CS455 – Algorithms and Data Structures
Week12 - Homework2

Question1:

https://npu85.npu.edu/~henry/npu/classes/algorithm/graph_alg/slide/exercise_graph_alg.html

Q24 ==> Use Bellman Ford's Algorithm to find the shortest path of a maze

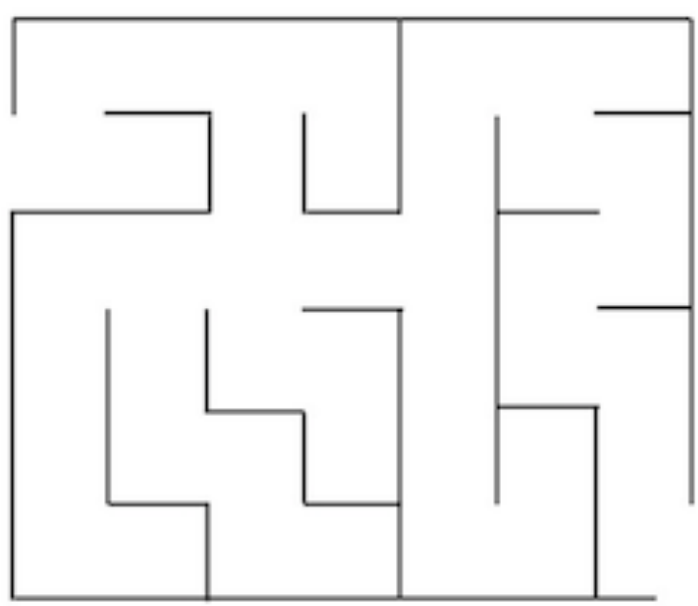
Solution:

Step1:

Bellman Ford Algorithm:

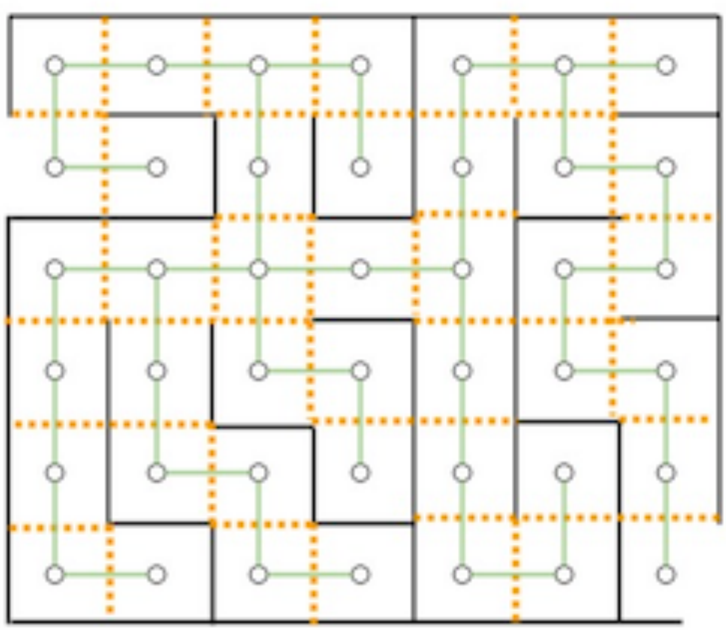
Part 1: Firstly let's create a graph from the maze.

Step 1:

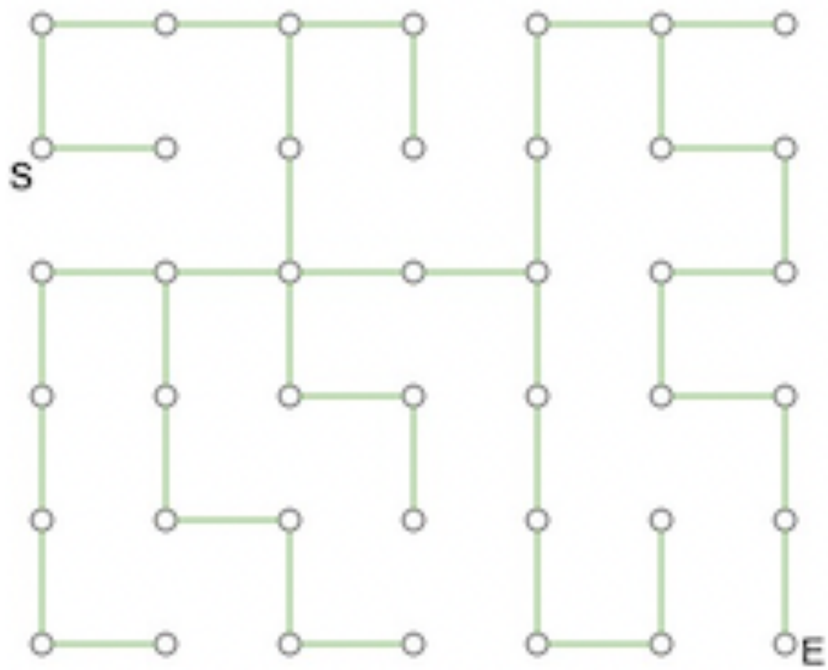


Step 2:

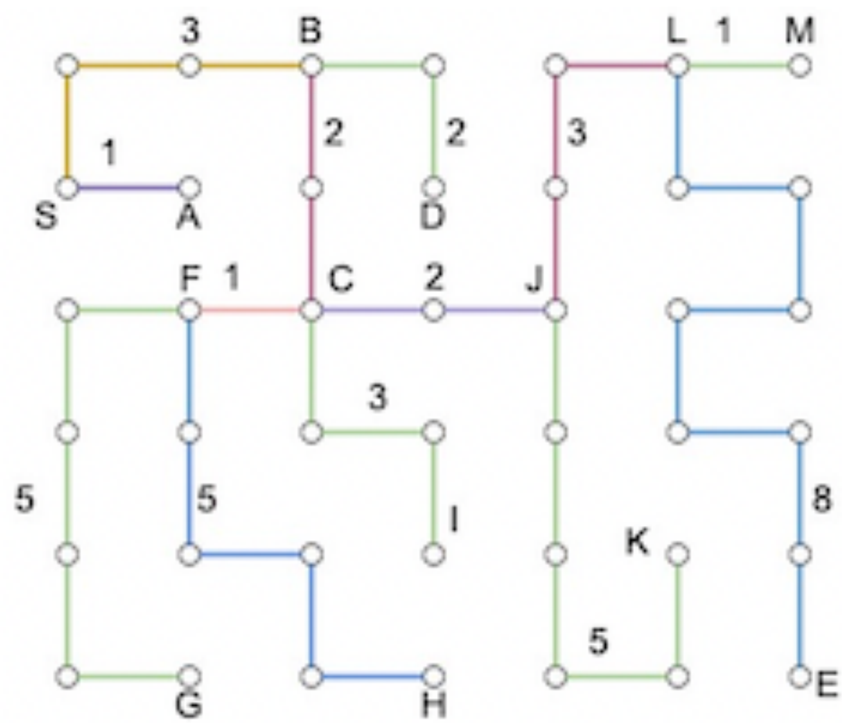
Step 3:



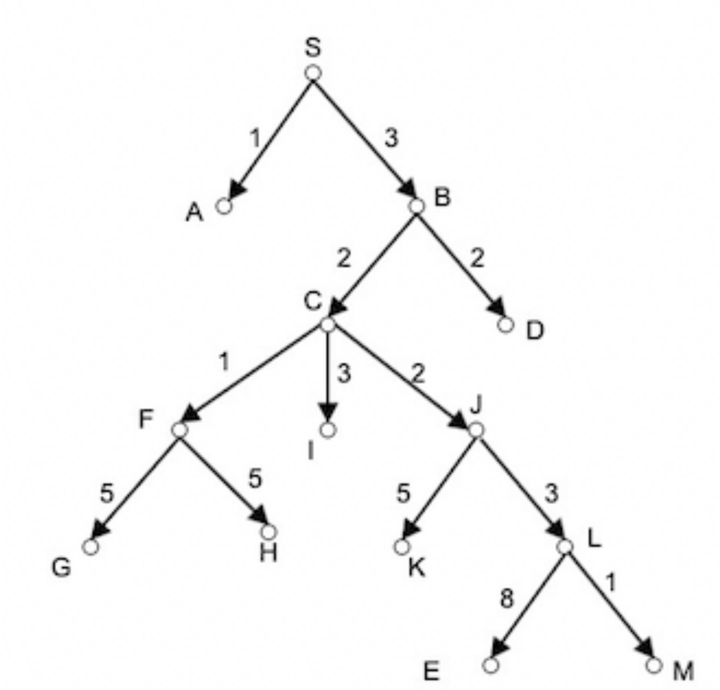
Step 4:



Step 5:



STEP 6: Convert into graph



Part 2: Shortest Path Using Bellman Ford

Cycle 1:

Set all the distances to infinity

0	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
S	A	B	C	D	E	F	G	H	I	J	K	L	M

Iteration 1:

From S, we can go to A and B.

S—>A = 0+1 = 1

S—>B = 0+3 = 3

0	1	3	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
S	A	B	C	D	E	F	G	H	I	J	K	L	M

Iteration 2:

From A, we can't go anywhere.

0	1	3	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
S	A	B	C	D	E	F	G	H	I	J	K	L	M

Iteration 3:

From B, we can go to C and D.

B—>C = 3+2 = 5

B—>D = 3+2 = 5

0	1	3	5	5	∞	∞	∞	∞	∞	∞	∞	∞	∞
S	A	B	C	D	E	F	G	H	I	J	K	L	M

Iteration 4:

From C, we can go to F , I and J.

C—>F = 5+1 = 6

C—>I = 5+3 = 8

C—>J = 5+2 = 7

0	1	3	5	5	∞	6	∞	∞	8	7	∞	∞	∞
S	A	B	C	D	E	F	G	H	I	J	K	L	M

Iteration 5:

From D, we can't go anywhere.

0	1	3	5	5	∞	6	∞	∞	8	7	∞	∞	∞
S	A	B	C	D	E	F	G	H	I	J	K	L	M

Iteration 6:

From E, we can't go anywhere.

0	1	3	5	5	∞	6	∞	∞	8	7	∞	∞	∞
S	A	B	C	D	E	F	G	H	I	J	K	L	M

Iteration 7:

From F, we can go to G and H

F—>G = 6+5 = 11

F—>H = 6+5 = 11

0	1	3	5	5	∞	6	11	11	8	7	∞	∞	∞
S	A	B	C	D	E	F	G	H	I	J	K	L	M

Iteration 8:

From G, we can't go anywhere.

0	1	3	5	5	∞	6	11	11	8	7	∞	∞	∞
S	A	B	C	D	E	F	G	H	I	J	K	L	M

Iteration 9:

From H, we can’t go anywhere.

0	1	3	5	5	∞	6	11	11	8	7	∞	∞	∞
S	A	B	C	D	E	F	G	H	I	J	K	L	M

Iteration 10:

From I, we can’t go anywhere.

0	1	3	5	5	∞	6	11	11	8	7	∞	∞	∞
S	A	B	C	D	E	F	G	H	I	J	K	L	M

Iteration 11:

From J, we can go to K and L.

J—>K = 7+5 = 12

J—>L = 7+3 = 10

0	1	3	5	5	∞	6	11	11	8	7	12	10	∞
S	A	B	C	D	E	F	G	H	I	J	K	L	M

Iteration 12:

From K, we can’t go anywhere.

0	1	3	5	5	∞	6	11	11	8	7	12	10	∞
S	A	B	C	D	E	F	G	H	I	J	K	L	M

Iteration 13:

From L, we can go to E and M.

L—>E= 10+8 = 18

L—>M= 10+1 = 11

0	1	3	5	5	18	6	11	11	8	7	12	10	11
S	A	B	C	D	E	F	G	H	I	J	K	L	M

Iteration 14:

From M, we can't go anywhere.

0	1	3	5	5	18	6	11	11	8	7	12	10	11
S	A	B	C	D	E	F	G	H	I	J	K	L	M

Cycle 2:

0	1	3	5	5	18	6	11	11	8	7	12	10	11
S	A	B	C	D	E	F	G	H	I	J	K	L	M

Iteration 1:

From S, we can go to A and B.

S—>A = 0+1 = 1

S—>B = 0+3 = 3

Hence, no change in the values.

0	1	3	5	5	18	6	11	11	8	7	12	10	11
S	A	B	C	D	E	F	G	H	I	J	K	L	M

Iteration 2:

From A, we can't go anywhere.

0	1	3	5	5	18	6	11	11	8	7	12	10	11
S	A	B	C	D	E	F	G	H	I	J	K	L	M

Iteration 3:

From B, we can go to C and D.

B—>C = 3+2 = 5

B—>D = 3+2 = 5

Hence, no change in the values.

0	1	3	5	5	18	6	11	11	8	7	12	10	11
S	A	B	C	D	E	F	G	H	I	J	K	L	M

Iteration 4:

From C, we can go to F , I and J.

C—>F = 5+1 = 6

C—>I = 5+3 = 8

C—>J = 5+2 = 7

Hence, no change in the values.

0	1	3	5	5	18	6	11	11	8	7	12	10	11
S	A	B	C	D	E	F	G	H	I	J	K	L	M

Iteration 5:

From D, we can't go anywhere.

0	1	3	5	5	∞	6	∞	∞	8	7	∞	∞	∞
S	A	B	C	D	E	F	G	H	I	J	K	L	M

Iteration 6:

From E, we can't go anywhere.

0	1	3	5	5	18	6	11	11	8	7	12	10	11
S	A	B	C	D	E	F	G	H	I	J	K	L	M

Iteration 7:

From F, we can go to G and H

F—>G = 6+5 = 11

F—>H = 6+5 = 11

Hence, no change in the values.

0	1	3	5	5	18	6	11	11	8	7	12	10	11
S	A	B	C	D	E	F	G	H	I	J	K	L	M

Iteration 8:

From G, we can't go anywhere.

0	1	3	5	5	18	6	11	11	8	7	12	10	11
S	A	B	C	D	E	F	G	H	I	J	K	L	M

Iteration 9:

From H, we can't go anywhere.

0	1	3	5	5	18	6	11	11	8	7	12	10	11
S	A	B	C	D	E	F	G	H	I	J	K	L	M

Iteration 10:

From I, we can't go anywhere.

0	1	3	5	5	18	6	11	11	8	7	12	10	11
S	A	B	C	D	E	F	G	H	I	J	K	L	M

Iteration 11:

From J, we can go to K and L.

J—>K = 7+5 = 12

J—>L = 7+3 = 10

Hence, no change in the values.

0	1	3	5	5	18	6	11	11	8	7	12	10	11
S	A	B	C	D	E	F	G	H	I	J	K	L	M

Iteration 12:

From K, we can't go anywhere.

0	1	3	5	5	18	6	11	11	8	7	12	10	11
S	A	B	C	D	E	F	G	H	I	J	K	L	M

Iteration 13:

From L, we can go to E and M.

L—>E= 10+8 = 18

L—>M= 10+1 = 11

Hence, no change in the values.

0	1	3	5	5	18	6	11	11	8	7	12	10	11
S	A	B	C	D	E	F	G	H	I	J	K	L	M

Iteration 14:

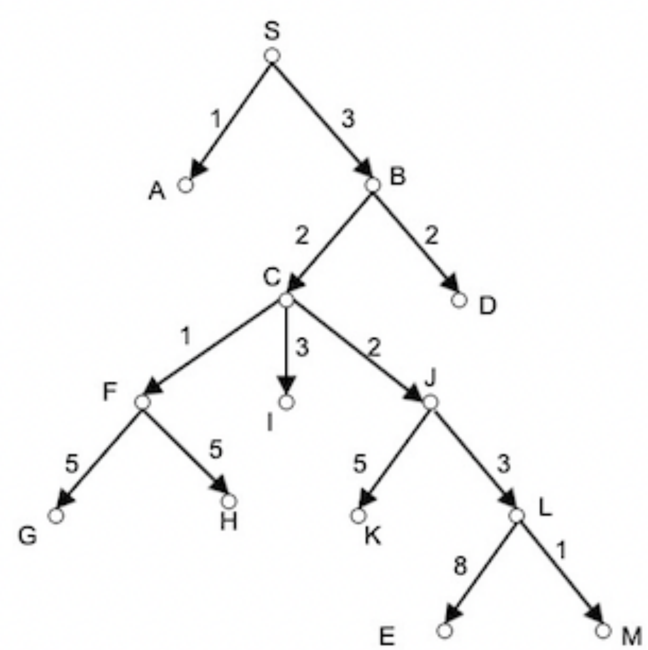
From M, we can't go anywhere.

0	1	3	5	5	18	6	11	11	8	7	12	10	11
S	A	B	C	D	E	F	G	H	I	J	K	L	M

Since, in this cycle no weights were updated, the process will stop here.

Hence, the weight of the shortest path from S to E is 18.

Dijkstra's Algorithm:



	INITIAL ()	VISITED: S	VISITED: S, A	VISITED: S, A, B	VISITED: S, A, B, C	VISITED: S, A, B, C, D	VISITED: S, A, B, C, D, F	VISITED: S, A, B, C, D, F, J	VISITED: S, A, B, C, D, F, J, I	VISITED: S, A, B, C, D, F, J, I, L	VISITED: S, A, B, C, D, F, J, I, L, G	VISITED: S, A, B, C, D, F, J, I, L, G, H	VISITED: S, A, B, C, D, F, J, I, L, G, H, M	VISITED: S, A, B, C, D, F, J, I, L, G, H, M, K	VISITED: VISITED: S, A, B, C, D, F, J, I, L, G, H, M, K , E
	NEXT STEP S	NEXT STEP A	NEXT STEP B	NEXT STEP C	NEXT STEP D	NEXT STEP F	NEXT STEP J	NEXT STEP I	NEXT STEP L	NEXT STEP G	NEXT STEP H	NEXT STEP M	NEXT STEP K	NEXT STEP E	End
		S	A	B	C	D	F	J	I	L	I	H	M	K	E
S	0	0	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~
A	∞	1	1	~~1~~	~~1~~	~~1~~	~~1~~	~~1~~	~~1~~	~~1~~	~~1~~	~~1~~	~~1~~	~~1~~	~~1~~
B	∞	3	3	3	~~3~~	~~3~~	~~3~~	~~3~~	~~3~~	~~3~~	~~3~~	~~3~~	~~3~~	~~3~~	~~3~~
C	∞	∞	∞	5	5	~~5~~	~~5~~	~~5~~	~~5~~	~~5~~	~~5~~	~~5~~	~~5~~	~~5~~	~~5~~
D	∞	∞	∞	5	5	5	~~5~~	~~5~~	~~5~~	~~5~~	~~5~~	~~5~~	~~5~~	~~5~~	~~5~~
F	∞	∞	∞	∞	6	6	6	~~6~~	~~6~~	~~6~~	~~6~~	~~6~~	~~6~~	~~6~~	~~6~~
G	∞	∞	∞	∞	∞	∞	11	11	11	11	11	~~11~~	~~11~~	~~11~~	~~11~~
H	∞	∞	∞	∞	∞	∞	11	11	11	11	11	11	~~11~~	~~11~~	~~11~~
I	∞	∞	∞	∞	8	8	8	8	8	~~8~~	~~8~~	~~8~~	~~8~~	~~8~~	~~8~~
J	∞	∞	∞	∞	7	7	7	7	~~7~~	~~7~~	~~7~~	~~7~~	~~7~~	~~7~~	~~7~~
K	∞	∞	∞	∞	∞	∞	∞	12	12	12	12	12	12	12	~~12~~
L	∞	∞	∞	∞	∞	∞	∞	10	10	10	~~10~~	~~10~~	~~10~~	~~10~~	~~10~~
M	∞	∞	∞	∞	∞	∞	∞	∞	∞	11	11	11	11	~~11~~	~~11~~
E	∞	∞	∞	∞	∞	∞	∞	∞	∞	18	18	18	18	18	18

V: the current visiting node

V: the next node to visit

~~V~~: this node has been visited

The shortest path from S to E has weight 18.(S—> B—> C—> J—> L—>E)

Step 2: Comparison of Big O of Bellman and Dijkstra Algorithm

Time Complexity of Dijkstra:

1. The complexity of this algorithm is fully dependent on the implementation of Extract-Min function.
2. If extract min function is implemented using linear search, the complexity of this algorithm is $O(V^2 + E)$.
3. A typical binary heap priority queue implementation has $O((|E|+|V|)\log|V|)$ time complexity.

Time Complexity of Bellman Ford: $O(|V||E|)$ complexity

- If we use Dijkstra with binary heap , the performance of this algorithm is much better than Bellman Ford. But Dijkstra has one big disadvantage that it cannot handle negative weights. Hence, in that case Bellman is used.
- But if there no negative weights in the graph Dijkstra is always preferred over Bellman.

Step Count for both the algorithm:

A step is defined as either comparing two numbers or replacing a number.

- No of step counts in Bellman Ford in all the cycles: 13 (no of replacements)
- No of step counts in Dijkstra: 13 (no of replacements)

Since the graph is sparse and there are no cycles , the step count for both the algorithm is same.