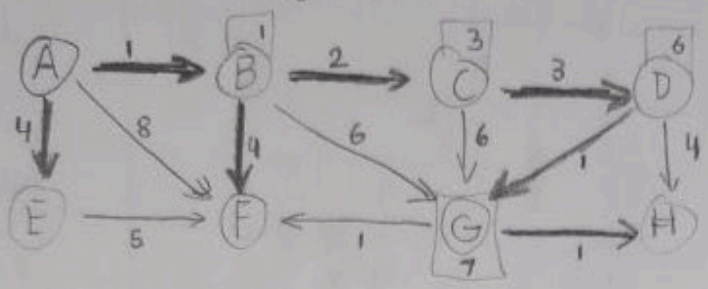
Assignment 14

1) Dijkstra's Algorithm



A ∞ 0
B ∞ 1
C ∞ 3
D ∞ 6
E ∞ 4
F ∞ ~~∞~~ 5
G ∞ 7
H ~~∞~~ ~~10~~ 8

unvisited
[A, B, C, D, E, F, G, H]

PQ:

A-B 1
A-E 4
A-F 8

→

B-C 3
A-E 4
B-F 5
B-G 7
A-F 8

→

A-E 4
B-F 5
C-D 6
B-G 7
A-F 8
C-G 9

→

B-F 5
C-D 6
B-G 7
~~A-F 8~~
~~E-F 9~~
C-G 9

← both nodes have now been visited so delete

→

C-D 6
B-G 7
C-G 9

→

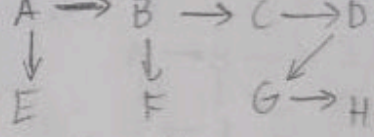
~~B-G 7~~
~~C-G 9~~
E-F 1

→

G-H 8

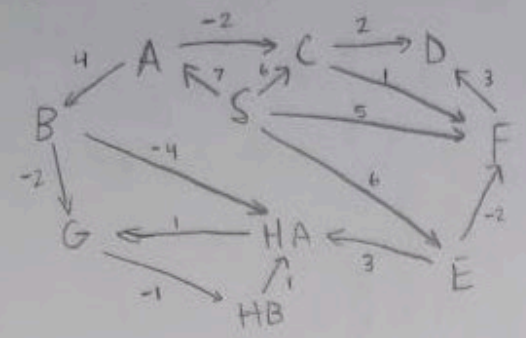
all nodes have been visited.

Final:



cost = 16

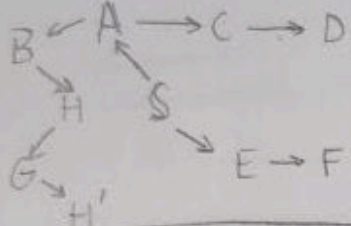
2) Bellman Ford



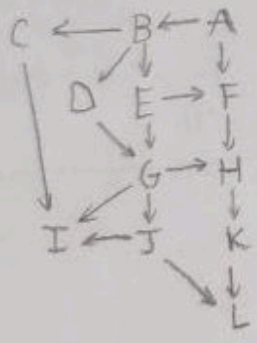
	S	A	C	E	B'	H	G
A	7						
B		11					
C		5					
D			1				
E	6						
F	5			4			
G					9	8	
HA					9	7	
HB							7

(cost, EdgeTo)
(7, S)
(15, A)
(5, A)
(7, C)
(6, S)
(4, E)
(8, H)
(7, B)
(7, 6)

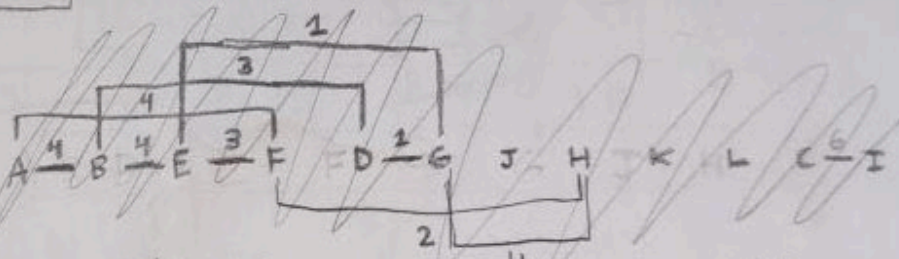
Final Tree (cost: 11)



3) DAG



1.



can solve using a queue based approach:

Topological sort:

DFS (A → B → D → G → I) — I
DFS (J, L) — L, J
DFS (H, K) — K, H, G, D
DFS (E, F) — F, E
DFS (C) — C, B, A

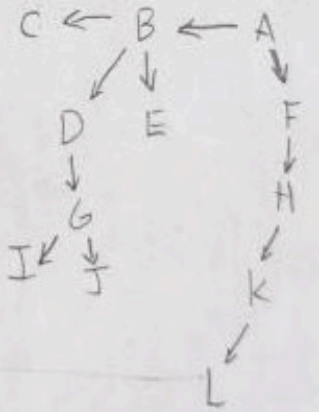
resulting stack: A B C D E F D G H K J L I

relax →

relaxing:	A	B	C	D	E	F	D	G	H	K	Final
A	0, A										0, A
B	4, A										4, A
C		9, B									9, B
D		7, B									7, B
E		8, B									8, B
F	4, A										9, A
G					~~9, E~~		8, D				8, D
H						6, F					6, F
I					~~15, C~~			10, G			10, G
J								11, G			11, G
K									9, H		9, H
L										10, K	10, K

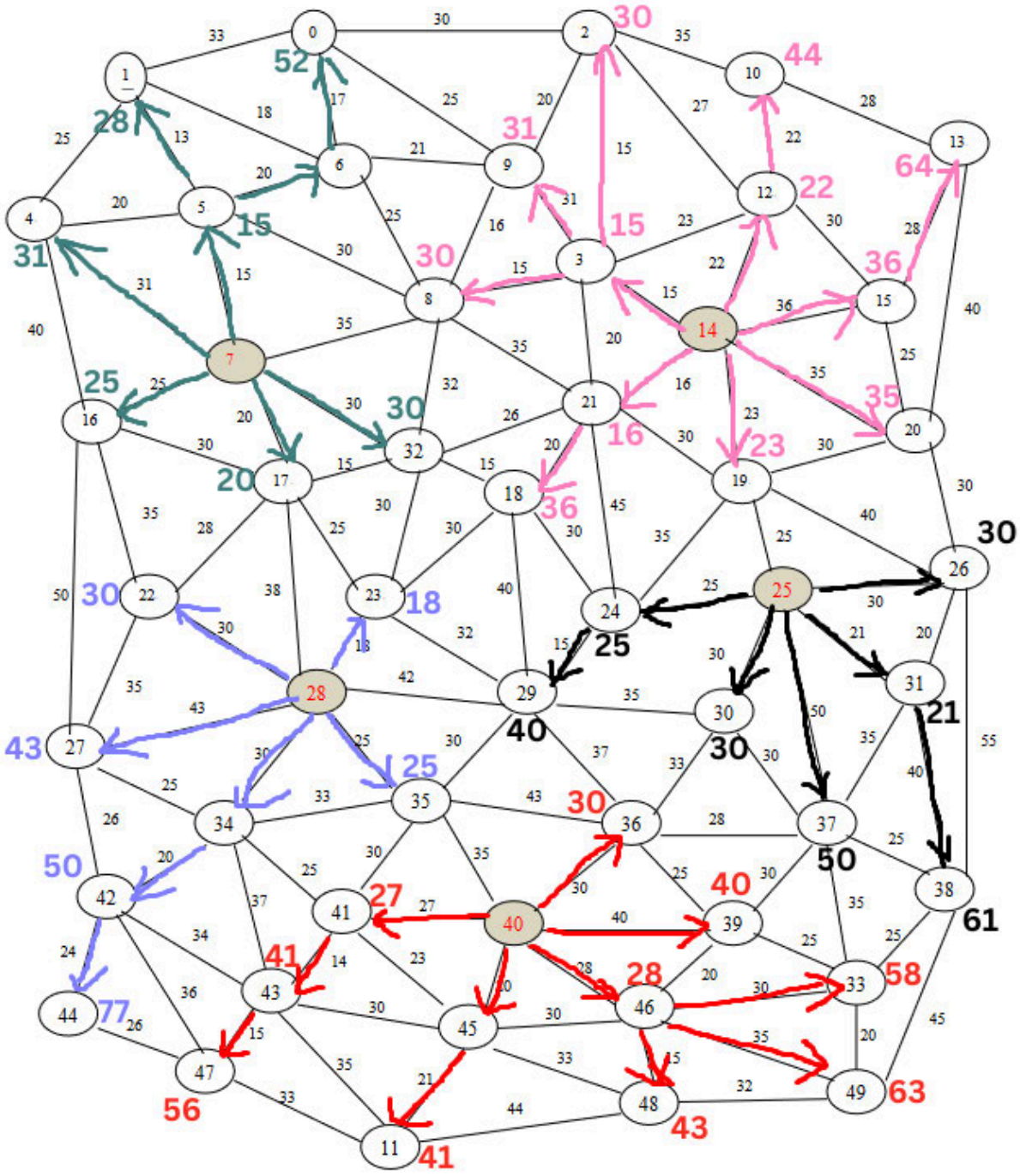
results in shortest path of:

cost = 32



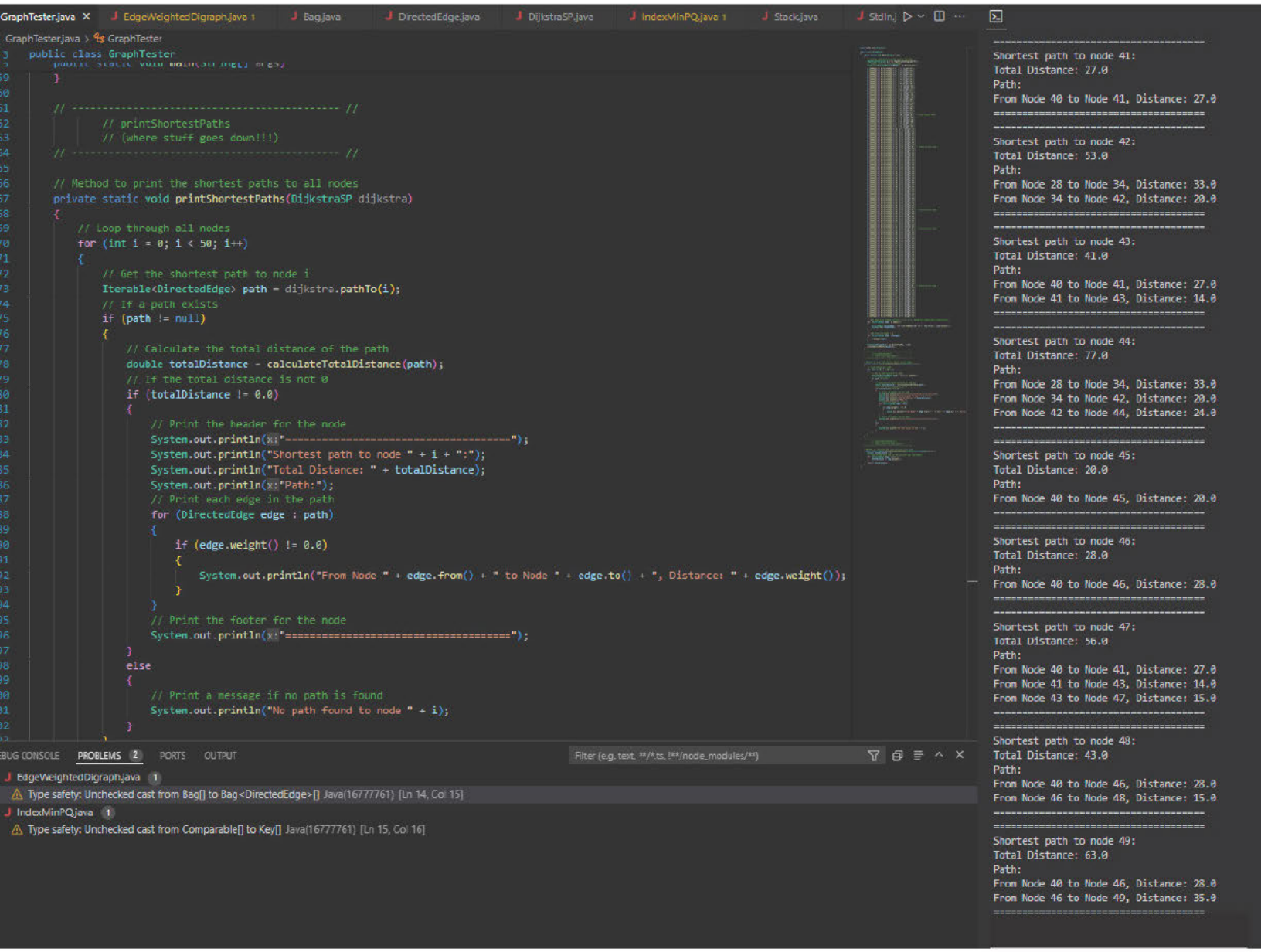
FOR PROGRAMMING ASSIGNMENT VVVV

Customer can efficiently serve through the following stations and paths:



If I had to chose a node to be a station, I would likely pick 18 as it is central and has quite cheap options. [redacted]

Just incase I missed any code, here is my output:



I hope this algorithm is efficient and suits your needs