1. 0 [A] → [B] → [E]∧

1 [B] → [C]∧

2. [C] → [t]∧

3. [D] → [A] → [E]∧

4. [E] → [C] → [F] → [I]∧

5. [F] → [C] → [t]∧

6. [G] → [D] → [E] → [H]∧

7 [H] → [E] → [I]∧

8 [I] → [F] → [F]∧

9 [S] → [A] → [D] → [G]∧

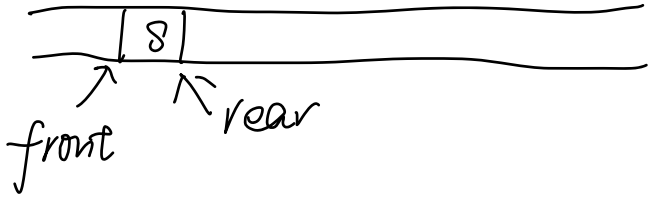
10. [t]∧

Indegree.

A	B	C	D	E	F	G	H	I	S	t
2	1	3	2	4	2	1	1	2	0	3

If indegree is 0, enqueue it.

Queue.



① Dequeue(S)

Output: S

Indegree

A	B	C	D	E	F	G	H	I	S	t
1	1	3	1	4	2	0	1	2	0	3

Enq G.

Q. rear

G

front

② Deq G.

Output S. G.

Inde

1	1	3	0	3	2	0	0	2	0	3
A	B	C	D	E	F	G	H	I	S	t.

Enq: D. H.

Queue.

D	H

front rear.

③ Deq D.

Output S G D

Indegree

A	B	C	D	E	F	G	H	I	S	t.
0	1	3	0	2	2	0	0	2	0	3

④ Dequeue I.

Output SGDHABEI.

Indegree

A	B	C	D	E	F	G	H	I	S	t
0	0	1	0	0	1	0	0	0	0	2

Enqueue F.

Queue.

F

front rear.

⑤ Dequeue (F)

Output SGDHABEIF.

Enqueue C.

Queue.

C

fr

⑥ Dequeue C.

Output SGDHABEIF.

Enqueue t.

Q. rear.

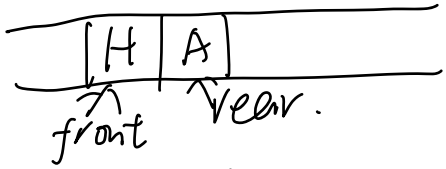
t

front

The result. SGDHABEIFCt.

Enqueue A.

Q.



④ Deq (H)

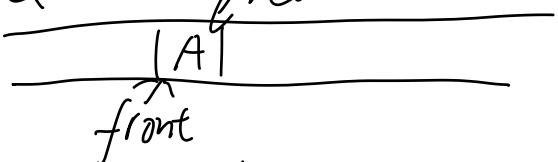
Output SGDH.

Indegree

A	B	C	D	E	F	G	H	I	S	t
0	1	3	0	1	2	0	0	1	0	3

No equeue.

Queue



⑭ Dequeue A.

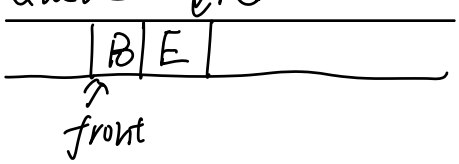
Output. SGDHA.

Indegree

A	B	C	D	E	F	G	H	I	S	t
0	0	3	0	0	2	0	0	1	0	3

Enqueue B.E.

Queue



⑮ Dequeue B.

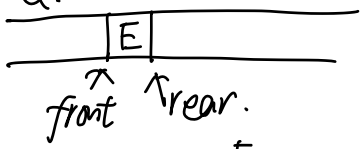
Output SGDHAB

Indegree

A	B	C	D	E	F	G	H	I	S	t
0	0	2	0	0	2	0	0	1	0	3

no enqueue

Q.



⑥ Dequeue E.

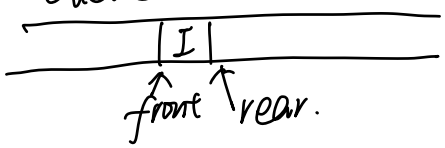
output S G D H A B E.

Indegree.

A	B	C	D	E	F	G	H	I	S	t
0	0	1	0	0	1	0	0	0	0	B

Enqueue I.

Queue

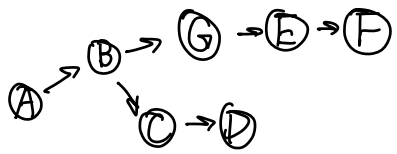


2.(a)

	B	C	D	E	F	G	S
Dist	5	3	∞	∞	∞	∞	
Init Pre	~~A~~	A	A	A	A	A	{A}
1. D	5		10	10	∞	∞	
1. P.	A		C	C	A	A	{A, C}
2. D			10	8	∞	6	
2. P			C	B	A	~~B~~	{A, C, B}
3. D			10	7	∞		
3. P.			C	G	A		{A, C, B, G}
4. D			10		8		
4. P			C		E		{A, C, B, G, E}

5. P. D 10 C — A C B G E F

6. A C B G E F C.

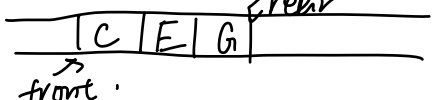


A→B:5 A→C:3 A→D 10 A→E 7 A→F 8 A→G:6.

b) ① Dequeue D

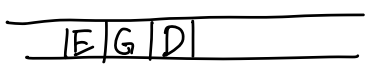
	A	B	C	D	E	F	G.
D	∞	5	3	∞	7	∞	6
P	0	0	B	0	B	0	B.

CEG will be enqueued



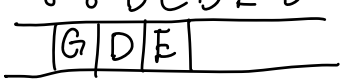
② Dequeue C.

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



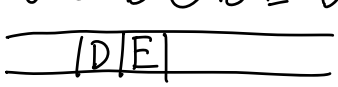
③ Dequeue E.

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



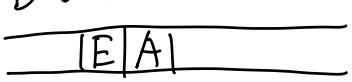
④ Dequeue G

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



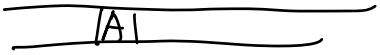
⑤ Dequeue D

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



⑥ Dequeue E

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



⑦ Dequeue A

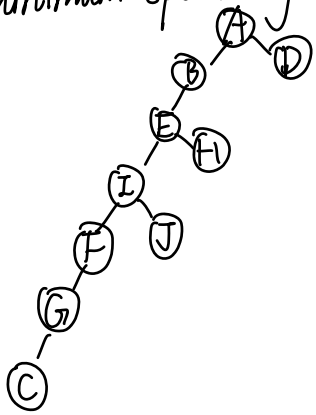
A	B	C	D	E	F	G
3	5	3	2	7	8	6
D	0	B	C	B	E	B

B→A:3
B→C:1
B→D:2
B→E:1
B→F:2
B→G:1

3. Assume A is the root of the minimum spanning tree.

	B	C	D	E	F	G	H	I	J	S
Dist	3	∞	4	4	∞	∞	∞	∞	∞	
Path	A	0	A	A	0	0	0	0	0	{A}
Known	0	0	0	0	0	0	0	0	0	
	3 A 1	10 B 0	4 A 0	2 B 0	3 B 0	∞ 0	∞ 0	∞ 0	∞ 0	{A,B}
		10 B 0	4 A 0	2 B 1	3 B 0	∞ 0	2 E 0	1 E 0	∞ 0	{A,B,E}
		10 B 0	4 A 0		3 I 0	∞ 0	2 E 0	1 E 1	7 I 0	{A,B,E,I}
		10 B 0	4 A 0		3 I 0	∞ 0	2 E 1		7 I 0	{A,B,E,I,H}
		6 F 0	4 A 0		3 I 1	2 F 0			7 I 0	{A,B,E,I,H,F}
		1 G 0	4 A 0			2 F 1			7 I 0	{A,B,E,I,H,F,G}
		1 G 1	4 A 0						7 I 0	{A,B,E,I,H,F,G,C}
			4 A 1						7 I 0	{A,B,E,I,H,F,G,C,D}
									7 I 1	{A,B,E,I,H,F,G,C,D,J}

The minimum spanning tree



4. The original Dijsktra is finding every node can be arrived from root using BFS. and add the shortest path into the final path. We can use a priority queue to solve this problem. Instead of adding the shortest path every step. we add the max value of the min value of weight of edge (u,v). the u position of the widest list and the v position of the widest list into the widest list. V is the node we are checking and u is the previous node of V..