20+6\times10+9\times5$
$+6\times15+8\times5+6\times5 = 635$

Total allocated cell = 8 = $m+n-1$
$= 4+5-1$
$= 8$

hence the solⁿ is non degenerate

optimality test using modi method.

Allocatⁿ table is

	W_1	W_2	W_3	W_4	W_5	Supp.
F_1	7 (30)	6 (10)	9	5	9	40
F_2	8	5 (20)	6 (10)	7	8	30
F_3	6	8	9 (5)	6 (15)	5	20
F_4	5	7	7	8 (5)	6 (5)	10
Dem.	30	30	15	20	5	

Iteratⁿ 1 of optimality test.

(1) subs' $u_1 = 0$, we get

(2) find $d_{ij} = C_{ij} - (u_i + v_j)$

$d_{13} = C_{13} - (u_1 + v_3) = 4 - (0+7) = -3$

$d_{14} = 5 - (0+4) = 1$

$d_{15} = 9 - (0+2) = 7$

$d_{21} = 8 - (-1+7) = 2$

$d_{24} = 7 - (-1+4) = 4$

$d_{25} = 8 - (-1+2) = 7$

$d_{31} = 6 - (2+7) = -3$

$d_{32} = 8 - (2+6) = 0$

$d_{35} = 5 - (2+2) = 1$

$d_{41} = 5 - (4+7) = -6$

$d_{42} = 7 - (4+6) = -3$

$d_{43} = 7 - (4+7) = -4$

3)

	w_1	w_2	w_3	w_4	w_5	supp	u_i
f_1	7 (30) (−)	6 (10) (+)	4^{-3}	5^{1}	9^{7}	40	0
f_2	8^{2}	5 (20) (−)	6 (10) (+)	7^{4}	8^{7}	30	−1
f_3	6^{-3}	8^{0}	9 (5) (−)	6 (15) (+)	5^{1}	20	2
f_4	5^{-6} (+)	7^{-3}	7^{-4}	8 (5) (−)	6 (5)	10	4
dem	30	30	15	20	5		
v_j	7	6	7	4	2		

min(dij) = -6 , $F_4 W_1$

closed path = $F_4 W_1 \to F_4 W_4 \to F_3 W_4 \to$
$F_3 W_5 \to F_3 W_3 \to F_2 W_2 \to$
$F_1 W_2 \to F_1 W_1$

4) min allocated val among all -ve is 5

	W_1	W_2	W_3	W_4	W_5	S
F_1	7 (25)	6 (15)	4	5 (E)	9	40
F_2	8	5 (15)	6 (15)	7	8	30
F_3	6	8	9	6 (20)	5	20
F_4	5 (5)	7	7	8	6 (5)	10
D	30	30	15	20	5	

The result is degenerate since no of allocation = 7 < 8

hence quantity E is assign. to $F_1 W_4$, which has min. T.P cost = 5.

$d_{24} = 7-(-1+5) = 3$

$d_{25} = 8-(-1+8) = 1$

$d_{31} = 6-(1+7) = -2$

$d_{32} = 8-(1+6) = 1$

$d_{33} = 9-(1+7) = 1$

$d_{35} = 5-(1+8) = -4$

$d_{42} = 7-(-2+6) = 3$

$d_{43} = 7-(-2+7) = 2$

$d_{44} = 8-(-2+5) = 5$

min -ve value from all $d_{ij} = -4$

and draw path from $F_3 W_5$

closed path is $F_3 W_5 \to F_3 W_4 \to$

$F_1 W_4 \to F_1 W_1 \to F_4 W_1 \to F_4 W_5$

min alloc val among all -ve

point^n on closed path = 5

hence substract from -ve

adding +ve .

	W_1	W_2	W_3	W_4	W_5	Supp
F_1	7 (20)	6 (15)	4	5 (5)	9	40
F_2	8	5 (15)	6 (15)	7	8	30
F_3	6	8	9	6 (15)	5 (5)	20
F_4	5 (10)	7	7	8	6	10
Dem.	30	30	15	20	5	

Iteratⁿ 3

Substituting $u_1 = 0$,

$v_1 = 7 - 0 = 7$

$u_4 = 5 - 7 = -2$

$v_2 = 6 - 0 = 6$

$u_2 = 5 - 6 = -1$

$v_3 = 6 + 1 = 7$

$v_4 = 5 - 0 = 5$

$u_3 = 6 - 5 = 1$

$v_5 = 5 - 1 = 4$

	W_1	W_2	W_3	W_4	W_5	S	u_i
F_1	7 (20)	6 (15) (−)	4^{-3} (+)	5 (5)	9^{5}	40	0
F_2	8^{2}	5 (15) (+)	6 (15) (−)	7^{2}	8^{5}	30	−1
F_3	6^{-2}	8^{1}	9^{1}	6 (15)	5 (5)	20	1
F_4	5 (10)	7^{3}	7^{2}	8^{5}	6^{4}	10	−2
D	30	30	15	20	5		
v_j	7	6	7	5	4		

~~Min~~ → $d_{13} = 4 - (0 + 7) = -3$

$d_{15} = 9 - (0 + 4) = 5$

$d_{21} = 8 - (-1 + 7) = 2$

$d_{24} = 7-(-1+5) = 3$

$d_{25} = 8-(-1+4) = 5$

$d_{31} = 6-(1+7) = -2$

$d_{32} = 8-(1+6) = 1$

$d_{33} = 9-(1+7) = 1$

$d_{42} = 7-(-2+6) = 3$

$d_{43} = 7-(-2+7) = 2$

$d_{44} = 8-(-2+5) = 5$

$d_{45} = 6-(-2+4) = 4$

→ $d_{13} = [-3]$ minimum.

closed path = $F_1W_3 \to F_1W_2 \to F_2W_2 \to F_2W_3$

→ min alloc among all (−ve) = 15

	W_1	W_2	W_3	W_4	W_5	Supp.
F_1	7 (20)	6 (ε)	4 (15)	5 (5)	9	40
F_2	8	5 (30)	6	7	8	30
F_3	8	8	9	6 (15)	5 (5)	20
F_4	5 (10)	7	7	8	6	10
Demand	30	30	15	20	5	

The sol^n is degenerate ∵ 7 < 8

hence ε is assign to F_1W_2

at min cost of 6.

Iteration 4

substituting $u_1 = 0$,

$v_1 = 7 - 0 = 7$

$u_4 = 5 - 7 = -2$

$v_2 = 6 - 0 = 6$

$u_2 = 5 - 6 = -1$

$v_3 = 4 - 0 = 4$

$v_4 = 5 - 0 = 5$

$u_3 = 6 - 5 = 1$

$v_5 = 5 - 1 = 4$

	W_1	W_2	W_3	W_4	W_5	S	u_i
F_1	7^{20} (−)	$6^{[illegible]}$	4^{15}	5^{5} (+)	9^{5}	40	0
F_2	8^{2}	5^{30}	6^{3}	7^{3}	8^{5}	30	−1
F_3	6^{-2} (+)	8^{1}	9^{4}	6^{15} (−)	5^{5}	20	1
F_4	5^{10}	7^{3}	7^{5}	8^{5}	6^{4}	10	−2
D	30	30	15	20	5		
v_j	7	6	4	5	4		

→ $d_{15} = 9 - (0 + 4) = 5$

$d_{21} = 8 - (-1 + 7) = 2$

$d_{23} = 6 - (-1 + 4) = 3$

Iteration 5

~~simultaneous~~ by putting $u_1 = 0$

$v_1 = 7 - 0 = 7$

$u_3 = 6 - 7 = -1$

$v_5 = 5 + 1 = 6$

$u_4 = 5 - 7 = -2$

$v_2 = 6 - 0 = 6$

$u_2 = 5 - 6 = -1$

$v_3 = 4 - 0 = 4$

$v_4 = 5 - 0 = 5$

	W_1	W_2	W_3	W_4	W_5	S.	u_i
F_1	$7^{(5)}$	$6^{(\varepsilon)}$	$4^{(15)}$	$5^{(20)}$	9^{3}	40	0
F_2	8^{2}	$5^{(20)}$	6^{3}	7^{3}	8^{3}	30	-1
F_3	$6^{(15)}$	8^{3}	9^{6}	6^{2}	$5^{(5)}$	20.	-1
F_4	$5^{(10)}$	7^{3}	7^{5}	8^{5}	6^{2}	10	-2
D	30	30	15	20	5		
v_j	7	6	4	5	6		

$d_{15} = 9 - (0 + 6) = 3$

$d_{21} = 8 - (-1 + 7) = 2$

$d_{23} = 6 - (-1 + 4) = 3$

$d_{24} = 7-(-1+5) = 3$

$d_{25} = 8-(-1+6) = 3$

$d_{32} = 8-(-1+6) = 3$

$d_{33} = 9-(-1+4) = 6$

$d_{34} = 6-(-1+5) = 2$

$d_{42} = 7-(-2+6) = 3$

$d_{43} = 7-(-2+6) = 5$

$d_{44} = 8-(-2+5) = 5$

$d_{45} = 6-(-2+6) = 2$

since all $d_{ij} \geq 0$

so final o.s →

	w_1	w_2	w_3	w_4	w_5	Supp.
F_1	7 (5)	6	4 (15)	5 (20)	9	40
F_2	8	5 (30)	6	7	8	30
F_3	6 (15)	8	9	6	5 (5)	20
F_4	5 (10)	7	7	8	6	10
D.	30	30	15	20	5	

min Tot. TPC $= 7\times5 + 4\times15 + 5\times20 + 5\times30 + 6\times15 + 5\times5 + 5\times10 = 510$.

Least cost method –

	w_1	w_2	w_3	w_4	w_5	supply
F_1	~~7~~ 5	~~6~~	~~4~~ 15	~~5~~ 20	~~9~~	~~40~~ ~~25~~ ~~5~~ 0
F_2	8	~~5~~ 30	6	7	8	~~30~~ 0
F_3	6 15	8	9	~~6~~	~~5~~ 5	~~20~~ ~~15~~ 0
F_4	~~5~~ 10	7	~~7~~	8	~~6~~	~~10~~ 0
Dem.	~~30~~ ~~20~~ ~~15~~ 0	~~30~~ 0	~~15~~ 0	~~20~~ 0	~~5~~ 0	

→ The smallest TP cost is 4 in F_1W_1

The alloc to F_1W_3 = min (40, 15)
= 15

This satisf. entire dem. of W_3 & leaving
40 − 15 = 25. with F_1

→ The next TP cost is 5 in F_2W_2

the alloc to F_2W_2 = min (30, 30)
= 30

$W_2 = 30$, $F_2 = 0$.

→ The next TP is 5 in $F_1 W_4$
the alloc to $F_1 W_4 = \min(25, 20)$
$= 20$
$W_4 = 0$, $F_1 = 5$

→ The next TP is 5 in $F_2 W_1$
alloc of $F_2 W_1 = \min(10, 30) = 10$
$W_1 = 20$, $F_2 = 0$

→ The next TP is 5 in $F_3 W_5$
alloc of $F_3 W_5 = \min(20, 5) = 5$
$W_5 = 0$, $F_3 = 15$.

→ next min TP is 6 in $F_3 W_1$
the alloc to $F_3 W_1 = \min(15, 20)$
$= 15$
This exhaust $F_3 = 0$ & $W_1 = 5$

→ next min TP is 7 in $F_1 W_1$
alloc to $F_1 W_1 = \min(5, 5) = 5$
$F_1 = 0$, $W_1 = 0$.

Hence the IBFS =

	W_1	W_2	W_3	W_4	W_5	S
F_1	7 (5)	6 (ϵ)	4 (15)	5 (20)	9	40
F_2	8	5 (30)	6	7	8	30
F_3	6 (15)	8	9	6	5 (5)	20
F_4	5 (10)	7	7	8	6	10
D	30	30	15	20	5	

min TPC $= 7\times5 + 4\times15 + 5\times20 + 5\times30 + 6\times15 + 5\times5 + 5\times10$

$= 510$

here ~~no of~~ alloc^n = ~~7~~

~~no of~~ $m+n-1 = 4+5-1 = 8$

hence it is degenerate.

to resolve degeneracy, of we add ϵ into $F_1 W_2$ at cost 6.

<u>Optimal test</u>

Substituting $u_1 = 0$, we get.

$v_1 = 7 - 0 = 7$

$u_3 = 6 - 7 = -1$

$v_5 = 5 + 1 = 6$

$u_4 = 5 - 7 = -2$

$v_2 = 6 - 0 = 6$

$u_2 = 5 - 6 = -1$

$v_3 = 4 - 0 = 4$

$v_4 = 5 - 0 = 5$

$\rightarrow$ $d_{15} = 9 - (0 + 6) = 3$

$d_{21} = 8 - (-1 + 7) = 2$

$d_{23} = 6 - (-1 + 4) = 3$

$d_{24} = 7 - (-1 + 5) = 3$

$d_{25} = 8 - (-1 + 6) = 3$

$d_{32} = 8 - (-1 + 6) = 3$

$d_{33} = 9 - (-1 + 4) = 6$

$d_{34} = 6 - (-1 + 5) = 2$

$d_{42} = 7 - (-2 + 6) = 3$

$d_{43} = 7 - (-2 + 4) = 5$

$d_{44} = 8 - (-2 + 5) = 5$

$d_{45} = 6 - (-2 + 6)$
$= 2$

Since all $d_{ij} > 0$.

so the sol^n is optimal & unique

The min total transp. cost =

$7 \times 5 + 4 \times 15 + 5 \times 20 + 5 \times 30 +$

$6 \times 15 + 5 \times 5 + 5 \times 10$

$= 510.$

Python Code:

```
import numpy as np

def check_loop(p, row, column):
        p[row, column] = -1
        flag = 1
        while flag != 0:
                flag = 0
                if p.size != 0:
                        row = np.count_nonzero(p, axis=1)
                        f = 0
                        for index in range(len(row)):
                                if row[index] < 2:
                                        flag = 1
                                        p = np.delete(p, (index - f), axis=0)
                                        f += 1
                if p.size != 0:
                        e = 0
                        col = np.count_nonzero(p, axis=0)
                        for index in range(len(col)):
                                if col[index] < 2:
                                        flag = 1
                                        p = np.delete(p, (index - e), axis=1)
                                        e += 1
        if p.size != 0:
                return 0
        else:
                return 1

def max_allocation_row(non_zero_list):
        max_values = np.zeros(len(non_zero_list))
        for val in non_zero_list:
                max_values[val[0]] += 1.0
        return np.where(max_values == np.amax(max_values))


def modi(c_modi, a_modi, m_modi, n_modi):
        iteration_count = 1
        while True:
        print()
        print("Iteration - ", iteration_count)
        print("Start AL \n", a_modi)
        u = np.array([np.nan] * m_modi)
        v = np.array([np.nan] * n_modi)
        p = np.zeros((m_modi, n_modi))
        _x, _y = np.where(a_modi > 0)
        nonzero = list(zip(_x, _y))
        _x1, _y1 = np.where(a_modi == -1)
        if -1 in a_modi:
                nz_a = np.where(a_modi == -1)
                nonzero.append((min(nz_a[0]), min(nz_a[1])))
        f = max_allocation_row(nonzero)
```

```
u[f[0][0]] = 0
for i, j in nonzero:
    if i == f[0][0]:
        v[j] = c_modi[i, j] - u[i]
print("U = ", u)
print("V = ", v)
while any(np.isnan(u)) or any(np.isnan(v)):
    for i, j in nonzero:
        for j2 in range(0, len(v)):
            if j2 == j and not math.isnan(v[j])
                and math.isnan(u[i]):
                u[i] = c_modi[i, j] - v[j]
    for i, j in nonzero:
        for j2 in range(0, len(u)):
            if j2 == i and not math.isnan(u[i])
                and math.isnan(v[j]):
                v[j] = c_modi[i, j] - u[i]
    print("U = ", u)
    print("V = ", v)

# Finding P-matrix
for i in range(m_modi):
    for j in range(n_modi):
        if not nonzero.__contains__((i, j)):
            p[i, j] = c_modi[i, j] - u[i] - v[j]
print("P-matrix")
print(p)
# Stop condition
small_val = np.min(p)
if small_val >= 0:
    break
i, j = np.argwhere(p == small_val)[0]
start = (i, j)
print("Start : ", start)
# Finding cycle elements
t = np.copy(a_modi)
t[start] = 1
while True:
    _xs, _ys = np.nonzero(t)
    xcount, ycount = Counter(_xs), Counter(_ys)
    for x, count in xcount.items():
        if count <= 1:
            t[x, :] = 0
    for y, count in ycount.items():
        if count <= 1:
            t[:, y] = 0
    if all(x > 1 for x in xcount.values()) and
        all(y > 1 for y in ycount.values()):
        break

# Finding cycle chain order
def dist(x1, y1, x2, y2):
    if x1 == x2 or y1 == y2:
        return abs(x1 - x2) + abs(y1 - y2)
```

```
        else:
            return np.inf

    fringe = set(tuple(p) for p in np.argwhere(t != 0))
    alloc_modi = fringe
    size = len(fringe)
    path = [start]
    while len(path) < size:
        last = path[-1]
        if last in fringe:
            fringe.remove(last)
        next_val = min(fringe, key=lambda xy: dist(last[0],
            last[1], xy[0], xy[1]))
        path.append(next_val)

    # Improving solution on cycle elements
    neg = path[1::2]
    pos = path[::2]
    print("Negative Value:", neg)
    print("Positive Value:", pos)
    ql = []
    for row in neg:
        ql.append(a_modi[row[0], row[1]])
    q = int(min(ql))
    for row in neg:
        if a_modi[row[0], row[1]] == -1:
            a_modi[row[0], row[1]] = 0 - q
        else:
            a_modi[row[0], row[1]] -= q
    for row in pos:
        if a_modi[row[0], row[1]] == -1:
            a_modi[row[0], row[1]] = 0 + q
        else:
            a_modi[row[0], row[1]] += q
    x_al = np.nonzero(a_modi)[0]
    y_al = np.nonzero(a_modi)[1]
    for i in range(len(x_al)):
        alloc_modi.add((x_al[i], y_al[i]))
    print("Mid Allocation Table : \n", a_modi)

    _x2, _y2 = np.where(a_modi > 0)
    # alloc_modi = list(zip(_x2, _y2))
    no_alloc_modi = np.count_nonzero(a_modi)
    unalloc_modi = []
    for i1 in range(m_modi):
        for j1 in range(n_modi):
            if not (i1, j1) in alloc_modi:
                unalloc_modi.append((i1, j1))
    print(a_modi)
    no_loop_modi = []
    if no_alloc_modi == m_modi + n_modi - 1:
        print("Non Degeneracy")
    else:
        print("Degeneracy")
```

```
            print("Values of epsilon is -1")
            for i1 in unalloc_modi:
                if check_loop(a_modi.copy(), i1[0], i1[1]) == 1:
                    no_loop_modi.append(i1)
            min_epi_list = []
            for i1 in no_loop_modi:
                min_epi_list.append(c_modi[i1[0], i1[1]])
            min_epi = min(min_epi_list)
            ind = min_epi_list.index(min_epi)
            loc = no_loop_modi[ind]
            a_modi[loc[0], loc[1]] = -1
            print("END AL : \n", a_modi)
            print()
        iteration_count += 1
    return a_modi


def nwcr(cm_nwcr, m_nwcr, n_nwcr, s_nwcr, d_nwcr):
        c_nwcr = cm_nwcr.copy()
        a = np.zeros(c_nwcr.shape)
        total_cost_nwcr = 0
        no_alloc_nwcr = 0
        alloc_nwcr = []
        i = 0
        j = 0
        while (i < m_main) and (j < n_main):
            x = min(s_nwcr[i], d_nwcr[j])
            s_nwcr[i] = s_nwcr[i] - x
            d_nwcr[j] = d_nwcr[j] - x
            total_cost_nwcr = total_cost_nwcr + x * c_nwcr[i, j]
            no_alloc_nwcr += 1
            alloc_nwcr.append((i, j))
            a[i, j] = x
            if s_nwcr[i] < d_nwcr[j]:
                i = i + 1
            elif s_nwcr[i] > d_nwcr[j]:
                j = j + 1
            else:
                i = i + 1
                j = j + 1
        print("Total Cost: ", total_cost_nwcr)
        unalloc_nwcr = []
        for i1 in range(m_main):
            for j1 in range(n_main):
                if not (i1, j1) in alloc_nwcr:
                    unalloc_nwcr.append((i1, j1))
        print("Allocated Positions: ", alloc_nwcr)
        print("Unallocated Positions: ", unalloc_nwcr)
        print("Allocation Matrix: ")
        print(a)
        no_loop_nwcr = []
        if no_alloc_nwcr == m_nwcr + n_nwcr - 1:
            print("Non Degeneracy")
        else:
```

```
            print("Degeneracy")
            print("Values of epsilon is -1")
            for i1 in unalloc_nwcr:
                if check_loop(a.copy(), i1[0], i1[1]) == 1:
                    no_loop_nwcr.append(i1)
            min_epi_list = []
            for i1 in no_loop_nwcr:
                min_epi_list.append(cm_nwcr[i1[0], i1[1]])
            min_epi = min(min_epi_list)
            ind = min_epi_list.index(min_epi)
            loc = no_loop_nwcr[ind]
            a[loc[0], loc[1]] = -1
            print("Allocation Matrix After Converting Degeneracy
                to Non-Degeneracy is : ")
            print(a)
        optl = modi(cm_nwcr.copy(), a.copy(), m_nwcr, n_nwcr, )
        print('optimised Allocation Matrix: ')
        print(optl)
        for row in range(0, m_nwcr):
            for column in range(0, n_nwcr):
                if optl[row][column] < 0:
                    optl[row][column] = 0
        print("Total Optimal Cost = ", np.sum(optl * cm_nwcr))


def lcm(cm_lcm, m_lcm, n_lcm, s_lcm, d_lcm):
        c_lcm = cm_lcm.copy()
        total_cost_lcm = 0
        no_alloc_lcm = 0
        alloc_lcm = []
        a = np.zeros(c_lcm.shape)
        min_cost = np.amin(c_lcm)
        while min_cost != np.inf:
            indexes = np.where(c_lcm == min_cost)
            i = indexes[0][0]
            j = indexes[1][0]
            x = min(s[i], d_lcm[j])
            s_lcm[i] -= x
            d_lcm[j] -= x
            total_cost_lcm += (x * c_lcm[i, j])
            no_alloc_lcm += 1
            a[i, j] = x
            alloc.append((i, j))
            if s_lcm[i] < d_lcm[j]:
                x = 0
                while x < n_lcm:
                    c_lcm[i, x] = np.inf
                    x += 1
            elif s_lcm[i] > d_lcm[j]:
                y = 0
                while y < m_lcm:
                    c_lcm[y, j] = np.inf
                    y += 1
            else:
```

```
                    x = 0
                    while x < n_lcm:
                            c_lcm[i, x] = np.inf
                            x += 1
                    y = 0
                    while y < m_lcm:
                            c_lcm[y, j] = np.inf
                            y += 1
            min_cost = np.amin(c_lcm)
    print("Total Cost: ", total_cost_lcm)
    unalloc = []
    for i in range(m_lcm):
            for j in range(n_lcm):
                    if not (i, j) in alloc:
                            unalloc.append((i, j))
    print("List of Allocated Positions: ", alloc)
    print("List of Unallocated Positions: ", unalloc)
    print("Allocation Matrix: ")
    print(a)
    no_loop_lcm = []
    if no_alloc_lcm == m_lcm + n_lcm - 1:
            print("Non Degeneracy")
    else:
            print("Degeneracy")
            for i in unalloc:
                    g = check_loop(a.copy(), i[0], i[1])
                    if g == 1:
                            no_loop_lcm.append(i)
                    min_epi_list = []
            for i in no_loop_lcm:
                    min_epi_list.append(cm[i[0], i[1]])
            min_epi = min(min_epi_list)
            ind = min_epi_list.index(min_epi)
            loc = no_loop_lcm[ind]
            a[loc[0], loc[1]] = -1
            print("Allocation Matrix After Converting
                    Degeneracy to Non-Degeneracy is : ")
            print(a)
    optl = modi(cm_lcm.copy(), a.copy(), m_lcm, n_lcm, )
    print('optimised Allocation Matrix: ')
    print(optl)
    for row in range(0, m_lcm):
            for column in range(0, n_lcm):
                    if optl[row][column] < 0:
                            optl[row][column] = 0
    print("Total Optimal Cost = ", np.sum(optl * cm_lcm))
```

```python
if __name__ == '__main__':
        cm = np.array([[6.0, 4.0, 1.0, 5.0],
                [8.0, 9.0, 2.0, 7.0],
                [4.0, 3.0, 6.0, 2.0]])
        s = np.array([14.0, 12.0, 4.0])
        d = np.array([6.0, 10.0, 10.0, 4.0])
        c = cm.copy()
        print("The Cost Matrix is: ")
        print(c)
        print("The Supply is: ", s)
        print("The Demand is: ", d)
        m, n = c.shape
        print("No of Rows & No of Columns: (", m, ", ", n, ")")
        total_cost = 0
        no_alloc = 0
        total_demand = np.sum(d)
        total_supply = np.sum(s)
        alloc = []
        if total_demand == total_supply:
                print("It is a Balanced Transportation Problem")
        else:
                print("It is an UnBalanced Transportation Problem")
                if total_demand > total_supply:
                        new = np.array(np.zeros(n))
                        c = np.row_stack((c, new))
                        s = np.append(s, total_demand - total_supply)
                        m = m + 1
                else:
                        new = np.array(np.zeros(m))
                        c = np.column_stack((c, new))
                        d = np.append(d, total_supply - total_demand)
                        n = n + 1
                print("The New Balanced Cost Matrix is: ")
                print(c)
                print("The Supply is: ", s)
                print("The Demand is: ", d)
        print("Northwest corner method")
        nwcr(c_main.copy(), m, n, s_main.copy(), d_main.copy())
        print()
        print("Least Cost Method")
        lcm(c_main.copy(), m, n, s_main.copy(), d_main.copy())
        print()
```

Output :

```
"/home/arghya/My Work/Python/pythonProject1/venv/bin/python" "/home/arghya/My Work/Python/pythonProject1/assignment4problem1.py"
The Cost Matrix is:
[[7. 6. 4. 5. 9.]
 [8. 5. 6. 7. 8.]
 [6. 8. 9. 6. 5.]
 [5. 7. 7. 8. 6.]]
The Supply is:  [40. 30. 20. 10.]
The Demand is:  [30. 30. 15. 20.  5.]
No of Rows & No of Columns: ( 4 ,  5 )
It is a Balanced Transportation Problem
Northwest corner method
Total Cost:  635.0
Allocated Positions:  [(0, 0), (0, 1), (1, 1), (1, 2), (2, 2), (2, 3), (3, 3), (3, 4)]
Unallocated Positions:  [(0, 2), (0, 3), (0, 4), (1, 0), (1, 3), (1, 4), (2, 0), (2, 1), (2, 4), (3, 0), (3, 1), (3, 2)]
Allocation Matrix:
[[30. 10.  0.  0.  0.]
 [ 0. 20. 10.  0.  0.]
 [ 0.  0.  5. 15.  0.]
 [ 0.  0.  0.  5.  5.]]
Non Degeneracy

Iteration -  1
Start AL
 [[30. 10.  0.  0.  0.]
 [ 0. 20. 10.  0.  0.]
```

```
Start AL
 [[30. 10.  0.  0.  0.]
 [ 0. 20. 10.  0.  0.]
 [ 0.  0.  5. 15.  0.]
 [ 0.  0.  0.  5.  5.]]
U =  [ 0. nan nan nan]
V =  [ 7.  6. nan nan nan]
U =  [ 0. -1. nan nan]
V =  [ 7.  6.  7. nan nan]
U =  [ 0. -1.  2. nan]
V =  [ 7.  6.  7.  4. nan]
U =  [ 0. -1.  2.  4.]
V =  [7. 6. 7. 4. 2.]
P-matrix
[[ 0.  0. -3.  1.  7.]
 [ 2.  0.  0.  4.  7.]
 [-3.  0.  0.  0.  1.]
 [-6. -3. -4.  0.  0.]]
Start :  (3, 0)
Negative Value: [(0, 0), (1, 1), (2, 2), (3, 3)]
Positive Value: [(3, 0), (0, 1), (1, 2), (2, 3)]
Mid Allocation Table :
 [[25. 15.  0.  0.  0.]
 [ 0. 15. 15.  0.  0.]
 [ 0.  0.  0. 20.  0.]
```

```
Mid Allocation Table :
 [[25. 15.  0.  0.  0.]
 [ 0. 15. 15.  0.  0.]
 [ 0.  0.  0. 20.  0.]
 [ 5.  0.  0.  0.  5.]]
[[25. 15.  0.  0.  0.]
 [ 0. 15. 15.  0.  0.]
 [ 0.  0.  0. 20.  0.]
 [ 5.  0.  0.  0.  5.]]
Degeneracy
Values of epsilon is -1
END AL :
 [[25. 15.  0. -1.  0.]
 [ 0. 15. 15.  0.  0.]
 [ 0.  0.  0. 20.  0.]
 [ 5.  0.  0.  0.  5.]]



Iteration -  2
Start AL
 [[25. 15.  0. -1.  0.]
 [ 0. 15. 15.  0.  0.]
 [ 0.  0.  0. 20.  0.]
 [ 5.  0.  0.  0.  5.]]
U =  [ 0. nan nan nan]
```

```
U =  [ 0. nan nan nan]
V =  [ 7.  6. nan  5. nan]
U =  [ 0. -1.  1. -2.]
V =  [7. 6. 7. 5. 8.]
P-matrix
[[ 0.  0. -3.  0.  1.]
 [ 2.  0.  0.  3.  1.]
 [-2.  1.  1.  0. -4.]
 [ 0.  3.  2.  5.  0.]]
Start :  (2, 4)
Negative Value: [(3, 4), (0, 0), (2, 3)]
Positive Value: [(2, 4), (3, 0), (0, 3)]
Mid Allocation Table :
 [[20. 15.  0.  5.  0.]
 [ 0. 15. 15.  0.  0.]
 [ 0.  0.  0. 15.  5.]
 [10.  0.  0.  0.  0.]]
[[20. 15.  0.  5.  0.]
 [ 0. 15. 15.  0.  0.]
 [ 0.  0.  0. 15.  5.]
 [10.  0.  0.  0.  0.]]
Non Degeneracy

Iteration -  3
Start AL
```

```
Iteration -  3
Start AL
 [[20. 15.  0.  5.  0.]
 [ 0. 15. 15.  0.  0.]
 [ 0.  0.  0. 15.  5.]
 [10.  0.  0.  0.  0.]]
U =  [ 0. nan nan nan]
V =  [ 7.  6. nan  5. nan]
U =  [ 0. -1.  1. -2.]
V =  [7. 6. 7. 5. 4.]
P-matrix
[[ 0.  0. -3.  0.  5.]
 [ 2.  0.  0.  3.  5.]
 [-2.  1.  1.  0.  0.]
 [ 0.  3.  2.  5.  4.]]
Start :  (0, 2)
Negative Value: [(0, 1), (1, 2)]
Positive Value: [(0, 2), (1, 1)]
Mid Allocation Table :
 [[20.  0. 15.  5.  0.]
 [ 0. 30.  0.  0.  0.]
 [ 0.  0.  0. 15.  5.]
 [10.  0.  0.  0.  0.]]
[[20.  0. 15.  5.  0.]
 [ 0. 30.  0.  0.  0.]
```

```
[[20.  0. 15.  5.  0.]
 [ 0. 30.  0.  0.  0.]
 [ 0.  0.  0. 15.  5.]
 [10.  0.  0.  0.  0.]]
Degeneracy
Values of epsilon is -1
END AL :
 [[20. -1. 15.  5.  0.]
 [ 0. 30.  0.  0.  0.]
 [ 0.  0.  0. 15.  5.]
 [10.  0.  0.  0.  0.]]



Iteration -  4
Start AL
 [[20. -1. 15.  5.  0.]
 [ 0. 30.  0.  0.  0.]
 [ 0.  0.  0. 15.  5.]
 [10.  0.  0.  0.  0.]]
U =  [ 0. nan nan nan]
V =  [ 7.  6.  4.  5. nan]
U =  [ 0. -1.  1. -2.]
V =  [7. 6. 4. 5. 4.]
P-matrix
[[ 0.  0.  0.  0.  5.]
```

```
P-matrix
[[ 0.  0.  0.  0.  5.]
 [ 2.  0.  3.  3.  5.]
 [-2.  1.  4.  0.  0.]
 [ 0.  3.  5.  5.  4.]]
Start :  (2, 0)
Negative Value: [(0, 0), (2, 3)]
Positive Value: [(2, 0), (0, 3)]
Mid Allocation Table :
 [[ 5. -1. 15. 20.  0.]
 [ 0. 30.  0.  0.  0.]
 [15.  0.  0.  0.  5.]
 [10.  0.  0.  0.  0.]]
[[ 5. -1. 15. 20.  0.]
 [ 0. 30.  0.  0.  0.]
 [15.  0.  0.  0.  5.]
 [10.  0.  0.  0.  0.]]
Non Degeneracy

Iteration -  5
Start AL
 [[ 5. -1. 15. 20.  0.]
 [ 0. 30.  0.  0.  0.]
 [15.  0.  0.  0.  5.]
 [10.  0.  0.  0.  0.]]
```

```
Iteration -  5
Start AL
 [[ 5. -1. 15. 20.  0.]
 [ 0. 30.  0.  0.  0.]
 [15.  0.  0.  0.  5.]
 [10.  0.  0.  0.  0.]]
U =  [ 0. nan nan nan]
V =  [ 7.  6.  4.  5. nan]
U =  [ 0. -1. -1. -2.]
V =  [7. 6. 4. 5. 6.]
P-matrix
[[0. 0. 0. 0. 3.]
 [2. 0. 3. 3. 3.]
 [0. 3. 6. 2. 0.]
 [0. 3. 5. 5. 2.]]
optimised Allocation Matrix:
[[ 5. -1. 15. 20.  0.]
 [ 0. 30.  0.  0.  0.]
 [15.  0.  0.  0.  5.]
 [10.  0.  0.  0.  0.]]
Total Optimal Cost =  510.0

Least Cost Method
Total Cost:  510.0
List of Allocated Positions:  [(0, 2), (0, 3), (1, 1), (2, 4), (3, 0), (2, 0), (0, 0)]
```

```
Least Cost Method
Total Cost:  510.0
List of Allocated Positions:  [(0, 2), (0, 3), (1, 1), (2, 4), (3, 0), (2, 0), (0, 0)]
List of Unallocated Positions:  [(0, 1), (0, 4), (1, 0), (1, 2), (1, 3), (1, 4), (2, 1), (2, 2), (2, 3), (3, 1), (3, 2), (3, 3), (3, 4)]
Allocation Matrix:
[[ 5.  0. 15. 20.  0.]
 [ 0. 30.  0.  0.  0.]
 [15.  0.  0.  0.  5.]
 [10.  0.  0.  0.  0.]]
Degeneracy
Values of epsilon is -1
Allocation Matrix After Converting Degeneracy to Non-Degeneracy is :
[[ 5. -1. 15. 20.  0.]
 [ 0. 30.  0.  0.  0.]
 [15.  0.  0.  0.  5.]
 [10.  0.  0.  0.  0.]]

Iteration -  1
Start AL
 [[ 5. -1. 15. 20.  0.]
 [ 0. 30.  0.  0.  0.]
 [15.  0.  0.  0.  5.]
 [10.  0.  0.  0.  0.]]
U =  [ 0. nan nan nan]
V =  [ 7.  6.  4.  5. nan]
```


```
Iteration -  1
Start AL
 [[ 5. -1. 15. 20.  0.]
 [ 0. 30.  0.  0.  0.]
 [15.  0.  0.  0.  5.]
 [10.  0.  0.  0.  0.]]
U =  [ 0. nan nan nan]
V =  [ 7.  6.  4.  5. nan]
```


```
U =  [ 0. -1. -1. -2.]
V =  [7. 6. 4. 5. 6.]
P-matrix
[[0. 0. 0. 0. 3.]
 [2. 0. 3. 3. 3.]
 [0. 3. 6. 2. 0.]
 [0. 3. 5. 5. 2.]]
optimised Allocation Matrix:
[[ 5. -1. 15. 20.  0.]
 [ 0. 30.  0.  0.  0.]
 [15.  0.  0.  0.  5.]
 [10.  0.  0.  0.  0.]]
Total Optimal Cost =  510.0



Process finished with exit code 0
```