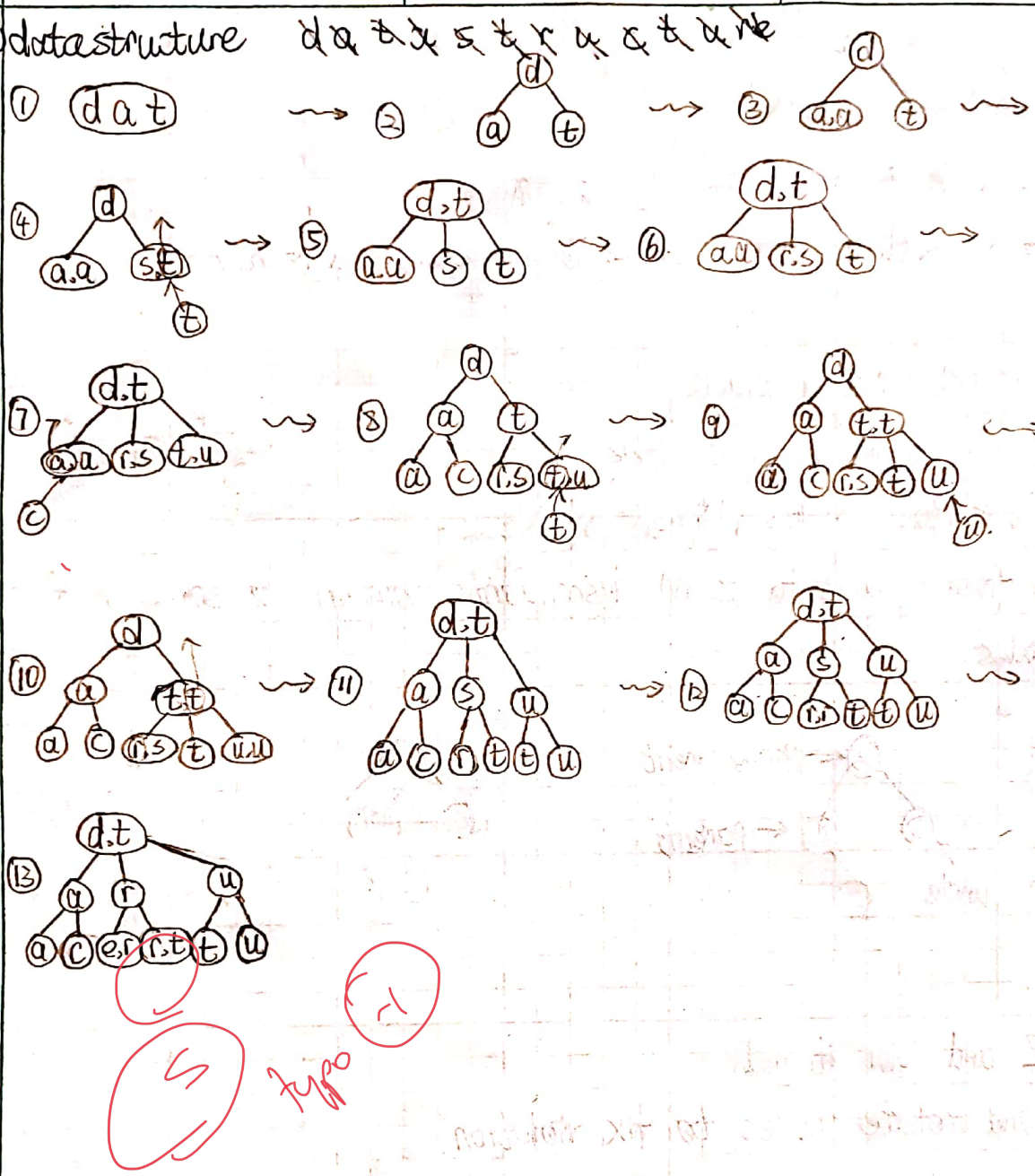
a c d e r s t u

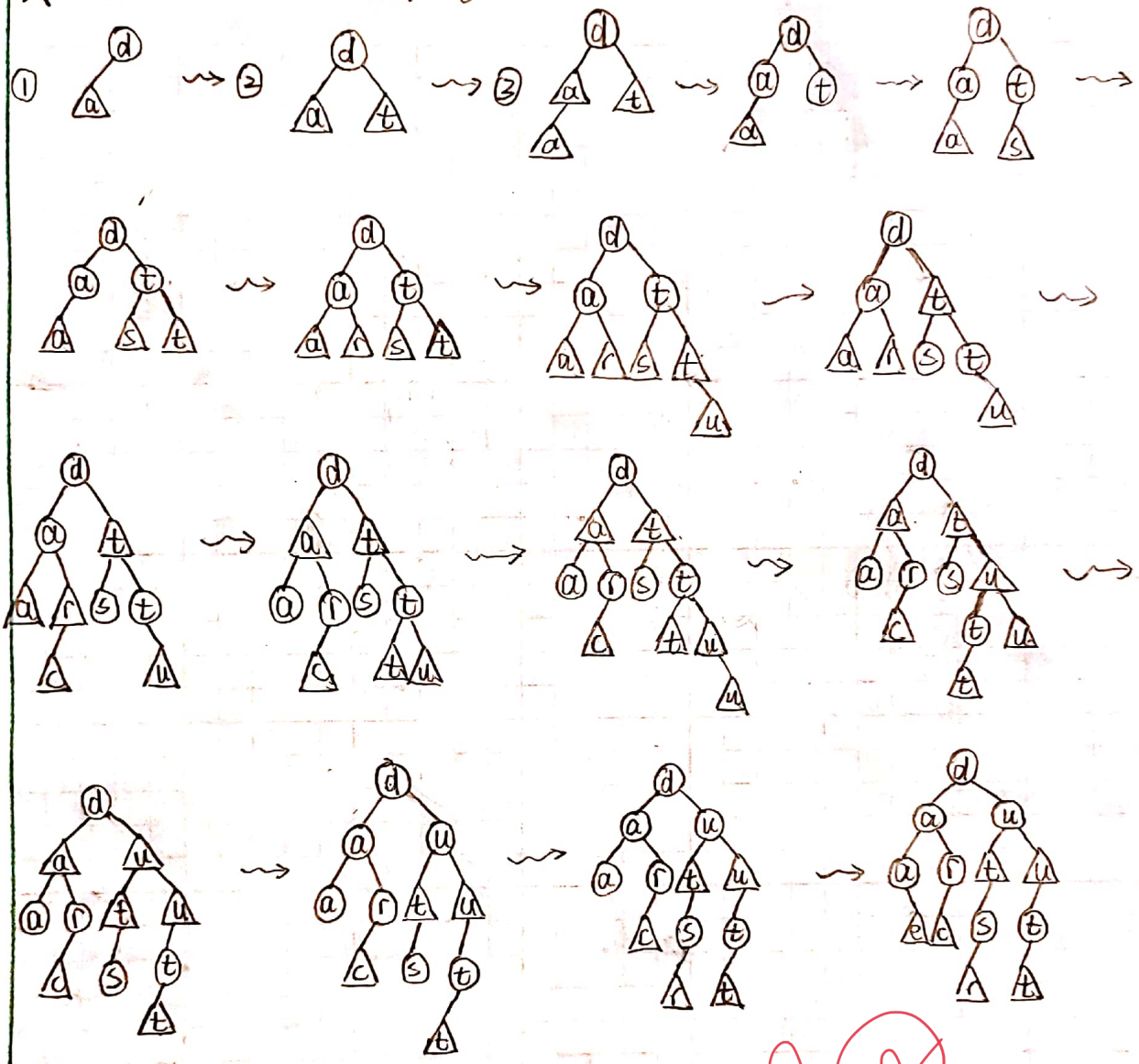
1.(1) data structure



a c d e r s t u

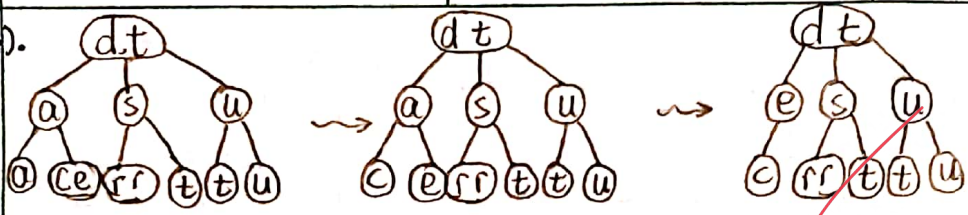
1.(2) d a t a s t r u c t u r e

* ○ ⇝ means black ; △ ⇝ means red.



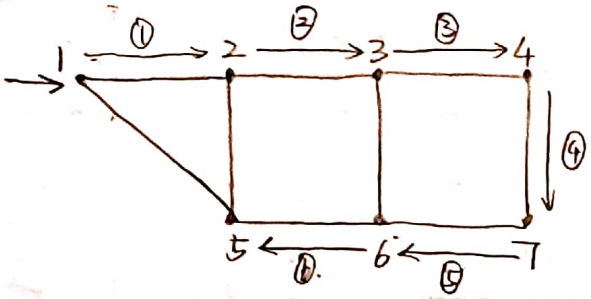
(1.2)

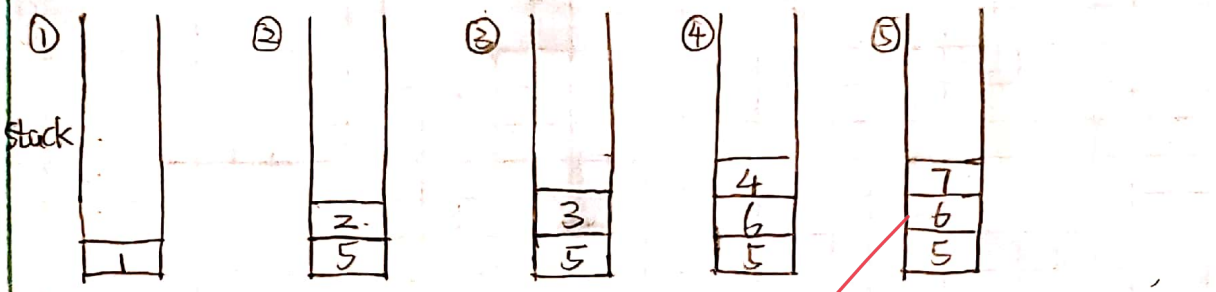
1.(3).

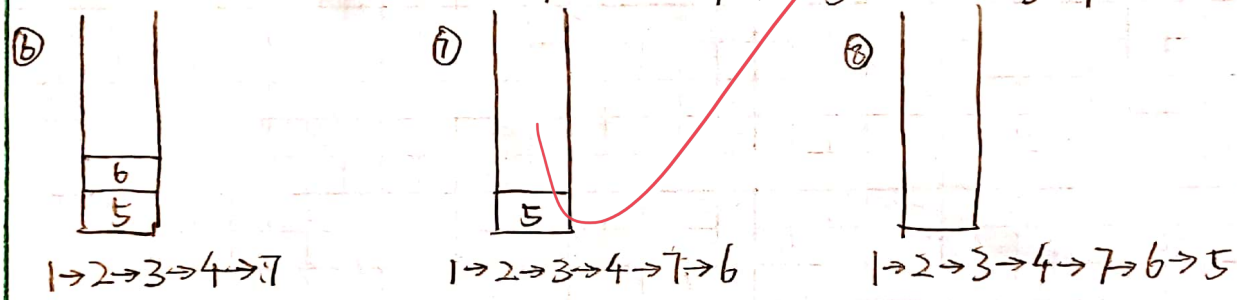


-5

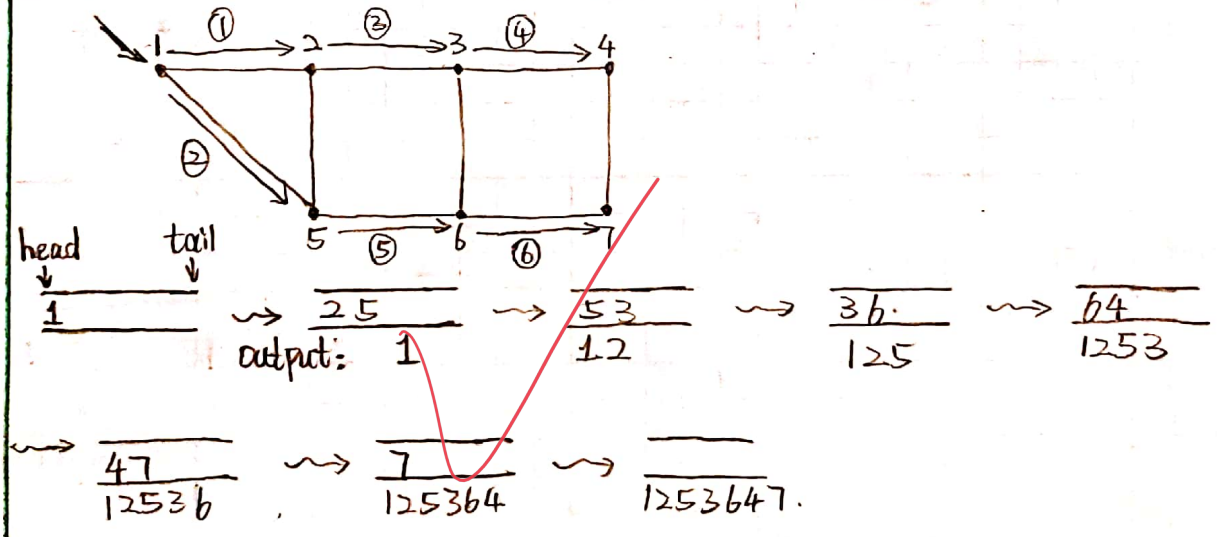
2. Depth-First Search:



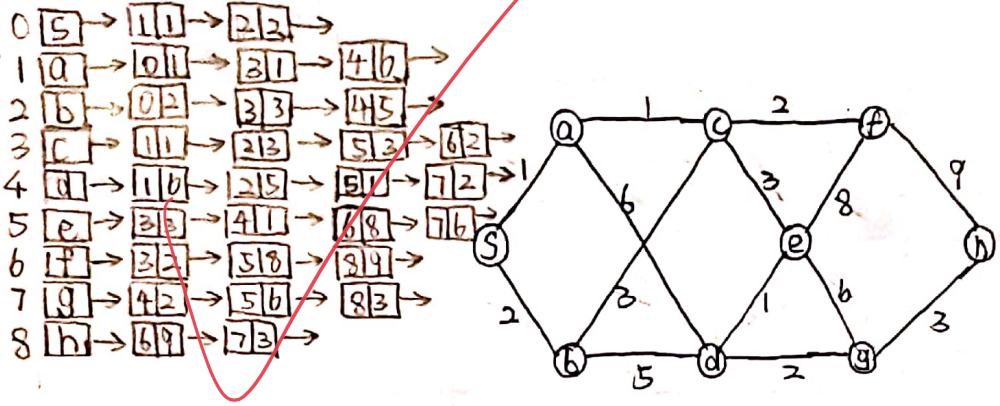




Breadth-First Search



3.(1).



(2). Kruskal's algorithm:

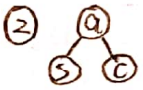
Edge	Weight
s-a	1
s-b	2
b-c	3
a-b	6
a-c	1
b-d	5
c-f	2
c-e	3
e-f	8
d-e	1
e-g	6
d-g	2
f-h	9
g-h	3

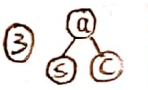
Sort →

Edge	Weight
✓ s-a	1
✓ a-c	1
✓ d-e	1
✓ s-b	2
✓ c-f	2
✓ d-g	2
✗ b-c	3
✓ c-e	3
✓ g-h	3
✗ b-d	5
✗ a-b	6
✗ e-g	6
✗ e-f	8
✗ f-h	9

①  {{s,a}}

Union 

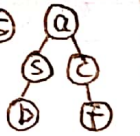
②  {{s,a,c}}

③  {{s,a,c},{d,e}}

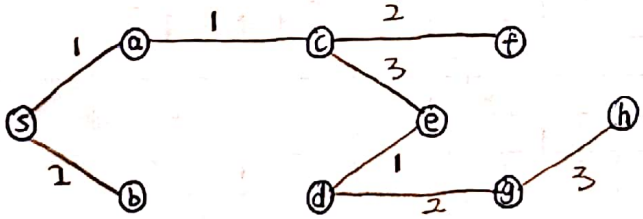
Union 

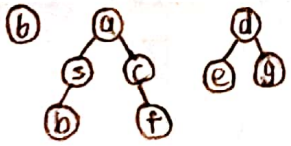
④  {{s,a,b,c},{d,e}}

Union 

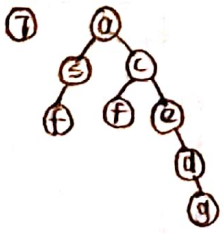
⑤  {{s,a,b,c,f},{d,e}}

Union 



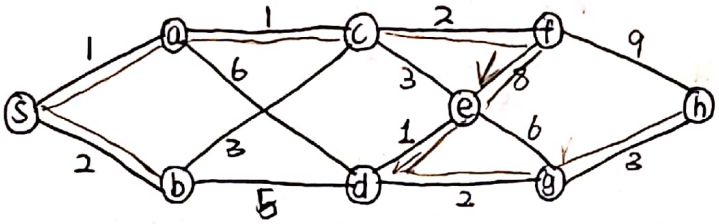
⑥  {{s,a,b,c,f},{d,e,g}}

Union 

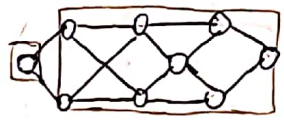
⑦  {{s,a,b,c,d,e,f,g}}

Union 

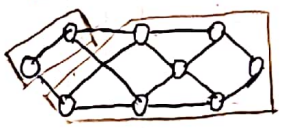
3.3 Prim's Algorithm:



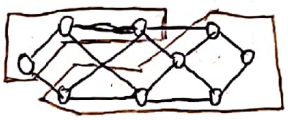
① node s a b c d e f g h
min 1 2
Pre s s



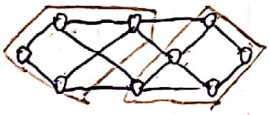
② node s a b c d e f g h
min - 2 1 6
Pre s s a a



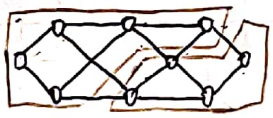
③ node s a b c d e f g h
min - 2 - 6 3 2
Pre s s a a c c



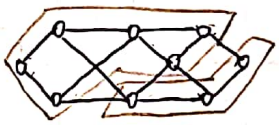
④ node s a b c d e f g h
min - - - 6 3 2
Pre s s a b c c



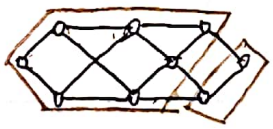
⑤ node s a b c d e f g h
min - - - 6 3 - 9
Pre s s a b f c f



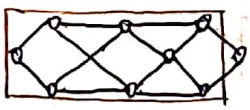
⑥ node s a b c d e f g h
min - - - 1 - - 6 9
Pre s s a e f c e f



⑦ node s a b c d e f g h
min - - - - - - 2 9
Pre s s a e f c d f



⑧ node s a b c d e f g h
min - - - - - - - 3
Pre s s a e f c d g



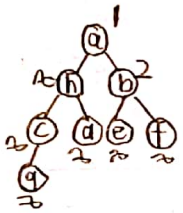
See next page for heap & in-heap

3B).

	s	a	b	c	d	e	f	g	h
Inheap	1	1	1	1	1	1	1	1	1
key	0	∞	∞	∞	∞	∞	∞	∞	∞
Pred	—	—	—	—	—	—	—	—	—
Index	1	2	3	4	5	6	7	8	9

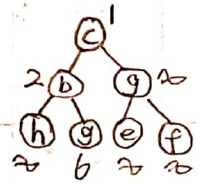
①

	s	a	b	c	d	e	f	g	h
Inheap	0	1	1	1	1	1	1	1	1
key	1	2	∞	∞	∞	∞	∞	∞	∞
Pred	—	s	s	—	—	—	—	—	—
Index	—	1	3	4	5	6	7	8	2



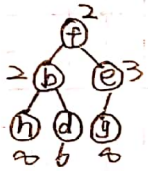
②

	s	a	b	c	d	e	f	g	h
Inheap	0	0	1	1	1	1	1	1	1
key	0	1	2	1	6	—	—	—	—
Pred	—	s	s	a	a	—	—	—	—
Index	—	—	2	1	5	6	7	3	4



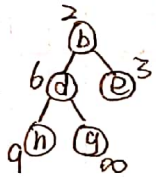
③

	s	a	b	c	d	e	f	g	h
Inheap	0	0	1	0	1	1	1	1	1
key	0	1	2	1	6	3	2	∞	∞
Pred	—	s	s	a	a	c	c	—	—
Index	—	—	2	—	5	3	1	6	4



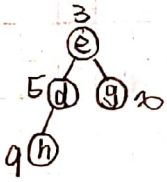
④

	s	a	b	c	d	e	f	g	h
Inheap	0	0	1	0	1	1	0	1	1
key	0	1	2	1	6	3	2	∞	9
Pred	—	s	s	a	a	c	c	—	f
Index	—	—	1	—	2	3	—	5	4



⑤

	s	a	b	c	d	e	f	g	h
Inheap	0	0	0	0	1	1	0	1	1
key	0	1	2	1	5	3	2	∞	9
Pred	—	s	s	a	b	c	c	—	f
Index	—	—	—	—	2	1	—	3	4

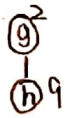


⑥

	s	a	b	c	d	e	f	g	h
Inheap	0	0	0	0	1	0	0	1	1
key	0	1	2	1	5	3	2	6	9
Pred	—	s	s	a	b	c	c	e	f
Index	—	—	—	—	1	—	—	3	2

⑦

	s	a	b	c	d	e	f	g	h
Inheap	0	0	0	0	0	0	0	1	1
key	0	1	2	1	5	3	2	2	9
Pred	—	s	s	a	b	c	c	d	f
Index	—	—	—	—	—	—		1	2

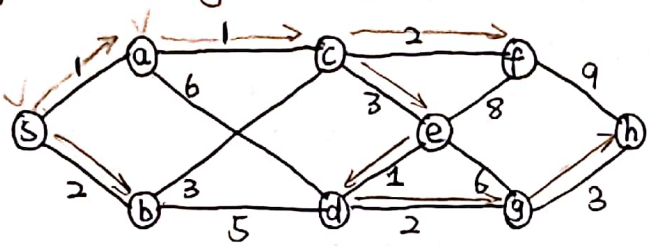


⑧

	s	a	b	c	d	e	f	g	h
Inheap	0	0	0	0	0	0	0	0	1
key	0	1	2	1	5	3	2	2	3
Pred	-	s	s	a	b	c	c	d	g
Index	-	-	-	-	-	-	-	-	1

⑧ h 3

3.(b). Dijkstra's Algorithm to find shortest paths.



Find shortest Path from S to h

	A	B	C	D	e	f	g	h
Predecesar	S	S	a	e	C	C	d	g

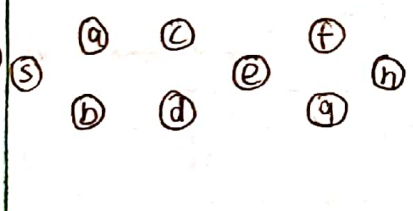
s → a:1, b:2	e→d:6
a → c:1, d:7	e→f:13, e→g:14
b → c:5, d:7	f→h:13
c → f:4, e:5	g→h:12
d → e:8, g:9	
f→e:12	
g→e:15	

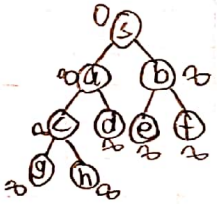
Priority Queue:

a,1 | b,2 | c,2 | f,4 | c,5 | e,5 | d,6 | d,7 | d,7 | e,8 | g,9 | e,12 | h,12

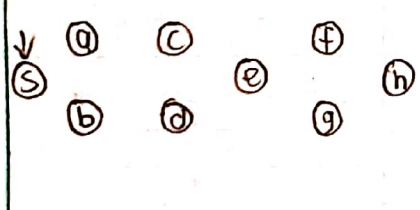
f,13 | h,13 | g,14 | e,15

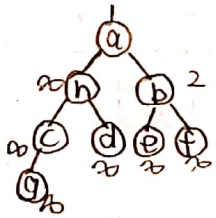
⇒ a,1 | b,2 | c,2 | f,4 | e,5 | d,6 | g,9 | h,12

① 

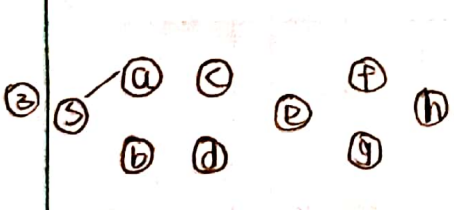


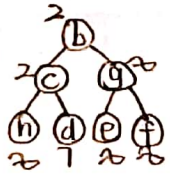
	s	a	b	c	d	e	f	g	h
Inheap	1	1	1	1	1	1	1	1	1
key	0	∞	∞	∞	∞	∞	∞	∞	∞
Pred	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ
Index	1	2	3	4	5	6	7	8	9

② 

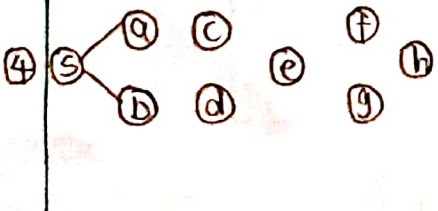


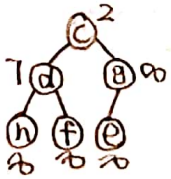
	s	a	b	c	d	e	f	g	h
Inheap	0	1	1	1	1	1	1	1	1
key	0	1	2	∞	∞	∞	∞	∞	∞
Pred	ϕ	s	s	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ
Index	ϕ	1	3	4	5	6	7	8	2

③ 

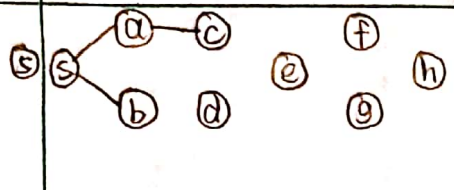


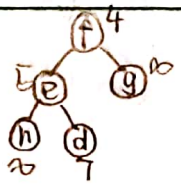
	s	a	b	c	d	e	f	g	h
Inheap	0	0	1	1	1	1	1	1	1
key	0	1	2	2	7	∞	∞	∞	∞
pred	ϕ	s	s	a	a	ϕ	ϕ	ϕ	ϕ
Index	ϕ	ϕ	1	2	5	6	7	3	4

④ 

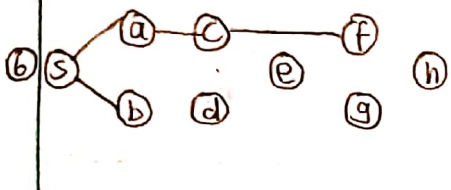


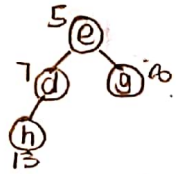
	s	a	b	c	d	e	f	g	h
Inheap	0	0	0	1	1	1	1	1	1
key	0	1	2	2	7	∞	∞	∞	∞
Pred	ϕ	s	s	a	a	ϕ	ϕ	ϕ	ϕ
Index	ϕ	ϕ	ϕ	1	2	6	5	3	4



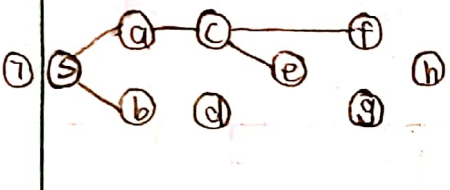


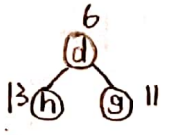
	s	a	b	c	d	e	f	g	h
Inheap	0	0	0	0	1	1	1	1	1
key	0	1	2	2	7	5	4	∞	∞
Pred	ϕ	s	s	a	a	c	c	ϕ	ϕ
Index	ϕ	ϕ	ϕ	ϕ	5	2	1	3	4



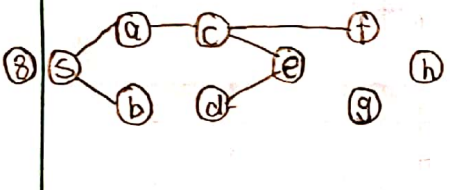


	s	a	b	c	d	e	f	g	h
Inheap	0	0	0	0	1	1	0	1	1
key	0	1	2	2	7	5	4	∞	13
Pred	ϕ	s	s	a	a	c	c	ϕ	f
Index	ϕ	ϕ	ϕ	ϕ	2	1	ϕ	3	4

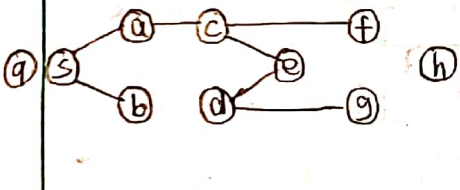




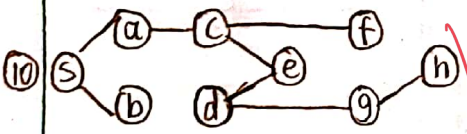
	s	a	b	c	d	e	f	g	h
Inheap	0	0	0	0	1	0	0	1	1
key	0	1	2	2	6	5	4	11	13
Pred	ϕ	s	s	a	e	c	c	e	f
Index	ϕ	ϕ	ϕ	ϕ	1	ϕ	ϕ	3	2



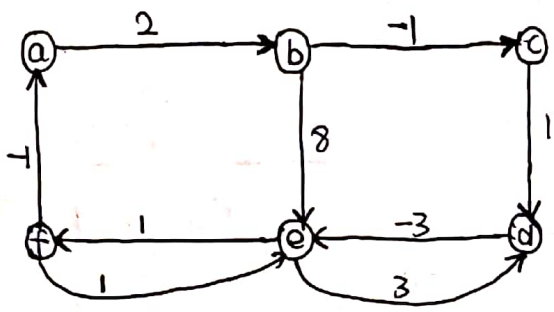
	s	a	b	c	d	e	f	g	h
Inheap	0	0	0	0	0	0	0	1	1
key	0	1	2	2	6	5	4	8	13
Pred	ϕ	s	s	a	e	c	c	d	f
Index	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	1	2



	s	a	b	c	d	e	f	g	h
Inheap	0	0	0	0	0	0	0	0	1
key	0	1	2	2	6	5	4	8	13
Pred	ϕ	s	s	a	e	c	c	d	g
Index	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	1



4.



Bellman-Ford:

1st iteration:

ab	a 2
bc	b 1
cd	c 2
de	d -1
ef	e 0
fa	f -1
fe	—
ed	—
be	—

2nd iteration:

a	1
b	0
c	1
d	-2
e	-1
f	-2
	—
	—
	—

3rd iteration:

a	0
b	-1
c	0
d	-3
e	-2
f	-3
	—
	—
	—

4th iteration:

a	-1
b	-2
c	-1
d	-4
e	-3
f	-4
	—
	—
	—

5th iteration:

a	-2
b	-3
c	-2
d	-5
e	-4
f	-5
	—
	—
	—

6th iteration:

a	-3
b	-4
c	-3
d	-6
e	-5
f	-6
	—
	—
	—

This graph has a negative cycle because the number of the iterations is greater than N-1.