Name:____________________

Find the shortest distance from 'S' to 'E', using Dijkstra's Algorithm.

1. Fill in 0 for the distance of the location that stores 'S' and set linked from to null
2. Find the location with the smallest distance that is not marked as complete (referred to as **current**)
3. If **current** is the position with 'E' stop, otherwise is not marked complete go to step 3
4. For each neighboring location (referred to as **neighbor**)
    a. Let **distanceToNeighbor** be **current**.distance + **neighbor**.terrianCost
    b. If **neighbor.**distance is not set or (**distanceToNeighbor < neighbor.**distance) go to step c, otherwise continue to next neighbor
    c. Set **neighbor**.distance to **distanceToNeighbor**
    d. Set Linked from to **current**
5. Set **current**.state to complete.

TerrainCosts: (S-0), (E-0), (G-1), (F-2), (H-3), (W-7), (M-12)

M(0)	H(1)	H(2)	S(3)	F(4)
E(5)	M(6)	H(7)	F(8)	G(9)
W(10)	W(11)	G(12)	F(13)	F(14)

Results Tables:

Location (row, column)	Linked From (Points)	Distance	Status
(0,0) - (0)	(0,1)	18	
(0,1) - (1)	(0,2)	6	complete
(0,2) - (2)	(0,3)	3	complete
(0,3) - (3)	null	0	complete
(0,4) - (4)	(0,3)	2	complete
(1,0) - (5)	(1,1)	17	
(1,1) - (6)	(1,2)	17	complete
(1,2) - (7)	(1,3)	5	complete
(1,3) - (8)	(0,3)	2	complete
(1,4) - (9)	(0,4)	3	complete
(2,0) - (10)	(2,1)	19	
(2,1) - (11)	(2,4)	12	complete
(2,2) - (12)	(2,3)	5	complete
(2,3) - (13)	(1,3)	4	complete
(2,4) - (14)	(0,4)	5	complete

Find the shortest distance from 'S' to 'E', using Dijkstra's Algorithm.

Name:____________________

1. Fill in 0 for the distance of the location that stores ‘S’ and set linked from to null
2. Find the location with the smallest distance that is not marked as complete (referred to as **current**)
3. If **current** is the position with ‘E’ stop, otherwise is not marked complete go to step 3
4. For each neighboring location (referred to as **neighbor**)
    a. Let **distanceToNeighbor** be **current**.distance + **neighbor**.terrianCost
    b. If **neighbor.**distance is not set or (**distanceToNeighbor < neighbor.**distance) go to step c, otherwise continue to next neighbor
    c. Set **neighbor**.distance to **distanceToNeighbor**
    d. Set Linked from to **current**
5. Set **current**.state to complete.

TerrainCosts: (S-0), (E-0), (G-1), (F-2), (H-3), (W-7), (M-12)

F(0)	H(1)	M(2)	F(3)	E(4)
F(5)	S(6)	H(7)	G(8)	G(9)
F(10)	G(11)	M(12)	M(13)	M(14)

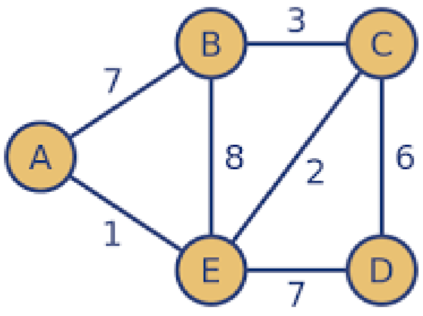
Results Tables:

Location (row, column)	Linked From	Distance	Status
(0,0) - (1)	(1,0)	4	complete
(0,1) - (1)	(1,1)	3	complete
(0,2) - (2)	(0,1)	15	
(0,3) - (3)	(1,3)	6	
(0,4) - (4)	(1,4)	5	
(1,0) - (5)	(1,1)	2	complete
(1,1) - (6)	null	0	complete
(1,2) - (7)	(1,1)	3	complete
(1,3) - (8)	(1,2)	4	complete
(1,4) - (9)	(1,3)	5	complete
(2,0) - (10)	(2,1)	3	complete
(2,1) - (11)	(1,1)	1	complete
(2,2) - (12)	(2,1)	13	
(2,3) - (13)	(1,3)	16	
(2,4) - (14)	(1,4)	17	

Name:____________________

Find the shortest distance from ‘A’ to ‘C’, using Dijkstra's Algorithm.

1. Fill in 0 for the distance of node ‘A’ and set linked from to null
2. Find the location with the smallest distance that is not marked as complete (referred to as **current**)
3. If **current** is the node with ‘C’ stop, otherwise is not marked complete go to step 3
4. For each neighboring location (referred to as **neighbor**)
    a. Let **distanceToNeighbor** be **current**.distance + **edgeCost** from **current->neighbor**
    b. If **neighbor.**distance is not set or (**distanceToNeighbor < neighbor.**distance) go to step c, otherwise continue to next neighbor
    c. Set **neighbor**.distance to **distanceToNeighbor**
    d. Set Linked from to **current**
5. Set **current**.state to complete.

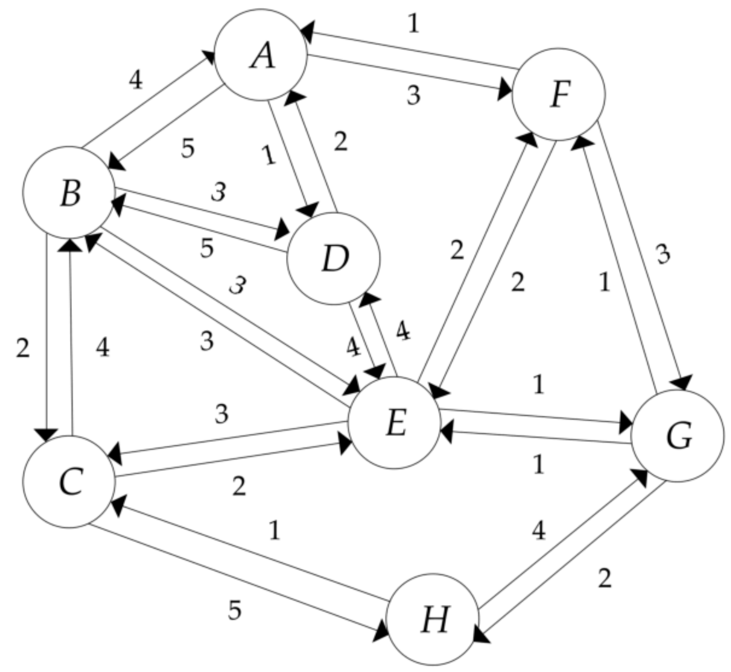


Node	**Linked From**	**Distance**	**Status**
A	null	0	complete
B	A	7	
C	C	3	complete
D	C	8	
E	A	1	complete

Name:___________________

Find the shortest distance from 'H' to 'F', using Dijkstra's Algorithm.

1. Fill in 0 for the distance of node 'H' and set linked from to null
2. Find the location with the smallest distance that is not marked as complete (referred to as **current**)
3. If **current** is the node with 'F' stop, otherwise is not marked complete go to step 3
4. For each neighboring location (referred to as **neighbor**)
    a. Let **distanceToNeighbor** be **current**.distance + **edgeCost** from **current->neighbor**
    b. If **neighbor.**distance is not set or (**distanceToNeighbor < neighbor.**distance) go to step c, otherwise go to step 5
    c. Set **neighbor**.distance to **distanceToNeighbor**
    d. Set Linked from to **current**
5. Set **current**.state to complete.



Node	Linked From	Distance	Status
A		∞	
B	C	5	
C	H	1	complete
D	E	7	
E	C	3	complete
F	E	5	complete
G	H	4	complete
H	null	0	complete

Name:____________________

Find the shortest distance from 'S' to 'E', using A*.

1. Fill in f=4, g=0, h=4 for the location that stores 'S' and set linked from to null
2. Find the location with the smallest f that is not marked as complete (referred to as **current**). If 2 locations have the same f value, use the h value instead.
3. For each neighboring location (referred to as **neighbor**)
   a. If **neighbor**.g is not set or **neighbor**.g > (**current.**g + **neighbor**.terrainCost) go to step b, otherwise continue to next neighbor
   b. Set Linked from to **current**
   c. Set **neighbor**.g to (**current.**f + **neighbor**.terrainCost)
   d. Set **neighbor**.h to (abs(row difference to E) + abs(column difference to E))
   e. Set **neighbor**.f to (**neighbor**.g + **neighbor**.f)
4. Set **current**.state to complete.
5. If **current** stores 'E' stop, otherwise go to step 2

TerrainCosts: (S-0), (E-0), (G-1), (F-2), (H-3), (W-7), (M-12)

M(0)	H(1)	H(2)	S(3)	F(4)
E(5)	M(6)	H(7)	F(8)	G(9)
W(10)	W(11)	G(12	F(13)	F(14)

Results Tables:

Location (row, column)	Linked From	f	g	h	Complete
(0,0) - (1)	(0,1)	28	27	1	
(0,1) - (1)	(0,2)	15	13	2	complete
(0,2) - (2)	(0,3)	10	7	3	complete
(0,3) - (3)	null	4	0	4	complete
(0,4) - (4)	(0,3)	11	6	5	complete
(1,0) - (5)	(1,1)	22	22	0	complete
(1,1) - (6)	(2,2)	22	21	1	complete
(1,2) - (7)	(1,3)	14	12	2	complete
(1,3) - (8)	(0,3)	9	6	3	comlpete
(1,4) - (9)	(1,3)	11	7	4	complete
(2,0) - (10)	(2,1)	31	30	1	
(2,1) - (1`)	(2,2)	23	21	2	complete
(2,2) - (12)	(2,3)	16	13	3	complete
(2,3) - (13)	(1,3)	12	8	4	complete
(2,4) - (14)	(1,4)	18	13	5	complete

Name:____________________

Find the shortest distance from 'S' to 'E', using A*.

1. Fill in f=3, g=0, h=3 for the location that stores 'S' and set linked from to null
2. Find the location with the smallest f that is not marked as complete (referred to as **current**) If 2 locations have the same f value, use the h value instead.
3. For each neighboring location (referred to as **neighbor**)
   a. If **neighbor**.g is not set or **neighbor**.g > (**current.**g + **neighbor**.terrainCost) go to step b, otherwise continue to next neighbor
   b. Set Linked from to **current**
   c. Set **neighbor**.g to (**current.**f + **neighbor**.terrainCost)
   d. Set **neighbor**.h to (abs(row difference to E) + abs(column difference to E))
   e. Set **neighbor**.f to (**neighbor**.g + **neighbor**.f)
4. Set **current**.state to complete.
5. If **current** stores 'E' stop, otherwise go to step 2

TerrainCosts: (S-0), (E-0), (G-1), (F-2), (H-3), (W-7), (M-12)

F(0)	S(1)	M(2)	F(3)	E(4)
F(5)	G(6)	H(7)	G(8)	G(9)
F(10)	G(11)	M(12)	M(13)	M(14)

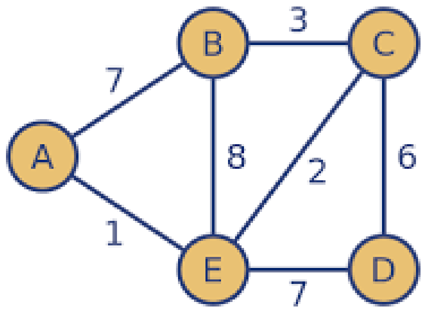
Results Tables:

Location (row, column)	Linked From	f	g	h	Complete
(0,0) - (0)	(0,1)	9	5	4	complete
(0,1) - (1)	null	3	0	3	complete
(0,2) - (2)	(0,1)	17	15	2	complete
(0,3) - (3)	(1,3)	20	19	1	
(0,4) - (4)	(1,4)	19	19	0	complete
(1,0) - (5)	(1,1)	15	10	5	complete
(1,1) - (6)	(0,1)	8	4	4	complete
(1,2) - (7)	(1,1)	14	11	3	complete
(1,3) - (8)	(1,2)	17	15	2	complete
(1,4) - (9)	(1,3)	19	18	1	complete
(2,0) - (10)	(2,1)	22	16	6	
(2,1) - (11)	(1,1)	14	9	5	complete
(2,2) - (12)	(1,2)	30	26	4	
(2,3) - (13)	(1,3)	32	29	3	
(2,4) - (14)	(1,4)	33	31	2	

Name:____________________

Find the shortest distance from ‘B’ to ‘A’, using A*.

1. Fill in f=1, g=0, h=1 for the location that stores ‘S’ and set linked from to null
2. Find the location with the smallest f that is not marked as complete (referred to as **current**) If 2 locations have the same f value, use the h value instead.
3. For each neighboring location (referred to as **neighbor**)
   a. If **neighbor**.g is not set or **neighbor**.g > (**current.**g + **neighbor**.terrainCost) go to step b, otherwise continue to next neighbor
   b. Set Linked from to **current**
   c. Set **neighbor**.g to (**current.**f + **neighbor**.terrainCost)
   d. Set **neighbor**.h to (measured distance in inches rounded up to the next inch)
   e. Set **neighbor**.f to (**neighbor**.g + **neighbor**.f)
4. Set **current**.state to complete.

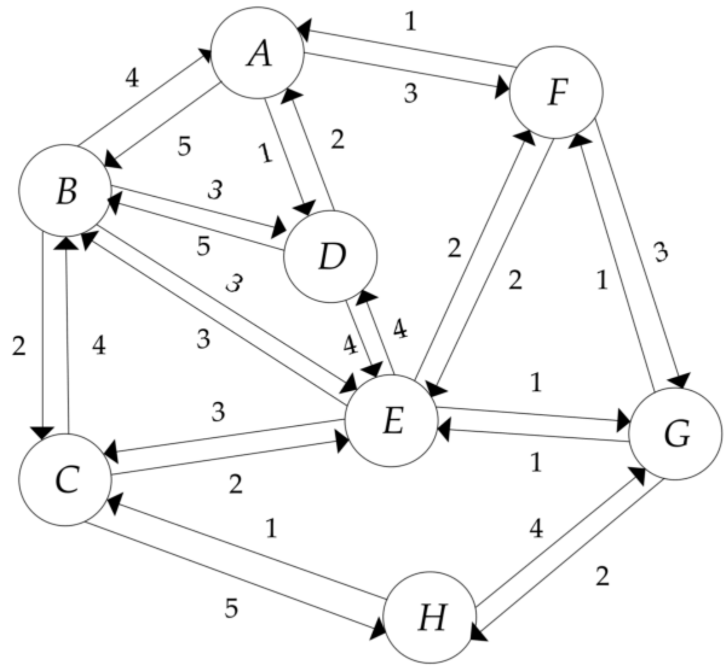


Node	Linked From	f	g	h	Complete
A	E	6	6	0	complete
B	null		0		complete
C	B	5	3	2	complete
D	C	11	9	2	
E	C	8	7	1	

Name:___________________

Find the shortest distance from ‘B’ to ‘G’, using A*.

1. Fill in f=1, g=0, h=1 for the location that stores ‘S’ and set linked from to null
2. Find the location with the smallest f that is not marked as complete (referred to as **current**) If 2 locations have the same f value, use the h value instead.
3. For each neighboring location (referred to as **neighbor**)
    a. If **neighbor**.g is not set or **neighbor**.g > (**current.**g + **neighbor**.terrainCost) go to step b, otherwise continue to next neighbor
    b. Set Linked from to **current**
    c. Set **neighbor**.g to (**current.**f + **neighbor**.terrainCost)
    d. Set **neighbor**.h to (measured distance in inches rounded up to the next inch)
    e. Set **neighbor**.f to (**neighbor**.g + **neighbor**.f)
4. Set **current**.state to complete.



Node	Linked From	f	g	h	Complete
A	B	6	4	2	
B	null	3	0	3	complete
C	B	5	2	3	
D	B	5	3	2	
E	B	4	3	1	complete
F	E	7	5	2	
G	E	4	4	0	complete
H					