Graph → Simple → degree is 0 or upto n-1

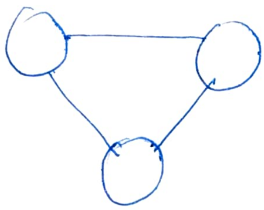
Multi → ~~d~~ maximum degree is ∞

Complete graph → If it follows mess topology

no. of edges = $\frac{n(n-1)}{2}$

forest → If there is no connection, deployed individually

$n=3$



No. of complete graph = $2^{\frac{n(n-1)}{2}}$

Spanning tree → A subgraph S of given graph graph is called Spanning tree if and only if

① S should contain all the vertices of G

② S should contain $(V-1)$ edges where V is the number of vertices

③ S cannot contain cycle.

Minimum Cost Spanning Tree :- To find the minimum cost spanning tree for a given graph there are two algorithm

① Kruskal's

② Prim's

★ Kruskal's

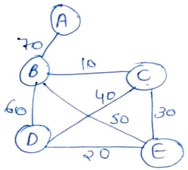
MST_Kruskal (G, W)

1. A = ϕ
2. For each vertex $V \in G.V$
3. MAKE_SET (V)
4. Sort the edges of G.E into non-decreasing order by weight W
5. For each edge $(u, v) \in G.E$ taken in non-decreasing order by weight

6. If FIND-SET(u) ≠ FIND-SET(v)
7. A = A ∪ {(u,v)}
8. UNION (u,v)
9. return (A)

OR

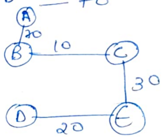
① Sort all the edges in increasing order by weight w.
② Take minimum weight edge and add to MST.
③ Take next minimum weight edge and add to MST.
④ Take minimum next weight edge and add to MST if cycle not form.
⑤ Repeat step ④.



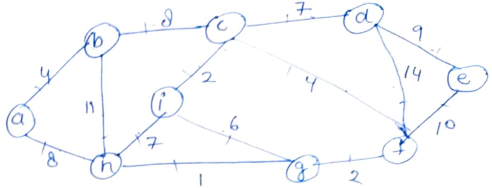
Step-1 ~~10 20 30 40 50 60 70~~

~~Step 2~~
B — C — 10
D – E — 20
C – E — 30
C – D — 40
B – E — 50
B – D — 60
A – B — 70

Step-2

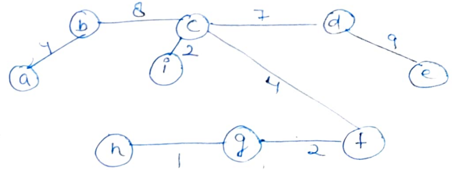


Step-3 The cost of minimum spanning tree is 130



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

g.



The cost of minimum spanning tree = 37

Note – If the weights of more than one edges are same then more than one MST is possible but cost should be same.

T.C. of kruskal algorithm

Best case → $O(E) + (V-1) \times \log E + O(E+V)$ — Using min heap

- $O(E)$ → build heap
- $(V-1) \times \log E$ → extract the vertices
- $O(E+V)$ → cycle checking

Worst case ~~$O(E) + (V-1) \times l$~~

$O(E) + E \times \log E + O(E+V)$

~~$(V-1) \times \log$~~

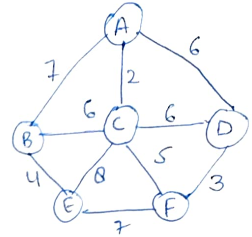
Best case → $(V-1) \times O(1) + O(E+V)$ } If array is sorted
Worst case → $E \times O(1) + O(E+V)$

If we are using selection sort instead of min heap.

Best case → $(V-1) \times O(E) + O(E+V)$
Worst case → $E \times O(E) + O(E+V)$

MST_PRIM(G, W, r)

1) For each $u \in G.V$
2) $u.key = \infty$
3) $u.\pi = NIL$
4) $r.key = 0$
5) $Q = G.V$
6) while $Q \neq \phi$
7) $u = \text{Extract-Min}(Q)$
8) For each $v \in G.Adj[u]$
9) if $v \in Q$ and $w(u,v) < v.key$
10) $v.\pi = u$
11) $v.key = w(u,v)$



	A	B	C	D	E	F
	0	∞ N	∞ N	∞ N	∞ N	∞ N
A		7 A	2 A	6 A	∞ N	∞ N
C		6 C		6 A	8 C	5 C
F		6 C		3 F	7 F	
D		6 C			7 F	
B					4 B	
G						

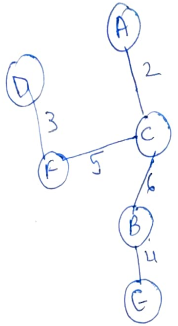
$u = \{A\}$
$v = \{B, C, D\}$
$w(A,B)$
$7 < \infty$
$w(A,D)$
$6 < \infty$
$w(A,C)$
$2 < \infty$

$u = \{C\}$
$V = \{B, D, E, F\}$
$W(C,B)$
$6 < 7$
$W(C,D)$
$6 < 6$
$w(C,F)$
$5 < \infty$

$u = \{D\}$

$u = \{B\}$
$v = \{E\}$
$W(B,E)$
$4 < 7$

$u = \{F\}$ $v = \{E, D\}$
$w(F,E)$
$7 < 8$



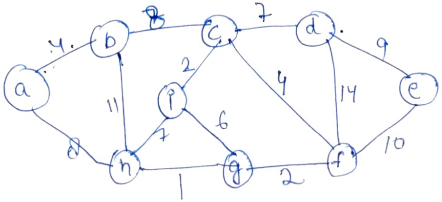
Step-1 Take a root node of a graph and find the adjacent of that node and take minimum from them.

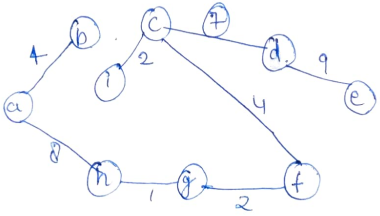
Step-2 find the adjacent of new node and take minimum from them & previous.

Step-3 find the adjacent of new node and take minimum from them & previous if cycle not form.

Step-4 Repeat step-3

Ex-2





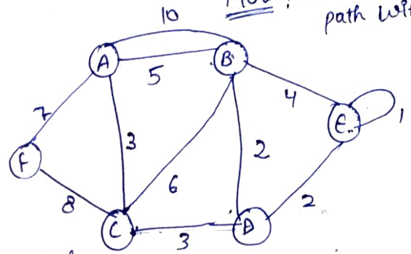
Kruskal's

Note: Prim's ~~Kruskal's~~ is always a connected graph in b/w. but Kruskal's ~~prims~~
check: may or may not in Intermediate path.

T.C of Prims = $O(E+V)\log V$ (Binary heap + adjacency list)

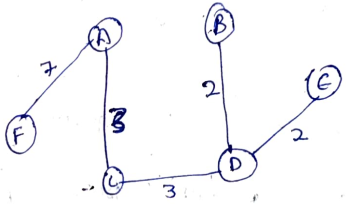
Data Structure used in Prims = Binary heap + adjacency list

Note: first remove cycle & then parallel path with maximum value



By Kruskals

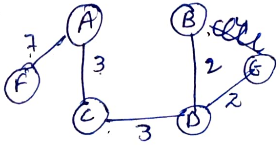
E – E ⇒ 1
D – E ⇒ 2
B – D ⇒ 2
C – D ⇒ 3
A – C ⇒ 3
B – E ⇒ 4
A – B ⇒ 5
B – C ⇒ 6
A – F ⇒ 7
C – F ⇒ 8
A – B ⇒ 10



MST cost = 17.

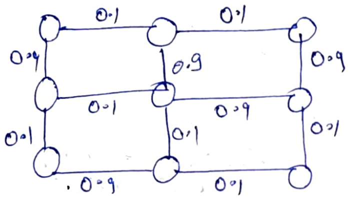
By prims

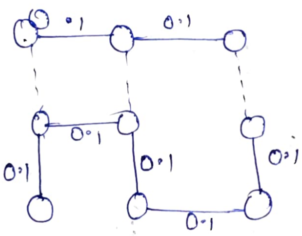
	A	B	C	D	E	F
A	0	∞ N	∞ N	∞ N	∞ N	∞ N
C		5 A	3 A	∞ N	∞ N	7 A
D		5 A		3 C	∞ N	7 A
E		2 D			2 D	7 A
		2 D			1 E	7 A



MST cost = 17

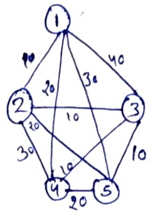
How many minimum number of MST are possible?



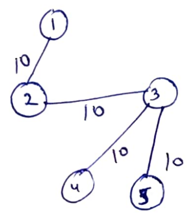


No. of MST will be 3 because we connect the graph in 3 possible ways by edge having weight 0.9

Total No. of spanning trees = n^{n-2}.



	1	2	3	4	5
1	0	10	40	10	30
2	10	0	10	30	20
3	40	10	0	10	10
4	10	30	10	0	20
5	30	20	10	20	0



Bellman Ford → to find a single-source shortest path.

```
Bellman-Ford (G, W, S)
Initialise-single-source(G, S)
    for i = 1 to |G.V|-1
    for each edge (u, v) ∈ G.E
        Relax (u, v, w)
    for each edge (u, v) ∈ G.E.
      If v.d > u.d + w(u, v)
        return FALSE
    return TRUE

Initialise-Single-source (G, S)
    for each vertex v ∈ G.V
        v.d = ∞
        v.π = NIL
        s.d = 0

Relax (u, v, w)
    If v.d > u.d + w(u, v)
        v.d = u.d + w(u, v)
        v.π = u
```

⇒ If a graph contains all edges with +ve edge weight, then dijkstra algorithm give optimal soln. but if any edge have -ve edge weight, then dijkstra algorithm ~~give~~ may or may not give optimal solution

⇒ If a graph contain negative weight cycle, then bellman ford algorithm cannot terminate.

⇒ In dijkstra algorithm, we have to call a relax function, single time for all the edges.

⇒ In bellman ford algorithm, we have to call a relax function (V-1) times for all the edges.

Time Complexity

⇒ When the graph is complete graph, time complexity will be.

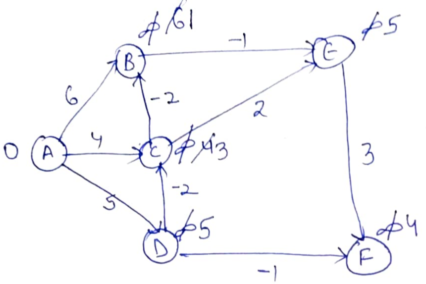
$$\frac{V(V-1)}{2} * (V) = V^3.$$

⇒ Time complexity

$\Rightarrow E * (V-1)$

$\Rightarrow E * V$

$\Rightarrow V^2$



(A-B) (A-C) (A-D) (B-E) (C-E) (D-C) (D-F) (E-F) (C-B)

Step-1

$V = 6$

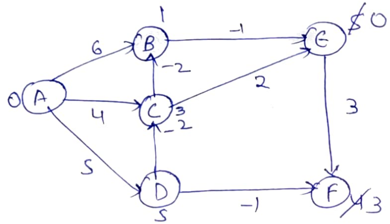
Relax = 5

$\overset{\infty}{v.d} \geq \overset{0}{u.d} + \overset{6}{w(u,v)}$

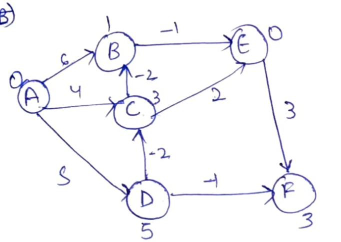
$v.d = 0+6$

C-E $\quad \overset{5}{v.d} > \overset{4}{u.d} + \overset{2}{w(u,v)}$ X

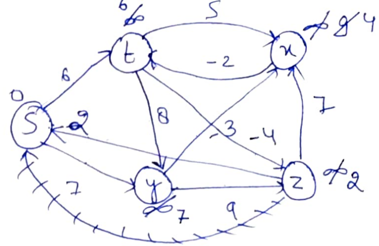
Step 2



~~(A-B)~~

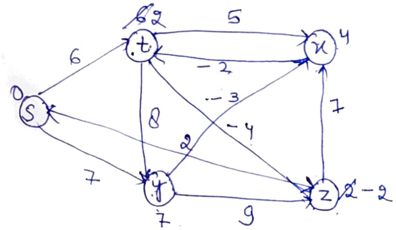


1 > 0+6
5 > 1-1
5 > 0.
0 > 3+2
3 > 5-2
3 > 3
4 > 5-1
4 > 3+0
1 > 3-2
1 > 1

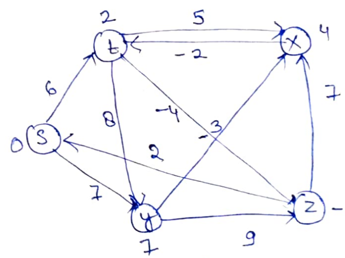


s-t, x-t, s-y, t-y, t-z, y-z, z-x
t-x, y-x, z-s

$\infty > 0+6$
$\infty > 0+7$
$7 > 6+8$
$\infty > 9$
$9 > 11$
$9 > 7-3$
4
$0 > 4.$

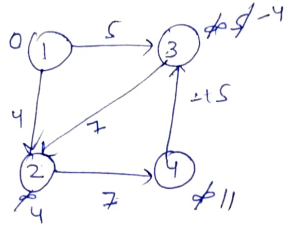


6 > 6	2 > 6
6 > 2	2 > 2
7 > 7	7 > 7
7 > 10	7 > 10
2 > -2.	4 > 5
4 > 5	4 > 7
4 > 7	4
4 > 4	0
0 > 0	



Ans.

Ex-3

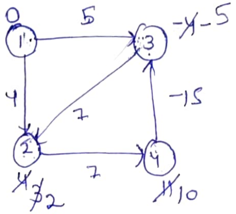


1-3, 1-2, 2-4, 3-2, 4-3

$\infty > 5$

∞

Step 2-3 -4



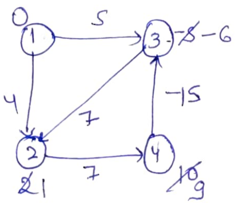
4 >

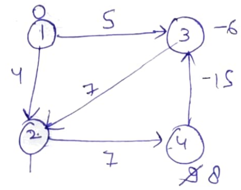
5 > 11 - 15

5 > -4

-4 > 5	-4 > 5	-5 > 5
4 > 4	3 > 4	3 > 4
11 > 11	11 > 10	10 > 10
4 > 3	3 > 3	3 > 2
-4 > -4	-4 > -5	-5 > -5

Step-5





-5 > 5

2 > 4   1 > 4

10 > 9   9 > 8

-5 > -6

2 > 1

⇒ It is not possible to find shortest path from Bellman ford as it is forming negative cycle.

Fractional Knapsack → he/she can take fractional part of any item to earn maximum profit.

```
FKnapSack()
{
   for i = 1 to n
   Compute Pi/Wi                     → O(n)
   Sort objects in non increasing order → O(n log n)
   For i = 1 to n from sorted list
    if (m>0 && Wi <= m)              → O(n)
      m = m - Wi
      P = P + Pi
    else
       break;
    if (m>0)                         → O(1)
      P = P + Pi (m/Wi)
```

m = capacity of knapsack

W = weight of items

P = Profit of items.

It is an application of greedy approach.

Total time complexity

$T.C. = O(n) + O(n \log n) + O(n) + O(1)$

$T.C. = O(n \log n)$

Ex-1

I	1	2	3	4	5	6	7
P	5	10	15	7	8	9	4
W	1	3	5	4	1	3	2
Pi/Wi	5	3.33	3	1.75	8	3	2

m = 15
n = 7

Approach 1

$I_1 \Rightarrow P = 0 + 5$

$m = m - W_i$
$= 15 - 1 = 14$ kg

$I_2 \Rightarrow P = 5 + 10 = 15$ ~~kg~~.

$m = m - W_i$
$14 - 3 = 11$ kg

$I_3 \Rightarrow P = 15 + 15 = 30$
$m = 11 - 5 = 6$

$I_4 \Rightarrow P = 30 + 7$
$= 37$
$m = 6 - 4 = 2$

$I_7 \Rightarrow P = 37 + 4 = 41$
$w = 2 - 2 = 0$

Approach-2

$I_1 \Rightarrow P = 0 + 5 = 5$
$m = 15 - 1 = 14$

$I_5 \Rightarrow P = 5 + 8 = 13$
$m = 14 - 1 = 13$

$I_7 \Rightarrow P = 13 + 4 = 17$
$m = 13 - 2 = 11$

$I_2 \Rightarrow P = 17 + 10 = 27$
$m = 11 - 3 = 8$

$I_6 \Rightarrow P = 27 + 9 = 36$
$m = 8 - 3 = 5$

$I_4 \Rightarrow P = 36 + 7 = 43$
$m = 5 - 4 = 1$

Approach-3

I	P	W	P_i/W_i
I5	8	15-1=14	8
I_1	8+5=13	14-1=13	5
I_2	13+10=23	13-3=10	
I_3	23+15=38	10-5=5	3.33
I6	38+9=47	5-3=2	3
I7	47+4=51	2-2=0	3
			2

Application

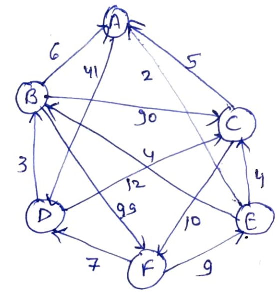
⇒ Portfolio optimization
⇒ Power Management

*** Dijkstra's Algorithm

Dijkstra (G, W, S)
(1) Initialise-single-source (G, S)
(2) $S = \phi$
(3) $Q = G.V$
(4) while $Q \neq \phi$
(5) u = Extract-min (Q)
(6) $S = S \cup \{u\}$
(7) For each vertex $v \in G.Adj[u]$
(8) Relax (u, v, w)

Initialise-single-source (G, S)
1) for each vertex $V \in G.V$
2) $v.d = \infty$
3) $v.\pi$ = NIL
4) $s.d = 0$

Relax (u, v, w)
if $v.d > u.d + w(u,v)$
$v.d = u.d + w(u,v)$
$v.\pi = u$



Q Print the sequence of vertices identified Dijkstra's ~~the~~ source B

Q What will be the cost of shortest path from B to E

	A	B	C	D	E	F
	∞ N	0 N	∞ N	∞ N	∞ N	∞ N
B	6 B		90 B	∞ N	∞ N	99 B
A			90 B	47 A	∞ N	99 B
D			59 D		∞ N	99 B
C					∞ N	69 E
F					78 E	

u = {D}
v = {C}
90 > 47 + 12
90 > 59

u = {C}
v = {F}
99 > 59 + 10
∞ > 69 + 9
78

Sequence = B A D C F E

Cost of shortest path from B to E = 78

T.C. ⇒ (E+V) log V
Worst case TC = V^2

Job Sequencing with deadline

⇒ The sequencing of jobs on a single processor with deadline constraints is called job sequencing with deadline.

* You have a set of jobs.
* Each job has defined deadline and some profit.
* The profit of job is given when that job is completed within its deadline.
* Only one processor is available for processing all the jobs.
* Processor takes one unit of time to complete a job.

Steps to find the solution

⇒ Sort all the jobs in decreasing order based on their profit

⇒ Check the value of maximum deadline and draw giant chart where maximum time on giant chart is the value of maximum deadline.

⇒ Pick up the jobs one by one and put the job on giant chart as far as possible from 0 (start point of giant chart) ensuring that the jobs get completed before deadline.

Jobs	J_1	J_2	J_3	J_4	J_5	J_6
Deadline	5	3	3	2	4	2
Profit	200	180	190	300	120	100

Sort.

Jobs	J_4	J_1	J_3	J_2	J_5	J_6
Deadline	2	5	3	3	4	2
Profit	300	200	190	180	120	100

Giant Chart.

J_2	J_4	J_3	J_5	J_1

0 1 2 3 4 5

J_6 cannot be placed due to no availability of processor.

Q1 Write the optimal schedule that gives maximum profit?

Q2 Are all the jobs completed in the optimal schedule?

Q3 What is the maximum earned profit?

Ans 1. J_2 J_4 J_3 J_5 J_1

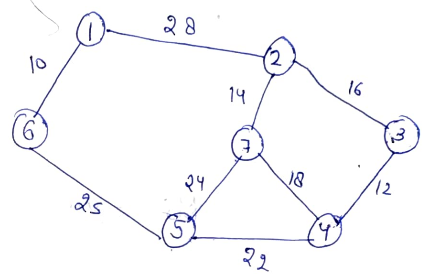
Ans 2 No

Ans 3. 990.

T.C. of job sequencing with deadline $\Rightarrow O(n \log n)$

Assignment - 2

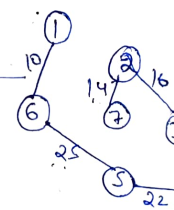
Q. What do you mean by minimum spanning tree? Write & explain algorithm for minimal spanning tree for a given graph using Prim's and Kruskal's both with source node 1.



Prim's

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

∞ = 0 + 10



MST = 99

Kruskal's

1 — 6 ⇒ 10

3 — 4 ⇒ 12

2 — 7 ⇒ 14

2 — 3 ⇒ 16

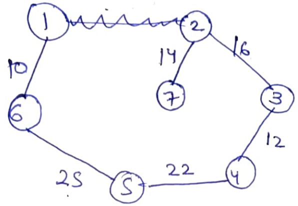
4 — 7 ⇒ 18

4 — 5 ⇒ 22

5 — 7 ⇒ 24

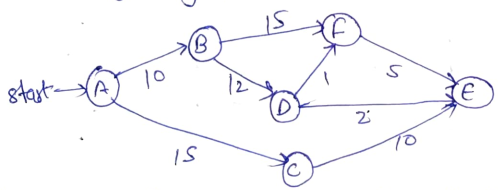
5 — 6 ⇒ 25

1 — 2 ⇒ 28



MST = 99.

Q. Write an algorithm to find single source shortest path for a given graph.



Q1. Show the sequence of vertices identified Dijkstra's source A.

A B C D F E

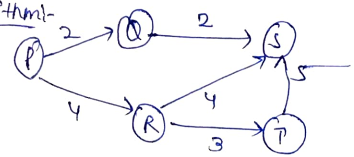
Q2 What will be the cost of shortest path from A to E
24

Q3 what will be the shortest path from A to E
A → B → D → E

	A	B	C	D	E	F
	0 N	∞ N	∞ N	∞ N	∞ N	∞ N
A		10 A	15 A	∞ N	∞ N	∞ N
B			15 A	22 B	∞ N	25 B
C				22 B	25 C	25 B
D					24 D	23 D
F					24 D	
E					24 E	

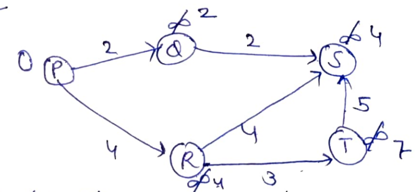
Q3. Write an algorithm to find single source shortest for a given graph.

Bellman ford Algorithm:-

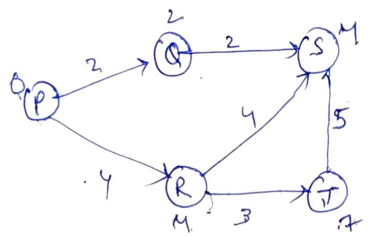


Soln

Step-1



(P-R), (P-Q), (Q-S), (R-S), (R-T), (T-S)



Q1 What will be the cost of shortest path from P to S

Ans 4

Q2. What will the shortest path from P to S

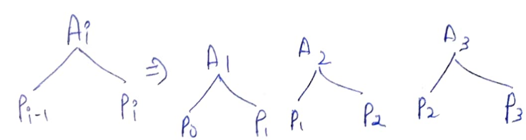
Ans P → Q - S

Matrix Chain Multiplication (MCM)

1) $n = P.length - 1$
2) let $m[1 \dots n, 1 \dots n]$ and $s[1 \dots n-1, 2 \dots n]$ be new tables
3) for $i = 1$ to n
4) $m[i,i] = 0$
5) For $l = 2$ to n
6) for $i = 1$ to $n - l + 1$
7) $j = i + l - 1$
8) $m[i,j] = \infty$
9) for $k = i$ to $j-1$
10) $q = m[i,k] + m[k+1,j] + P_{i-1} P_k P_j$
11) If $q < m[i,j]$
12) $m[i,j] = q$
13) $s[i,j] = k$
14) return m & s

Recurrence relation of MCM

$$m[i,j] = \begin{cases} 0 & \text{if } i == j \\ \min\limits_{i \le k < j} (m[i,k] + m[k+1,j] + P_{i-1} P_k P_j) \end{cases}$$



A_1	A_2	A_3	A_4
5×4	4×6	6×2	2×7

$P_0 = 5$ $P_1 = 4$ $P_2 = 6$ $P_3 = 2$ $P_4 = 7$

	A_1	A_2	A_3	A_4
	30x35	35x15	15x5	5x10

m

	1	2	3	4
1	0	120	88	158
2		0	48	104
3			0	84
4				0

→ This number of multiplication to compute solution.

$m[3,4] = M[3,3] +$
$m[4,4] + P_2 P_3 P_4$

	1	2	3	4
1	0	1	1	3
2		0	2	3
3			0	3
4				0

S

$m[1,3]$

$i=1$

$j=3$

$i \leq k < j$

$k = 1, 2$

$k=1 \Rightarrow m[1,1] + m[2,3] + P_0 P_1 P_3$
$= 0 + 48 + 5 \times 4 \times 2$
$= 88$

$k=2 \Rightarrow m(1,2) + m(3,3) + P_0 P_2 P_3$
$= 120 + 0 + 5 \times 6 \times 2 = 180$

min

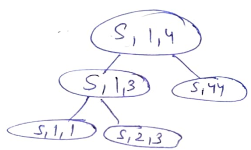
$m[2,4]$

$i = 2$
$j = 4$

$k=2 \Rightarrow m(2,2) + m(3,4) + P_1 P_2 P_4$
$= 0 + 84 + 4 \times 6 \times 7 = 252$

$k = 3$ $\quad m[2,3] + m(4,4) + P_1 P_3 P_4$
$= 48 + 0 + 4 \times 2 \times 7 = 104$

1. Print_optimal_Parens (S, i, j)
2. if i = = j
3. Print "Ai"
4. else Print "("
5. Print_optimal_Parens(S, i, S[i,j])
6. Print_optimal_Parens(S, S[i,j]+1, j)
7. Print ")"



$((A_1)(A_2A_3)(A_4))$

Time complexity of matrix chain multiplication $\frac{n^2}{2} * n$
$= O(n^3)$

Longest Common Sequence (LCS)

⇒ It is a sequence of any given sequence in which 0 or more symbols are left out

⇒ Ordering is necessary.

$X = \{A, B, C, D, A, A\}$

$SS_1 = \{A\}$ ✓
$SS_2 = \{A, B\}$ ✓
$SS_3 = \{A, A\}$ ✓
$SS_4 = \{A, A, D\}$ X

} Sequence.

$CS_1 = \{A, C\}$ ✓
$CS_2 = \{A, D\}$ ✓
$CS_3 = \{A, C, D, E\}$ X

} Common sub-sequence

Time complexity using bruteforce method = $O(2^n)$.

LCS_length (X, Y)

1) m = X.length
2) n = Y.length
3) Let b[1 --- m, 1-n] and c[0---m, 0--n] be new tables
4) for i = 0 to m
5) c[i, 0] = 0
6) for j = 0 to n
7) c[0, j] = 0
8) for i = 1 to m
9) for j = 1 to n
10) if $x_i == y_j$
11) c[i, j] = c[i-1, j-1] + 1
12) b[i, j] = "↖"
13) else if c[i-1, j] ≥ c[i, j-1]
14) c[i, j] = c[i-1, j]
15) b[i, j] = "↑"
16) else c[i, j] = c[i, j-1]
17) b[i, j] = "←"
18) return c & b

$X = \{A, B, C, B, D, A, B\}$

$Y = \{B, D, C, A, B, A\}$

b

	1	2	3	4	5	6
1	←	←	←	↖	←	↖
2	↖	←	←	↑	↖	←
3	↑	↑	↑	←	↑	↑
4	↖	↑	↑	↑	↖	←
5	↑	↖	↖	↑	↑	↑
6	↑	↑	↑	↖	↑	↖
7	←	↑	↑	↑	[illegible]	↑

c

	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	0	0	1	1	1
2	0	1	1	1	1	2	2
3	0	1	1	2	← 2	2	2
4	0	1	1	2	2	↖ 3	3
5	0	1	2	2	2	3 ↑	3
6	0	1	2	2	3	3	4 ↖
7	0	1	2	2	3	4	4 ↑

c[1, 4] = c[0, 3] + 1.
0 + 1
= 1

c[2, 1] = c[1, 0] + 1
0 + 1

c[2, 2] = 1
c[1, 2] > c[2, 1]
0 > 1
c[2, 3] = c[1, 3] [2, 2]

Print_LCS (b, X, i, j)

(1) if i == 0 or j == 0
(2) return
(3) if b[i, j] == "↖"
(4) Print_LCS [b, X, i-1, j-1)
(5) Print X_i
(6) else if b[i, j] == "↑"
(7) Print_LCS(b, X, i-1, j)
(8) else
Print_LCS (b, X, i, j-1)

c[1, 5] = max(c[0, 5], c[1, 4])

Ex-2 X = {A, B, C, A}
Y = {A, A, A, C}

Ans = 2
AC or AA

Ex-1. X = {A, B, C, B, D, A B}
Y = {B, D, C, A, B, A}

	1	2	3	4	5	6
1	↑	↑	↑	↖	←	↖
2	↖	←	←	↑	↖	←
3	↑	↑	↖			
4						
5						
6						
7						

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

C(2,1) = (1,0) + 1
C(2,2) = C(1,2)   C(2,1)
C(2,3) = C(1,3)   C(2,2)
C(2,4) = C(1,4) ≥ C(2,3)
C(2,5) = C(1,4)
C(2,6) = C(1,6)   (2,5)

C(3,1) = (2,0) ≥ (3,0)
C(3,2) = (2,2) ~ (3,1) L
C(3,3) = (2,2)+1
C(3,4) = C
C(3,5) =
C(3,6) =

Ex-3 X = {A, A, B}
Y = {A, B}

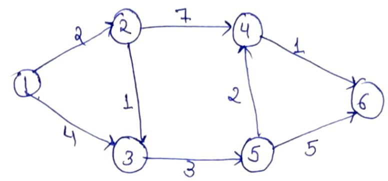
	1	2
1	↖	←
2	↖	↑
3	↑	↖

	0	1	2
0	0	0	0
1	0	1	1
2	0	1 ↖	1
3	0	1	2 ↖

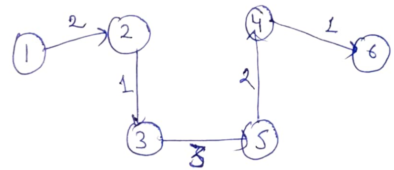
Ans (AB)

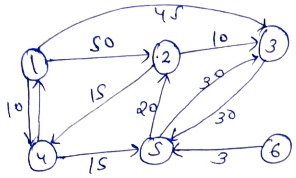
C(1,1) = 0,0 +1
C(1,2) = C(0,2)   (1,1)
C(2,2) = C(1,2)   (2,1)

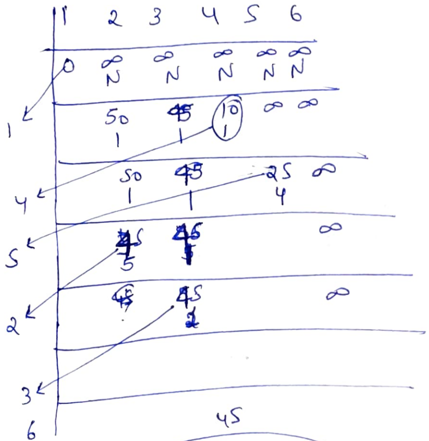
Dijkstra Algorithm

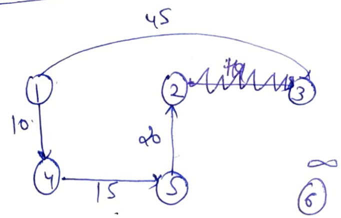


	1	2	3	4	5	6
	0	∞ N	∞ N	∞ N	∞ N	∞ N
1		2 1	4 1	∞ N	∞ N	∞ N
2			3 2	9 2	∞ N	∞ N
3				9 2	6 3	∞ N
5				8 5		11 5
4						9 4
6						





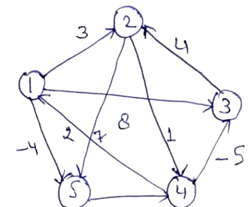




FLOYD-WARSHALL (W)

1) $n = W.rows$
2) $D^0 = W$
3) for $k = 1$ to n
4) let $D^k = d_{ij}^{k}$ be a new $n \times n$ matrix
5) for $i = 1$ to n
6) for $j = 1$ to n
7) $d_{ij}^{k} = \min(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1})$
8) return $D^{(n)}$

T.C $= O(n^3)$



Rough Work

8 $3+\infty$
∞, $3+1$
3 (1,2), (1,3) (3,2)
3, 8+4 12
∞ 8 ∞, 2+ (2,1) (2,3) (3,1)
-4
12 (1,4)+(4,2)
3 4+(-1)
3

$D^0 =$

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

$$D_1 = \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 3 & 8 & \infty & -4 \\ 2 & \infty & 0 & \infty & 1 & 7 \\ 3 & \infty & 4 & 0 & \infty & \infty \\ 4 & 2 & 5 & -5 & 0 & -2 \\ 5 & \infty & \infty & \infty & 6 & 0 \end{array}$$

$$D^2 = \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 3 & 8 & 4 & -4 \\ 2 & \infty & 0 & \infty & 1 & 7 \\ 3 & \infty & 4 & 0 & 5 & 11 \\ 4 & 2 & 5 & -5 & 0 & -2 \\ 5 & \infty & \infty & \infty & 6 & 0 \end{array}$$

$$D_3 = \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 3 & 8 & 4 & -4 \\ 2 & \infty & 0 & \infty & 1 & 7 \\ 3 & \infty & 4 & 0 & 5 & 11 \\ 4 & 2 & -1 & -5 & 0 & -2 \\ 5 & \infty & \infty & \infty & 6 & 0 \end{array}$$

$$D^4 = \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 3 & -1 & 4 & -4 \\ 2 & 3 & 0 & -4 & 1 & -1 \\ 3 & 7 & 4 & 0 & 5 & 3 \\ 4 & 2 & -1 & -5 & 0 & -2 \\ 5 & 8 & 5 & 1 & 6 & 0 \end{array}$$

$$D5 = \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 1 & -3 & 2 & -4 \\ 2 & 3 & 0 & -4 & 1 & -1 \\ 3 & 7 & 4 & 0 & 5 & 3 \\ 4 & 2 & -1 & -5 & 0 & -2 \\ 5 & 8 & 5 & 1 & 6 & 0 \end{array}$$