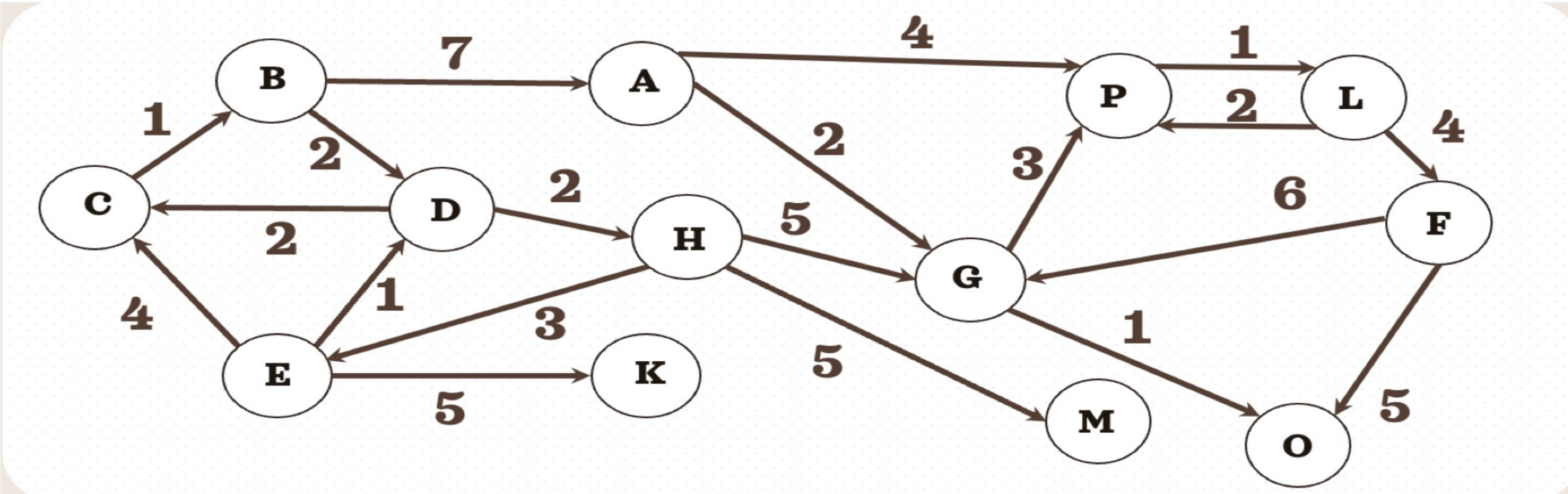
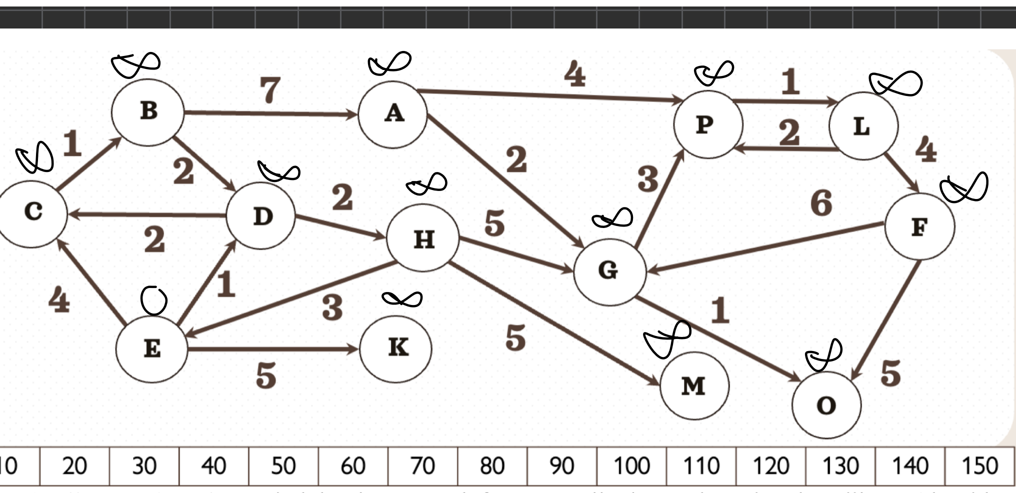


10	20	30	40	50	60	70	80	90	100	110	120	130	140	150

1 (text) Dijkstra [15 points] Find the shortest path from E to all other nodes using the Dijkstra Algorithm (Show your work and intermediate steps)

2 (text) A* [20 points] Find the shortest path from E to all other nodes using the A* Algorithm (Show your work and intermediate steps) **Hint:** Use the geographical location values to derive the heuristic to use.



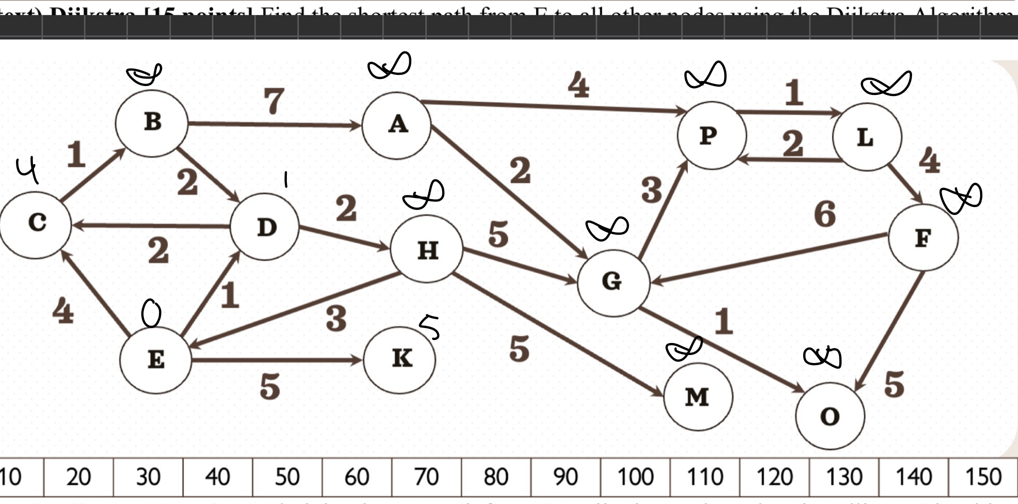
Dijkstra's Algorithm

Initialize all distances to ∞

Set distance to target to 0

Add to visited nodes

Visited: E   Current: E



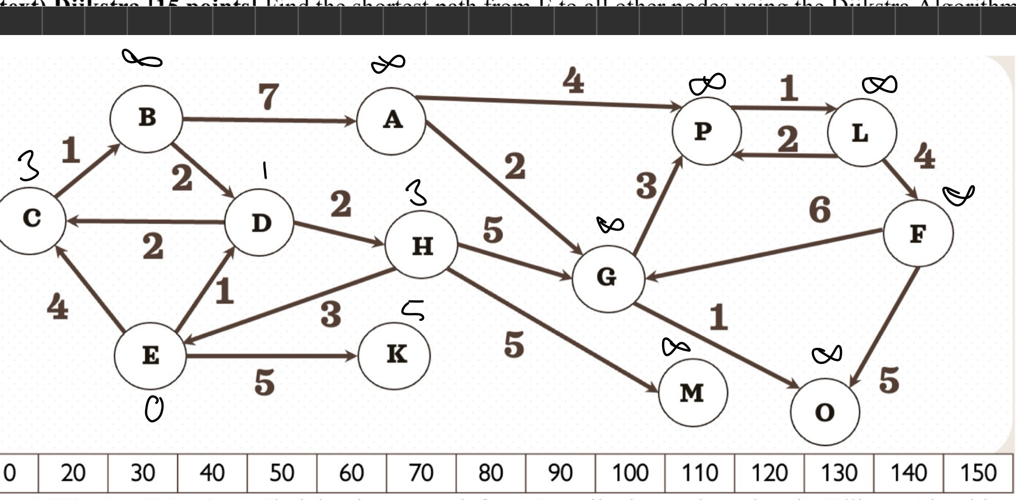
Update cost to adjacent nodes   1
if current.cost + Edge.cost < adjacent.cost;

Add adjacent nodes to Frontier Queue. Min Heap 2

Current = Frontier.Pop   3

Add Current to visited.   4

Visited: E, D   Current: D
Frontier: C, K
Repeat previous steps. 1-4



Current: D

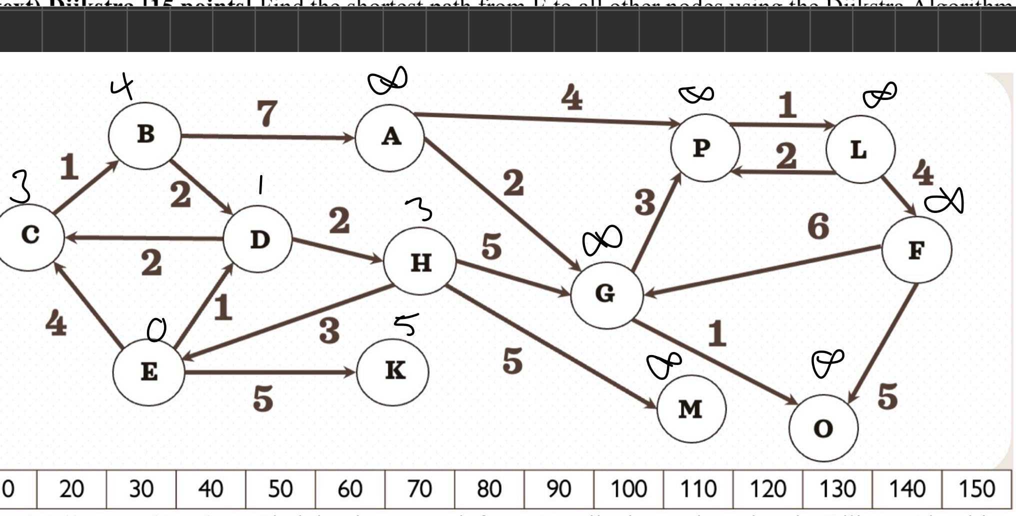
Adjacent: C, H   Update

Frontier: C, H, K.

New Current: C

Visited: E, D, C

Frontier: H, K



Current: C

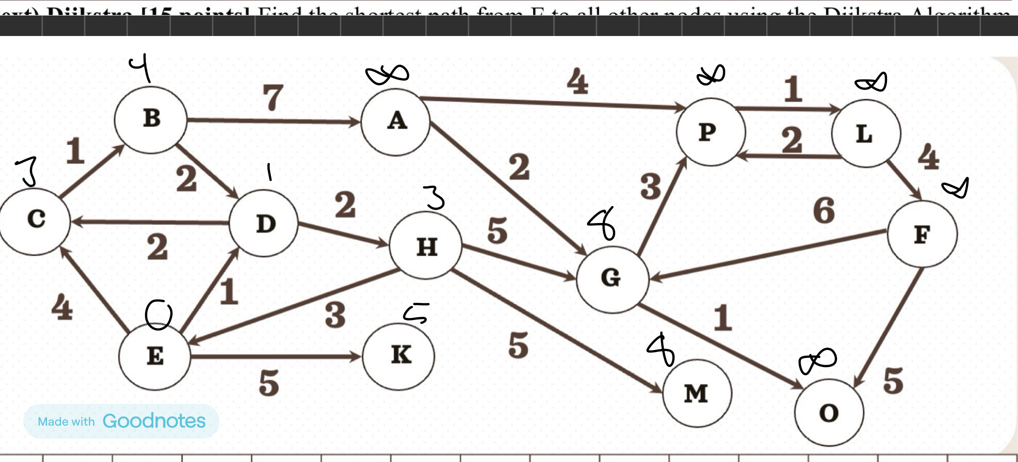
Adjacent: B   update

Frontier: H, B, K

New current: H

Visited: E, D, C, H

Frontier: B, K



Current: H

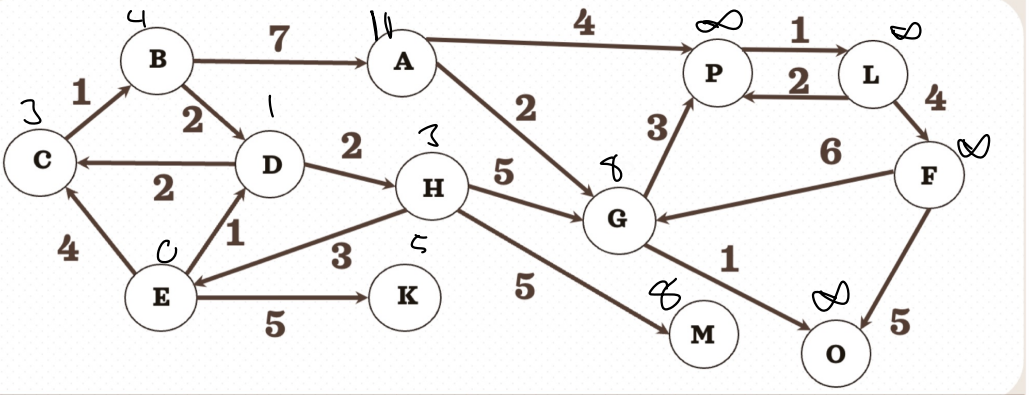
Adjacent: G, M

Frontier: B, K, G, M

New Current: B   Frontier: K, G, M

Visited: E, D, C, H, B

Made with Goodnotes



10	20	30	40	50	60	70	80	90	100	110	120	130	140	150

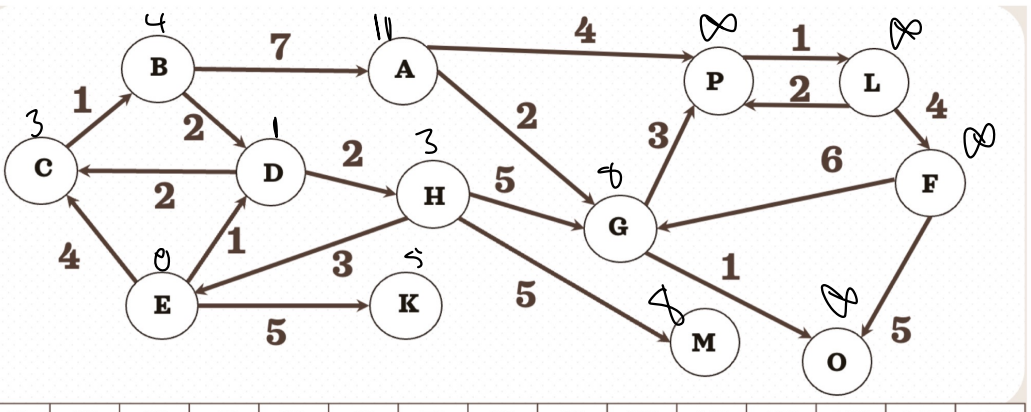
Current: B

Adjacent: A, D Update. D is already visited

Frontier: K, G, M, A

New Current: K

Visited: E, D, C, H, B, K Frontier: G, M, A



10	20	30	40	50	60	70	80	90	100	110	120	130	140	150

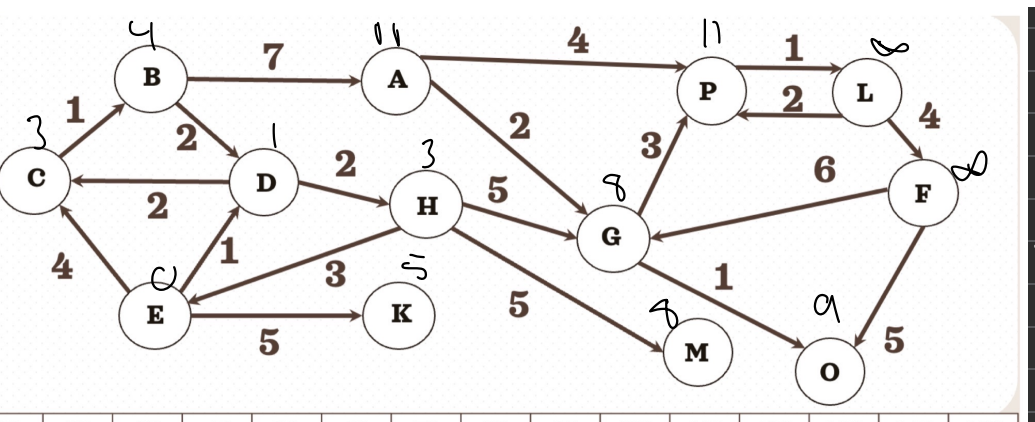
Current: K

Adjacent: null

Frontier: G, M, A

New Current: G

Visited: E, D, C, H, B, K, G Frontier: M, A



10	20	30	40	50	60	70	80	90	100	110	120	130	140	150

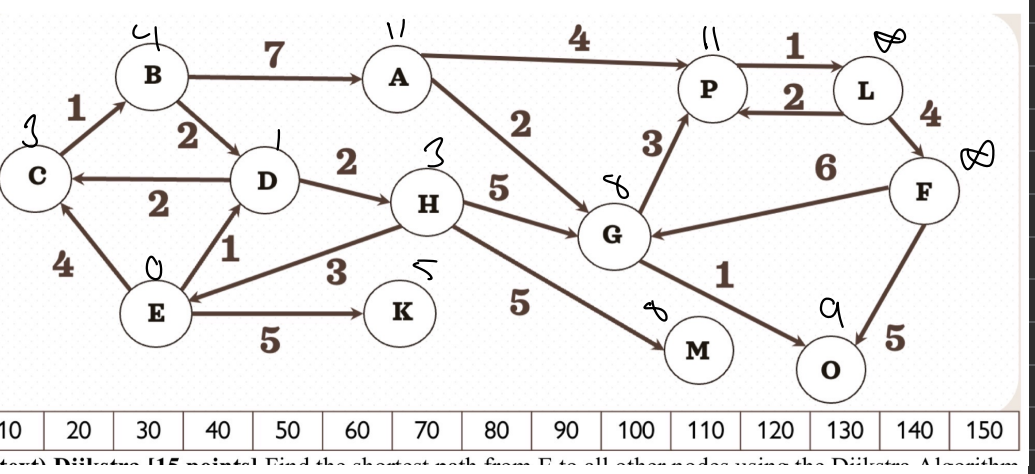
Current: G

Adjacent: P, O

Frontier: O, M, A, P

New Current: O

Visited: E, D, C, H, B, K, G, O Frontier: M, A, P



10	20	30	40	50	60	70	80	90	100	110	120	130	140	150

Current: O

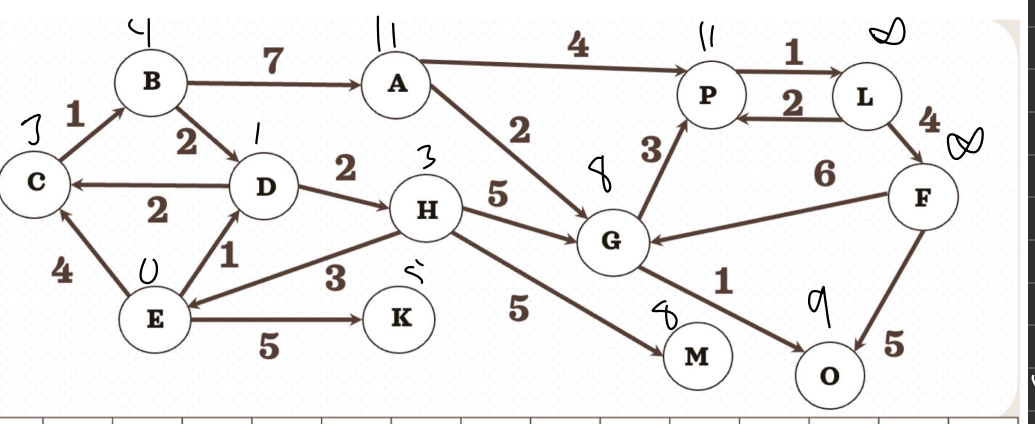
Adjacent: Null

Frontier: M, A, P

New Current: M

Visited: E, D, C, H, B, K, G, O, M

Frontier: A, P



10	20	30	40	50	60	70	80	90	100	110	120	130	140	150

Current: M

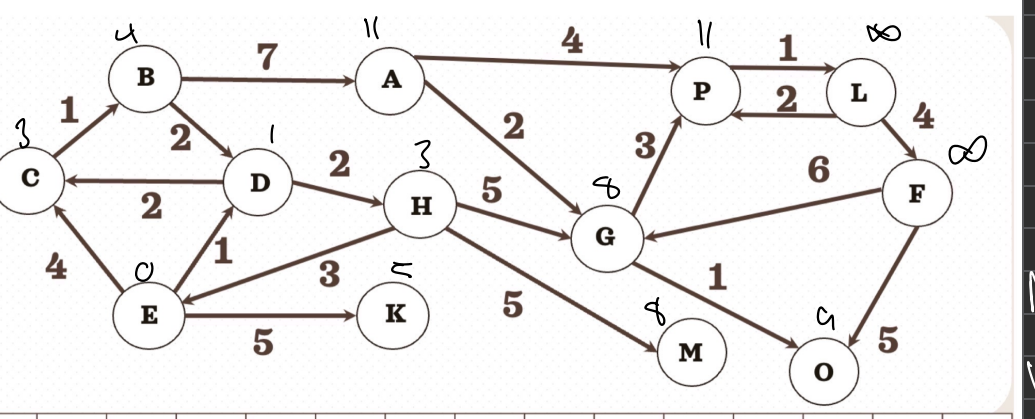
Adjacent: NULL

Frontier: A, P

New Current: A

Visited: E, D, C, H, B, K, G, O, M, A

Frontier: P



0	20	30	40	50	60	70	80	90	100	110	120	130	140	150

Current: A

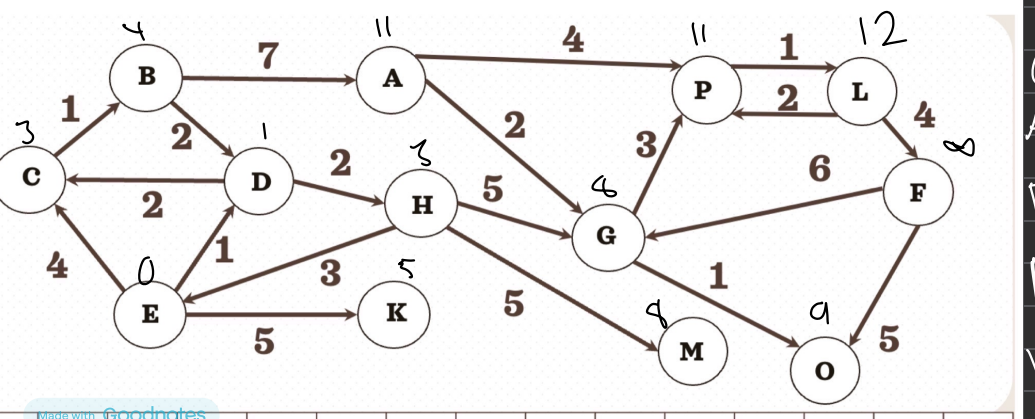
Adjacent: G, P

Frontier: P

New Current: P

Visited: E, D, C, H, B, K, G, O, M, A, P

Frontier:



0	20	30	40	50	60	70	80	90	100	110	120	130	140	150

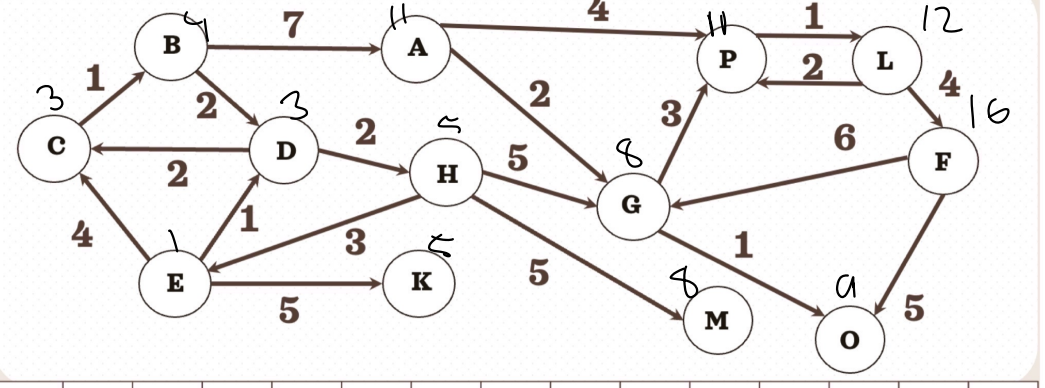
Current: P

Adjacent: L

Frontier: L

New Current: L

Visited: E, D, C, H, B, K, G, O, M, A, P, L Frontier:



10	20	30	40	50	60	70	80	90	100	110	120	130	140	150

(text) **Dijkstra [15 points]** Find the shortest path from F to all other nodes using the Dijkstra Algorithm

Current: L

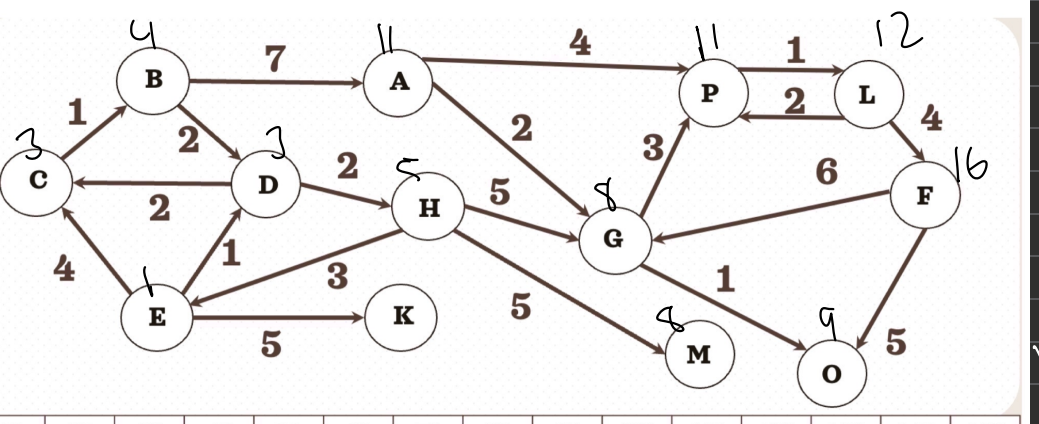
Adjacent: P, F

Frontier: F

New Current: F

Visited: E, D, C, H, B, K, G, O, M, A, P, L, F

Frontier:



10	20	30	40	50	60	70	80	90	100	110	120	130	140	150

(text) **Dijkstra [15 points]** Find the shortest path from F to all other nodes using the Dijkstra Algorithm

Current: F

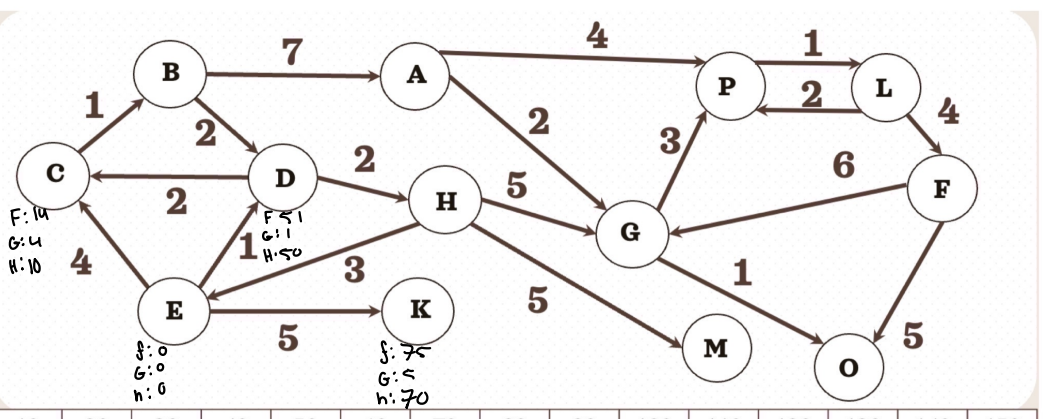
Adjacent: G, O

Frontier:

New current: Null

Visited: E, D, C, H, B, K, G, O, M, A, P, L, F

Frontier: Null.



10	20	30	40	50	60	70	80	90	100	110	120	130	140	150

(text) **Dijkstra [15 points]** Find the shortest path from E to all other nodes using the Dijkstra Algorithm

Visited: E, C
Frontier: D, K

A*

A*

Start at node:

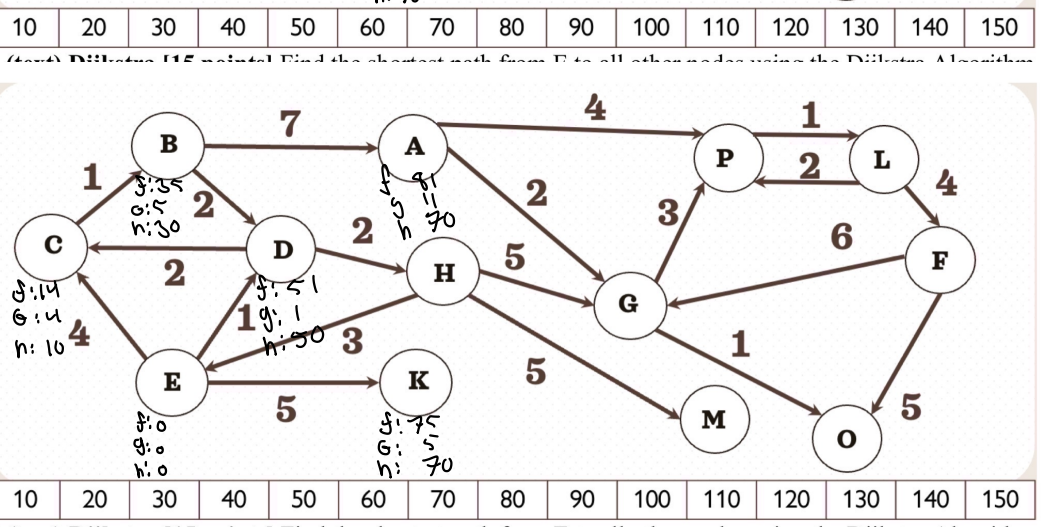
Initialize E with $f(n)=0$, $g(n)=0$, $h(n)=0$

Update adjacent nodes $f(n)$, $g(n)$, $h(n)$.

Adjacent: C, D, K

C	D	K
f: 14	f: 51	f: 75
G: 4	G: 1	G: 5
h: 10	h: 50	h: 70

Add start to visited.
Add: adjacent to frontier if not visited.
New Current: Frontier.Pop
Repeat.



10	20	30	40	50	60	70	80	90	100	110	120	130	140	150

(text) **Dijkstra [15 points]** Find the shortest path from E to all other nodes using the Dijkstra Algorithm

Current: C

Adjacent: B

B
f: 35
G: 5
h: 30

Frontier: B, D, K

New Current: B
Visited: E, C, B, Frontier: D, K

10	20	30	40	50	60	70	80	90	100	110	120	130	140	150

(text) **Dijkstra [15 points]** Find the shortest path from E to all other nodes using the Dijkstra Algorithm

Current: B

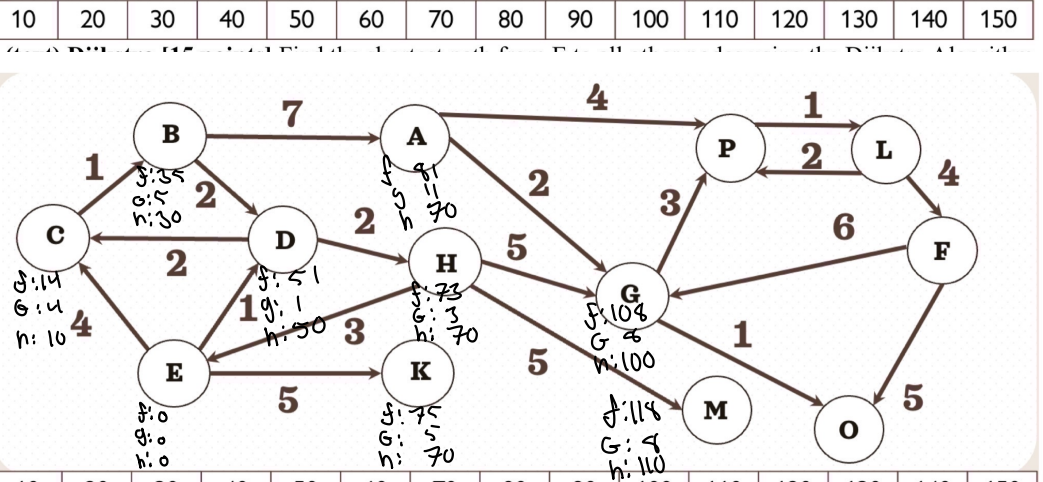
Adjacent: A, D (D not updated.)

A	D
f: 81	F: 57
G: 11	G: 7
h: 70	h: 50

Frontier: D, K, A ✓

New Current: D

Frontier: K, A Visited: E, C, B, D



10	20	30	40	50	60	70	80	90	100	110	120	130	140	150

(text) **Dijkstra [15 points]** Find the shortest path from E to all other nodes using the Dijkstra Algorithm

Current: D

Adjacent: H

H
f: 73
G: 3
H: 70

Frontier: H, K, A

New Current: H
Frontier: K, A Visited: E, C, B, D, H

10	20	30	40	50	60	70	80	90	100	110	120	130	140	150

(text) **Dijkstra [15 points]** Find the shortest path from E to all other nodes using the Dijkstra Algorithm

Current: H
Adjacent: M, G

M	G
f 118	f: 108
G: 8	G: 8
h: 110	H: 100

Frontier: K, A, G, M

New Current: K Visited: E, C, B, D, H, K
Frontier: A, G, M

Current: K

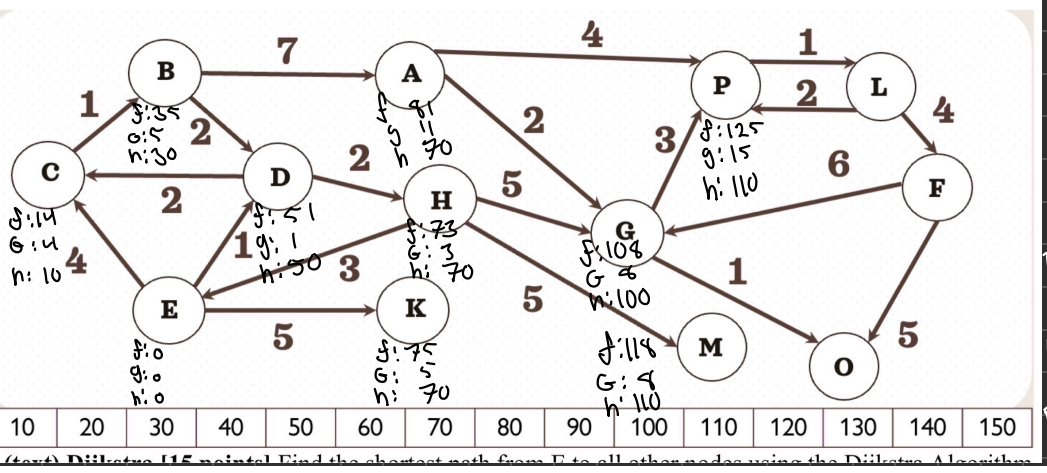
Adjacent: Null

Frontier: A, G, M

New Current: A

Frontier: G M

Visited: E, C, B, D, H, K, A



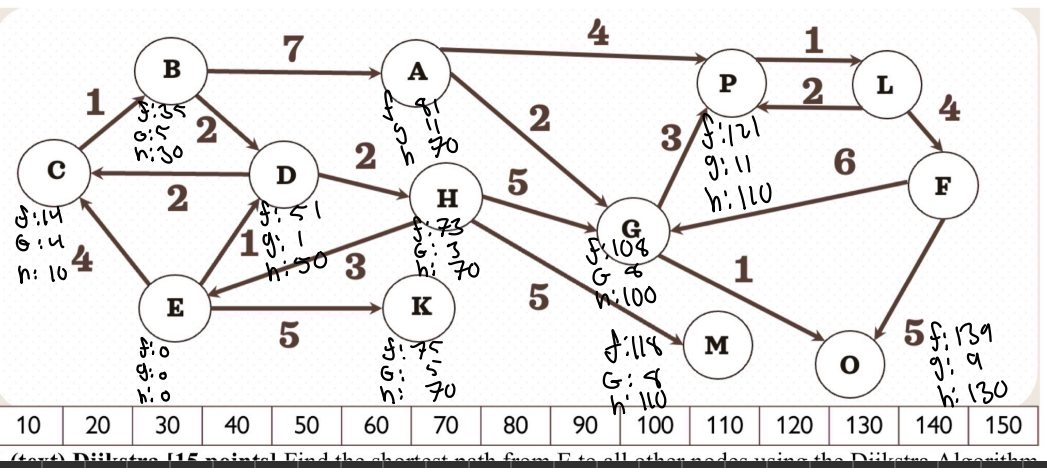
Current: A

Adjacent: G (f: 113, G: 13, h: 100) P (f: 125, G: 15, h: 110)

Frontier: G, M, P

New Current: G

Frontier: M, P Visited: E, C, B, D, H, K, A, G



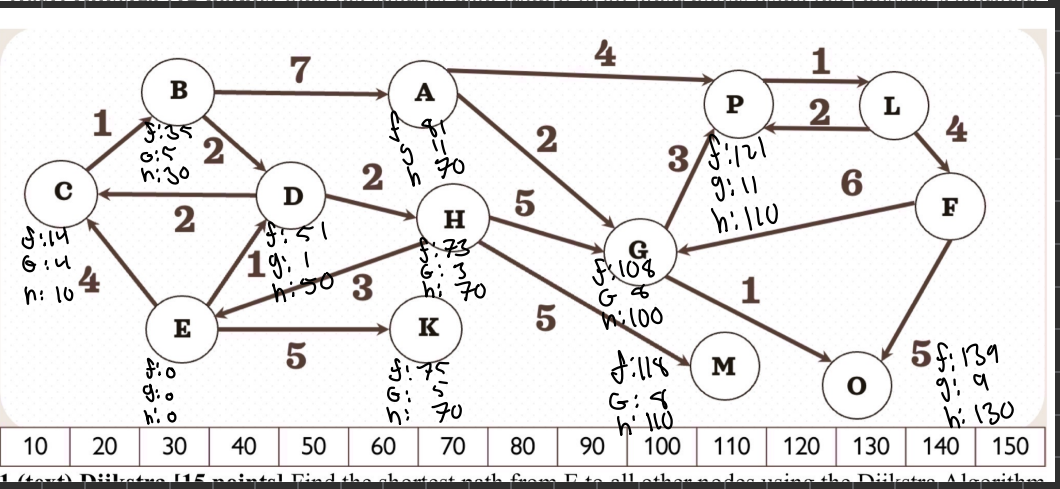
Current: G

Adjacent: P (f: 121, g: 11, h: 110) O (f: 134, g: 9, h: 130) Update P

Frontier: M, P, O

New Current: M

Frontier: P, O Visited: E, C, B, D, H, K, A, G, M



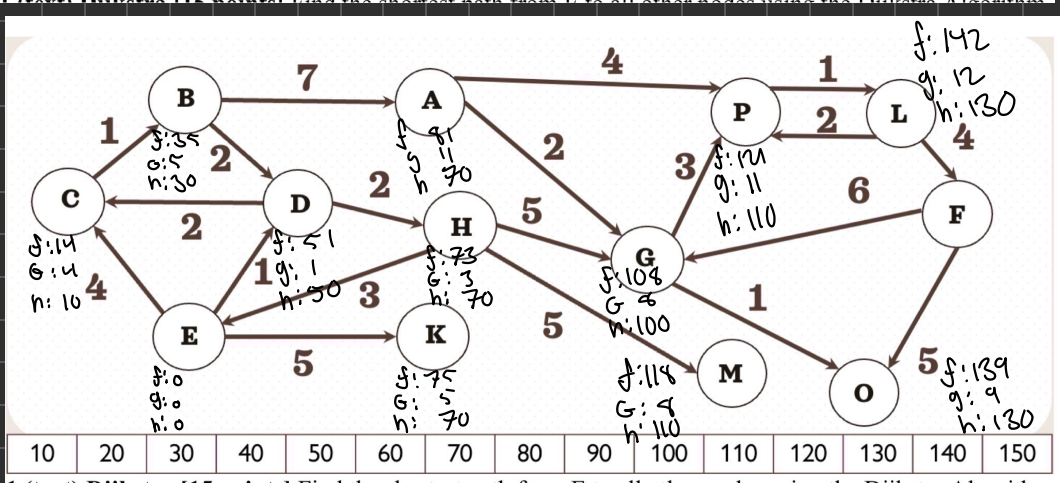
Current: M

Adjacent: Null

Frontier: P, O

New Current: P

Frontier: O Visited: E, C, B, D, H, K, A, G, M, P



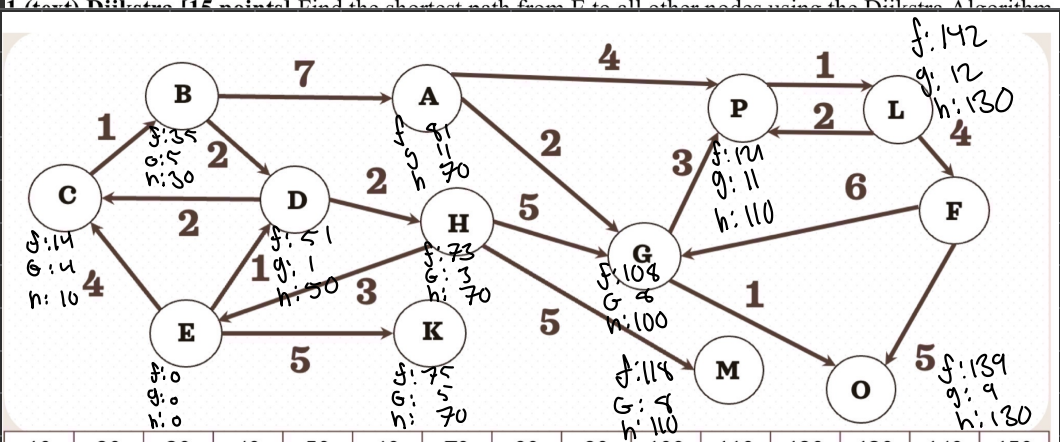
Current: P

Adjacent: L (f: 142, g: 12, h: 130)

Frontier: O, L

New Current: O

Frontier: L Visited: E, C, B, D, H, K, A, G, M, P, O



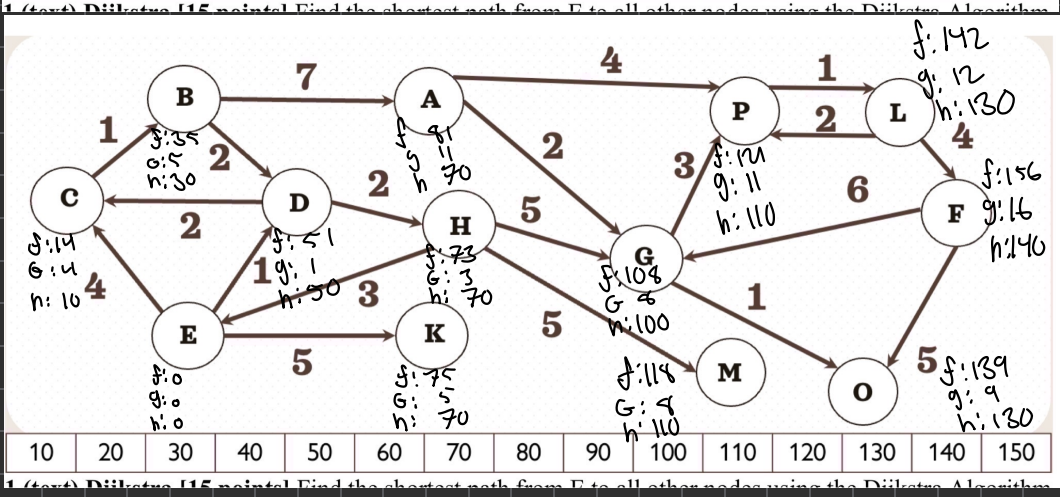
Current: O

Adjacent: Null

Frontier: L

New Current: L

Frontier: Visited: E, C, B, D, H, K, A, G, M, P, O, L



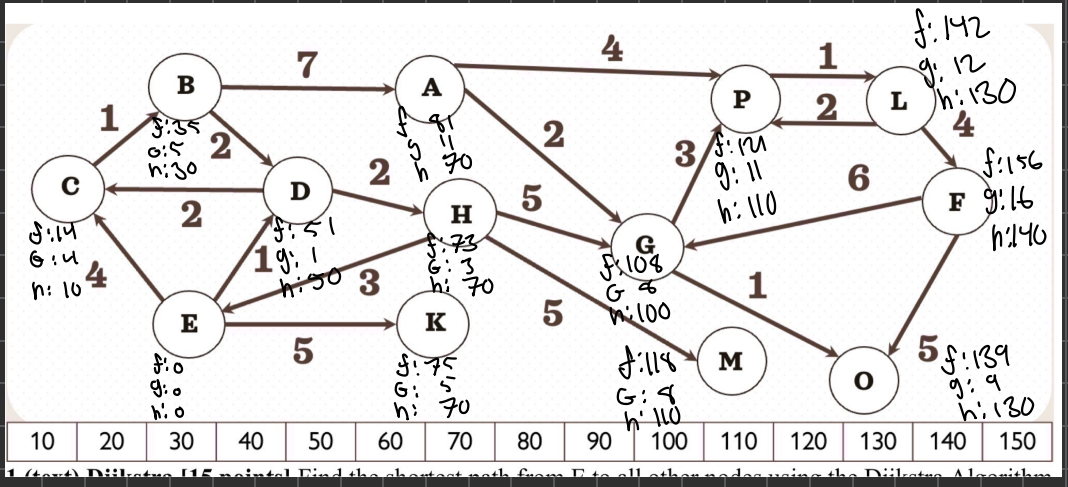
Current: L

Adjacent: P (f: 124, g: 14, h: 110) F (f: 156, g: 16, h: 140) P not updated

Frontier: F

New Current: F

Visited: E, C, B, D, H, K, A, G, M, P, O, L, F



Current: F

Adjacent:	O	G
f:	151	122
g:	21	22
h	130	100

None updated

frontier: Null.

Visited: E, C, B, D, H, K, A, G, M, P, O, L, F

3 (text) Comparison [5 points] Which algorithm found the shortest path to H in less iterations Dijkstra or A*? (Explain your answer)

Dijkstra found the shortest path first. Although both algorithms found H in four iterations. Dijkstra could have gone to H earlier. This because at the moment H and C were the shortest path available. If I had gone to H first then Dijkstra was in less iterations. A* saw that the f was too high on h so it didn't find the shortest path yet.