Graph – Dijkstra's algorithm

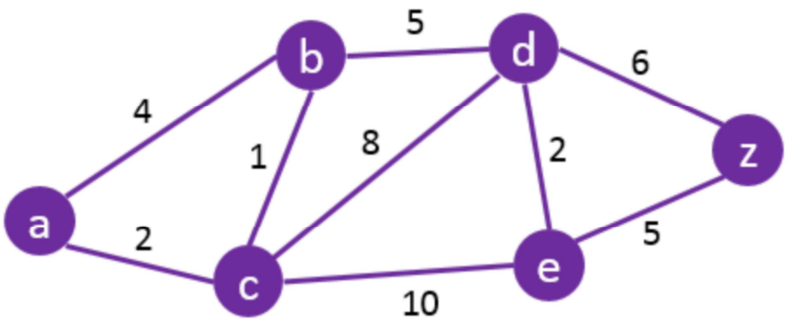
- Finding the shortest path

Note:

- **Must use a table**.
- **Show the shortest path as a conclusion**.

Graph – Dijkstra's algorithm

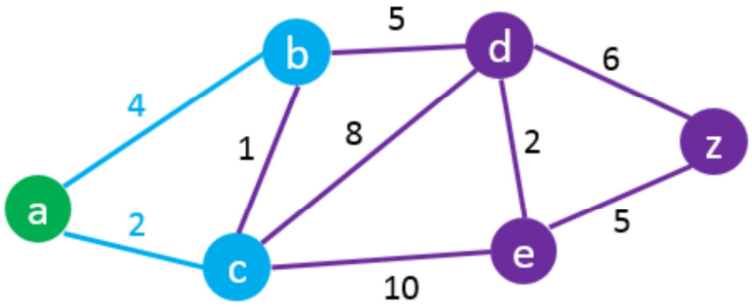
- Apply the **Dijkstra's algorithm** to find the shortest path from a to z



T	V_i	a	b	c	d	e	z
{a,b,c,d,e,z}	-	0	∞	∞	∞	∞	∞

Graph – Dijkstra's algorithm

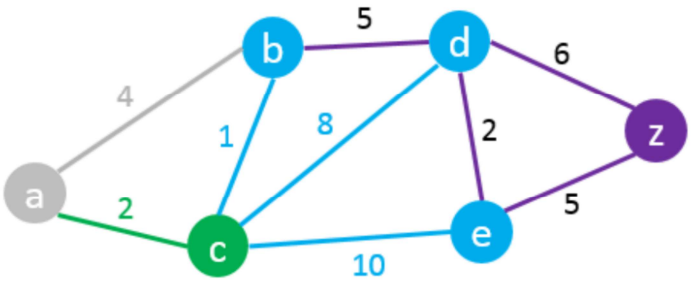
- Apply the **Dijkstra's algorithm** to find the shortest path from a to z



T	V_i	a	b	c	d	e	z
{a,b,c,d,e,z}	-	0	∞	∞	∞	∞	∞
{b,c,e,z}	a	*	(4,a)	**(2,a)**	∞	∞	∞

Graph – Dijkstra's algorithm

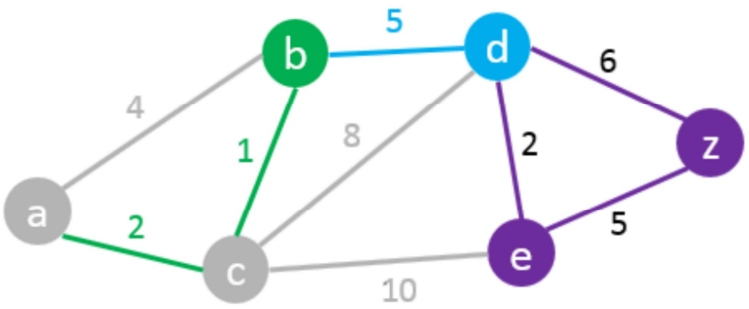
- Apply the **Dijkstra's algorithm** to find the shortest path from a to z



T	V_i	a	b	c	d	e	z
{a,b,c,d,e,z}	-	0	∞	∞	∞	∞	∞
{b,c,e,z}	a	-	(4,a)	**(2,a)**	∞	∞	∞
{b,e,z}	c	-	(3,c)	*	(10,c)	(12,c)	∞

Graph – Dijkstra's algorithm

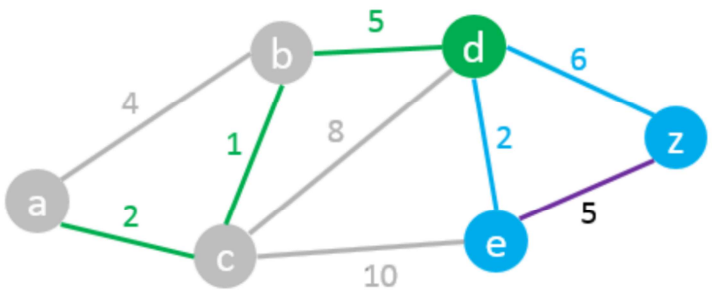
- Apply the **Dijkstra's algorithm** to find the shortest path from a to z



T	V_i	a	b	c	d	e	z
{a,b,c,d,e,z}	-	0	∞	∞	∞	∞	∞
{b,c,d,e,z}	a	-	(4,a)	**(2,a)**	∞	∞	∞
{b,d,e,z}	c	-	**(3,c)**	-	(10,c)	(12,c)	∞
{d,e,z}	b	-	*	-	(8,b)	(12,c)	∞

Graph – Dijkstra's algorithm

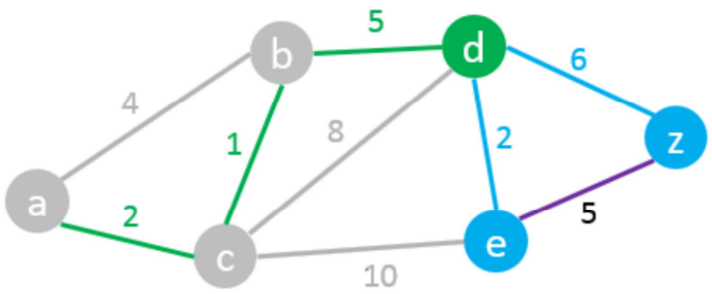
- Apply the **Dijkstra's algorithm** to find the shortest path from a to z



T	V_i	a	b	c	d	e	z
{a,b,c,d,e,z}	-	0	∞	∞	∞	∞	∞
{b,c,d,e,z}	a	-	(4,a)	**(2,a)**	∞	∞	∞
{b,d,e,z}	c	-	**(3,c)**	-	(10,c)	(12,c)	∞
{d,e,z}	b	-	-	-	**(8,b)**	(12,c)	∞
{e,z}	d	-	-	-	*	(10,d)	(14,d)

Graph – Dijkstra's algorithm

- Apply the **Dijkstra's algorithm** to find the shortest path from a to z



T	V_i	a	b	c	d	e	z
{a,b,c,d,e,z}	-	0	∞	∞	∞	∞	∞
{b,c,d,e,z}	a	-	(4,a)	**(2,a)**	∞	∞	∞
{b,d,e,z}	c	-	**(3,c)**	-	(10,c)	(12,c)	∞
{d,e,z}	b	-	-	-	**(8,b)**	(12,c)	∞
{e,z}	d	-	-	-	-	**(10,d)**	(14,d)
{z}	e	-	-	-	-	*	(14,d)

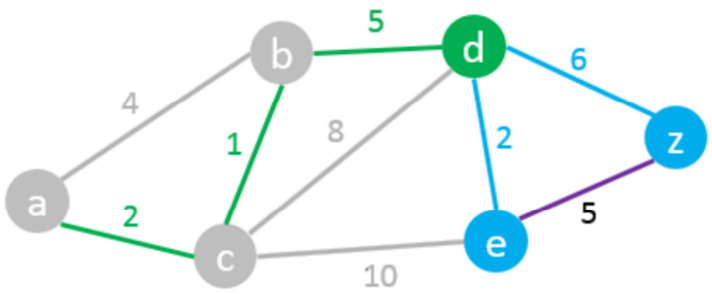
The shortest path from a to z is a –> c –> b –> d –> z with the length is 14

Note:

- End node = z -> previous node = d
- Current node = d -> previous node = b
- Current node = b -> previous node = c
- Current node = c -> Start node = a

Graph – Dijkstra's algorithm

- Apply the **Dijkstra's algorithm** to find the shortest path from a to z



T	V_i	a	b	c	d	e	z
{a,b,c,d,e,z}	-	0	∞	∞	∞	∞	∞
{b,c,d,e,z}	a	-	(4,a)	**(2,a)**	∞	∞	∞
{b,d,e,z}	c	-	**(3,c)**	-	(10,c)	(12,c)	∞
{d,e,z}	b	-	-	-	**(8,b)**	(12,c)	∞
{e,z}	d	-	-	-	-	**(10,d)**	(14,d)
{z}	e	-	-	-	-	-	(14,d)

Question: What is the shortest path from a to e ?

End node = e -> previous node = d
Current node = d -> previous node = b
Current node = b -> previous node = c
Current node = c -> Start node = a

Shortest path from a to e is a –> c –> b –> d –> e with length is 10

Graph – Prim algorithm

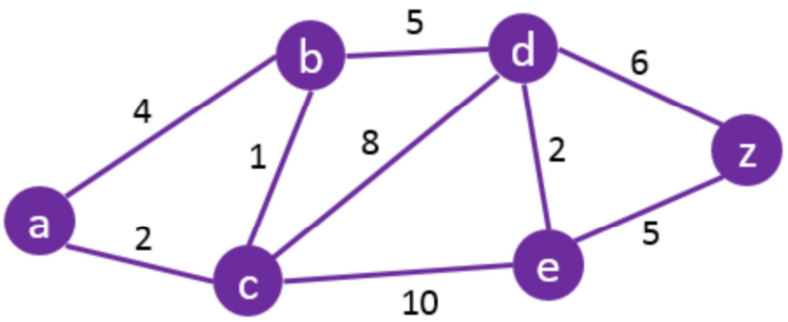
- Find the minimum spanning tree

Note:

- **Must use a table to show**.
- **Write a conclusion at the end.**

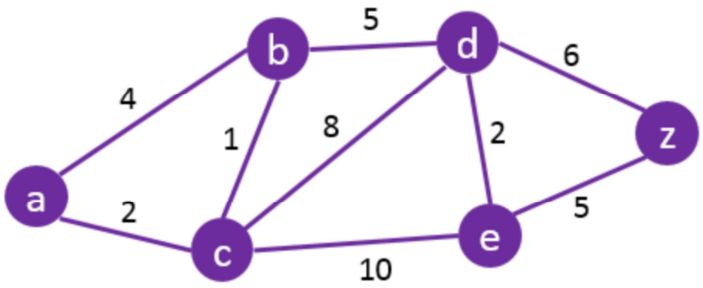
Graph – Prim algorithm

- Find the minimum spanning tree (start from a)



Graph – Prim algorithm

- Find the minimum spanning tree (start from a)

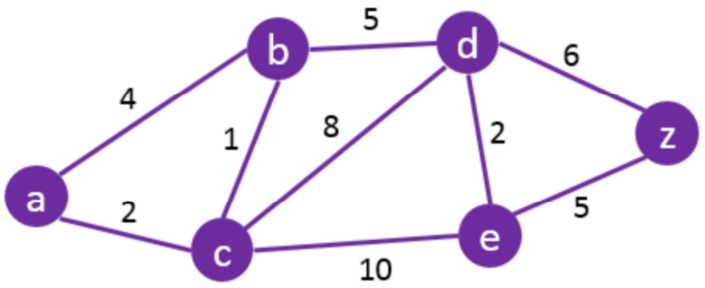


T	a	b	c	d	e	z
-	*	(4,a)	**(2,a)**	∞	∞	∞

Minimum Spanning tree

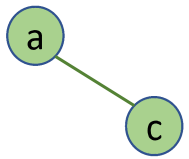
Graph – Prim algorithm

- Find the minimum spanning tree (start from a)



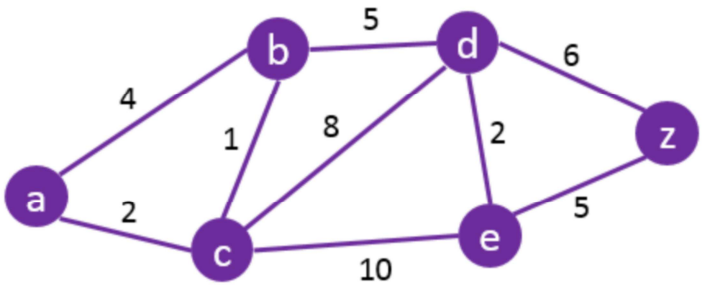
T	a	b	c	d	e	z
-	*	(4,a)	**(2,a)**	∞	∞	∞
(a,c)	-	(1,c)	*	(8,c)	(10,c)	∞

Minimum Spanning tree



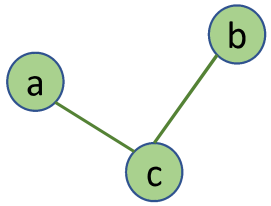
Graph – Prim algorithm

- Find the minimum spanning tree (start from a)



T	a	b	c	d	e	z
-	*	(4,a)	**(2,a)**	∞	∞	∞
(a,c)	-	**(1,c)**	*	(8,c)	(10,c)	∞
(b,c)	-	*	-	(5,b)	(8,c)	∞

Minimum Spanning tree



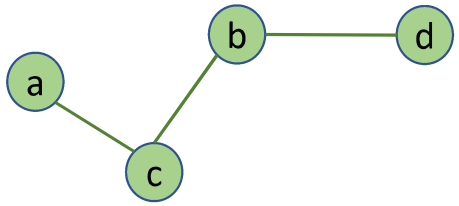
Graph – Prim algorithm

- Find the minimum spanning tree (start from a)

T	a	b	c	d	e	z
-	*	(4,a)	**(2,a)**	∞	∞	∞
(a,c)	-	**(1,c)**	*	(8,c)	(10,c)	∞
(b,c)	-	*	-	**(5,b)**	(8,c)	∞
(b,d)	-	-	-	*	(2,d)	(6,d)

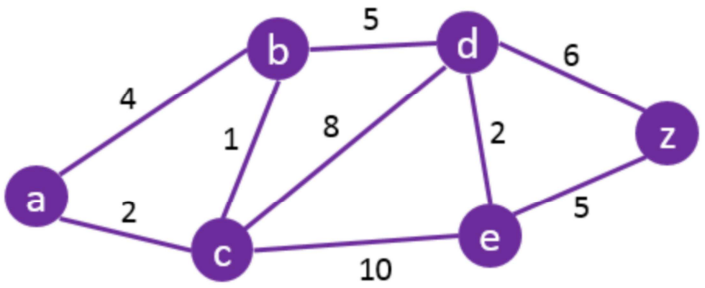
T	a	b	c	d	e	z
-	*	(4,a)	**(2,a)**	∞	∞	∞
(a,c)	-	**(1,c)**	*	(8,c)	(10,c)	∞
(b,c)	-	*	-	**(5,b)**	(8,c)	∞
(b,d)	-	-	-	*	(2,d)	(6,d)

Minimum Spanning tree



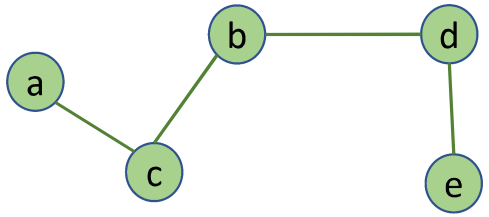
Graph – Prim algorithm

- Find the minimum spanning tree (start from a)



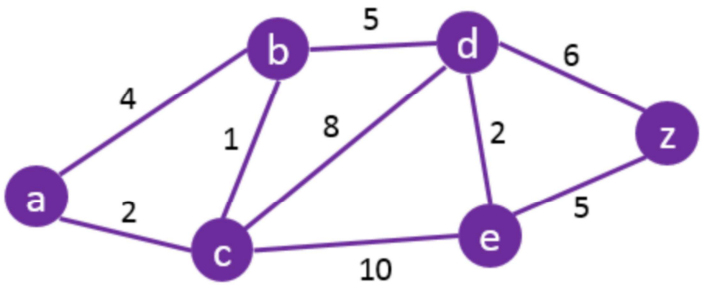
T	a	b	c	d	e	z
-	*	(4,a)	**(2,a)**	∞	∞	∞
(a,c)	-	**(1,c)**	*	(8,c)	(10,c)	∞
(b,c)	-	*	-	**(5,b)**	(8,c)	∞
(b,d)	-	-	-	*	**(2,d)**	(6,d)
(d,e)	-	-	-	-	*	(5,e)

Minimum Spanning tree



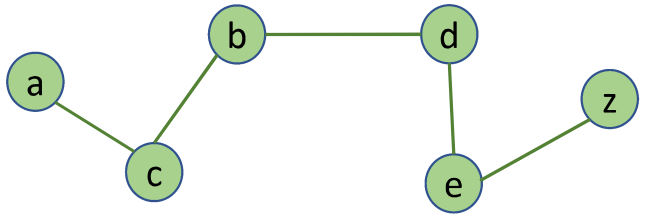
Graph – Prim algorithm

- Find the minimum spanning tree (start from a)



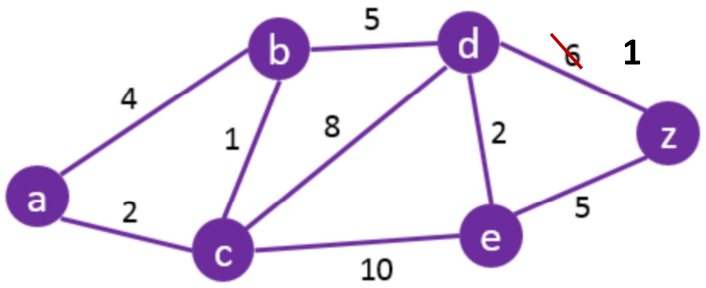
T	a	b	c	d	e	z
-	*	(4,a)	**(2,a)**	∞	∞	∞
(a,c)	-	**(1,c)**	*	(8,c)	(10,c)	∞
(b,c)	-	*	-	**(5,b)**	(8,c)	∞
(b,d)	-	-	-	*	**(2,d)**	(6,d)
(d,e)	-	-	-	-	*	(5,e)

Minimum Spanning tree



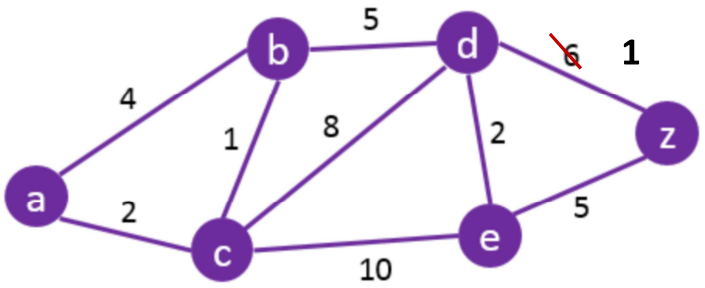
Graph – Prim algorithm

- Find the minimum spanning tree (start from a)



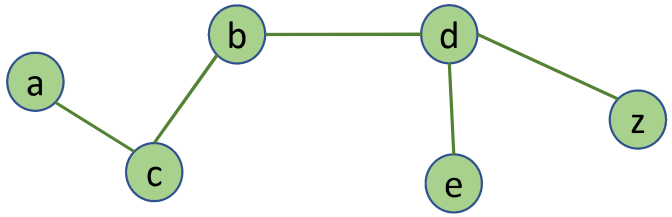
Graph – Prim algorithm

- Find the minimum spanning tree (start from a)



T	a	b	c	d	e	z
-	*	(4,a)	**(2,a)**	∞	∞	∞
(a,c)	-	**(1,c)**	*	(8,c)	(10,c)	∞
(b,c)	-	*	-	**(5,b)**	(8,c)	∞
(b,d)	-	-	-	*	(2,d)	**(1,d)**
(d,z)	-	-	-	-	(2,d)	*

Minimum Spanning tree

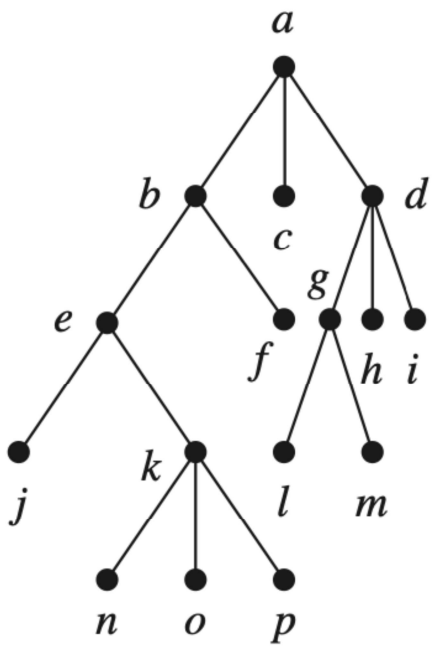


Tree

- Traversal algorithms
- Represent the expression using binary tree, prefix notation, postfix notation, infix notation
- Produce spanning tree using Deep-first search / Bread-first search

Tree – Traversal algorithm

- Determine the order in which a **Pre-order**, an **In-order**, and a **Post-order** traversal visits the vertices of the given ordered rooted tree as follows



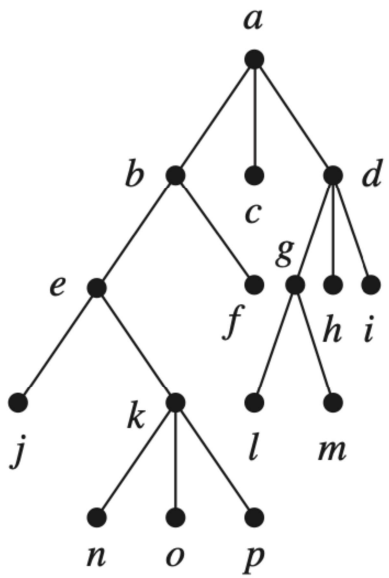
Pre-order: root – left –right

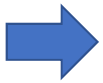
In-order: left – root – right

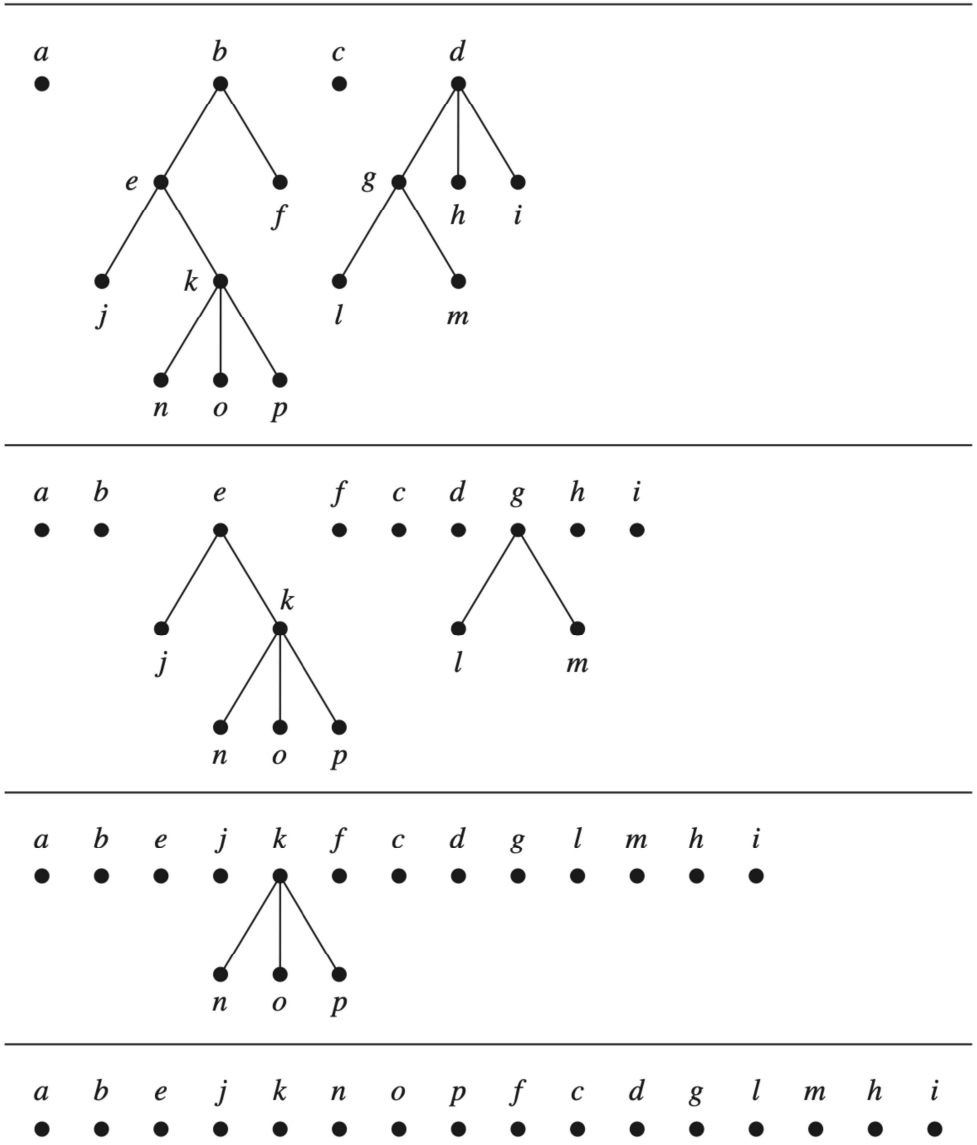
Post-order: left –right - root

Tree – Traversal algorithm

Pre-order: root – left –right

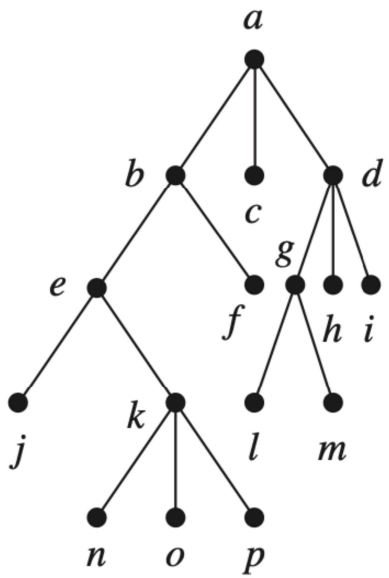


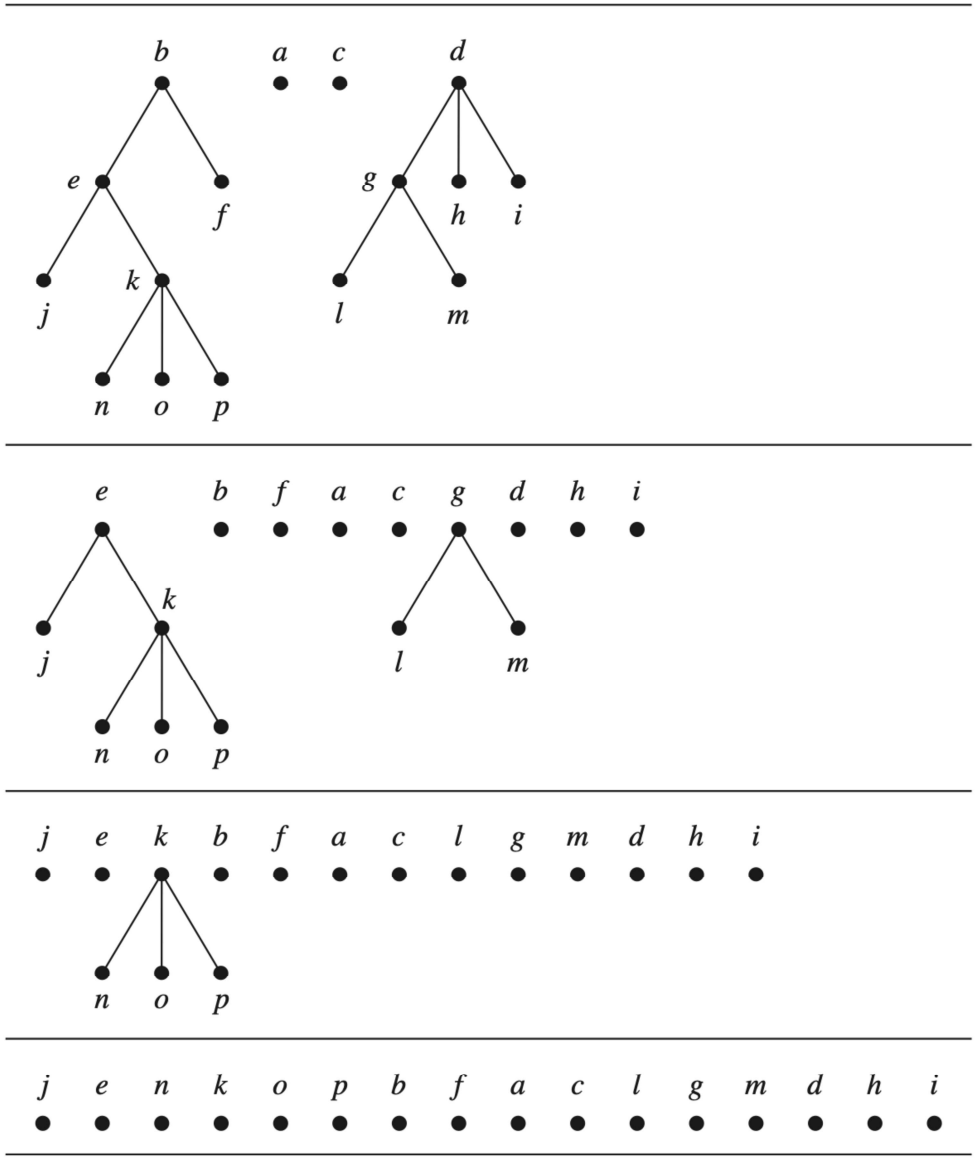




Tree – Traversal algorithm

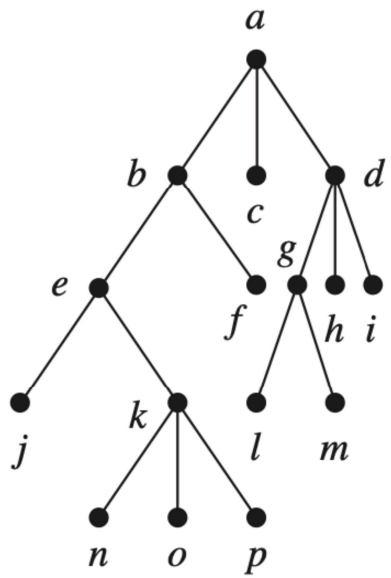
In-order: left – root – right

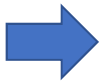


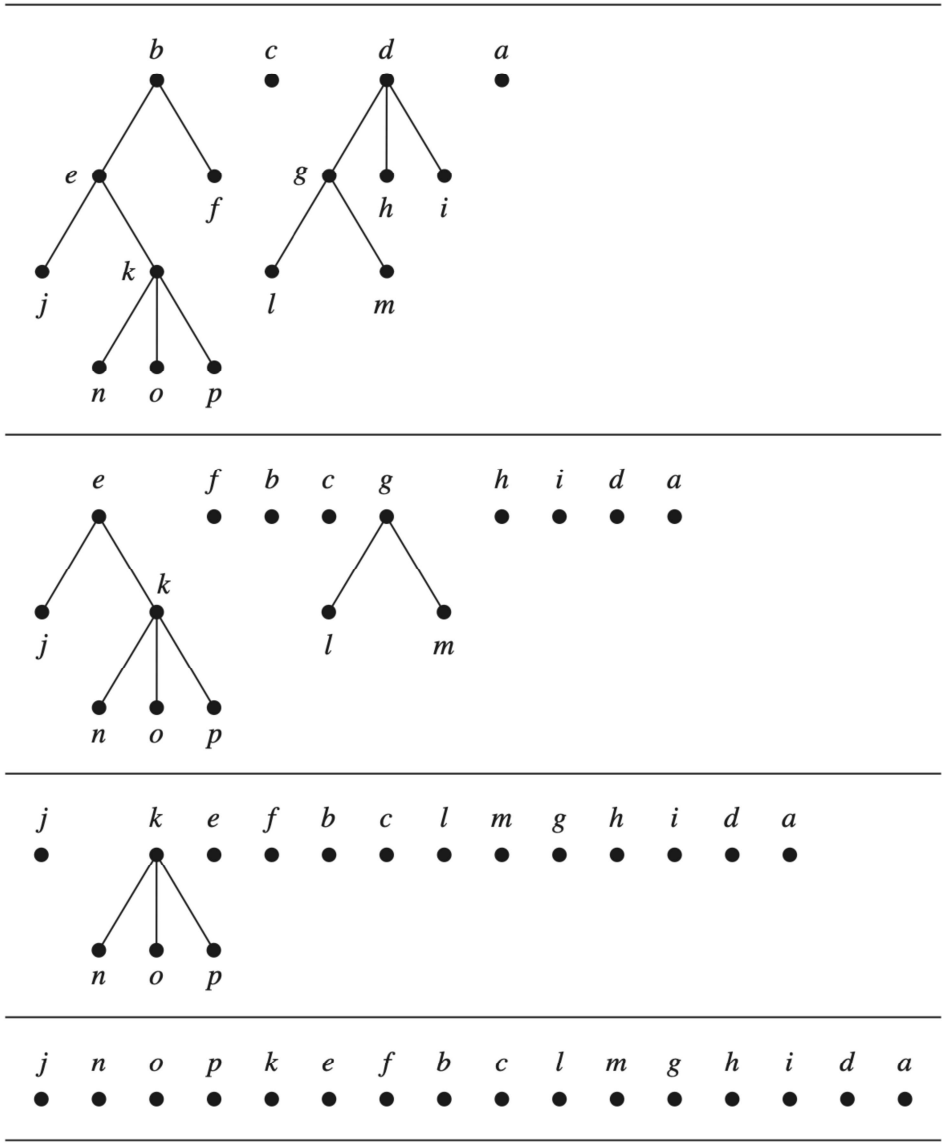


Tree – Traversal algorithm

Post-order: left –right - root







Tree – Represent the expression

Represent the expression using **binary tree, prefix notation, postfix notation, infix notation**

$$((x + y) \uparrow 2) + ((x - 4)/3)$$

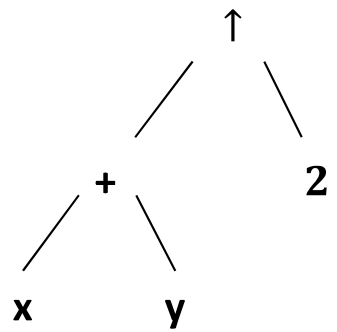
Tree – Represent the expression

Binary tree

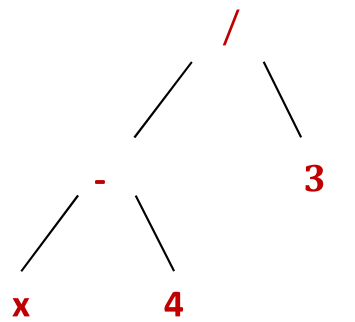
$$((x + y) \uparrow 2) + ((x - 4)/3)$$

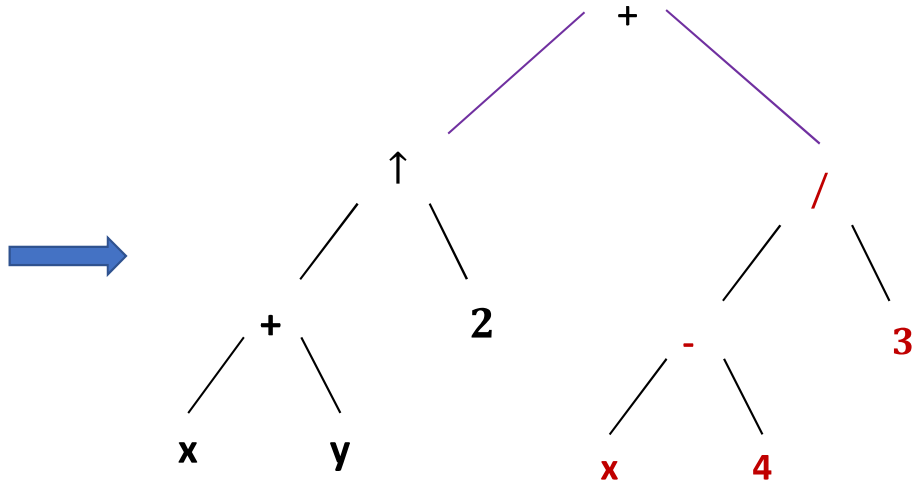
(1) + (2)

(1) (x + y) ↑ 2



(2) (x – 4)/3

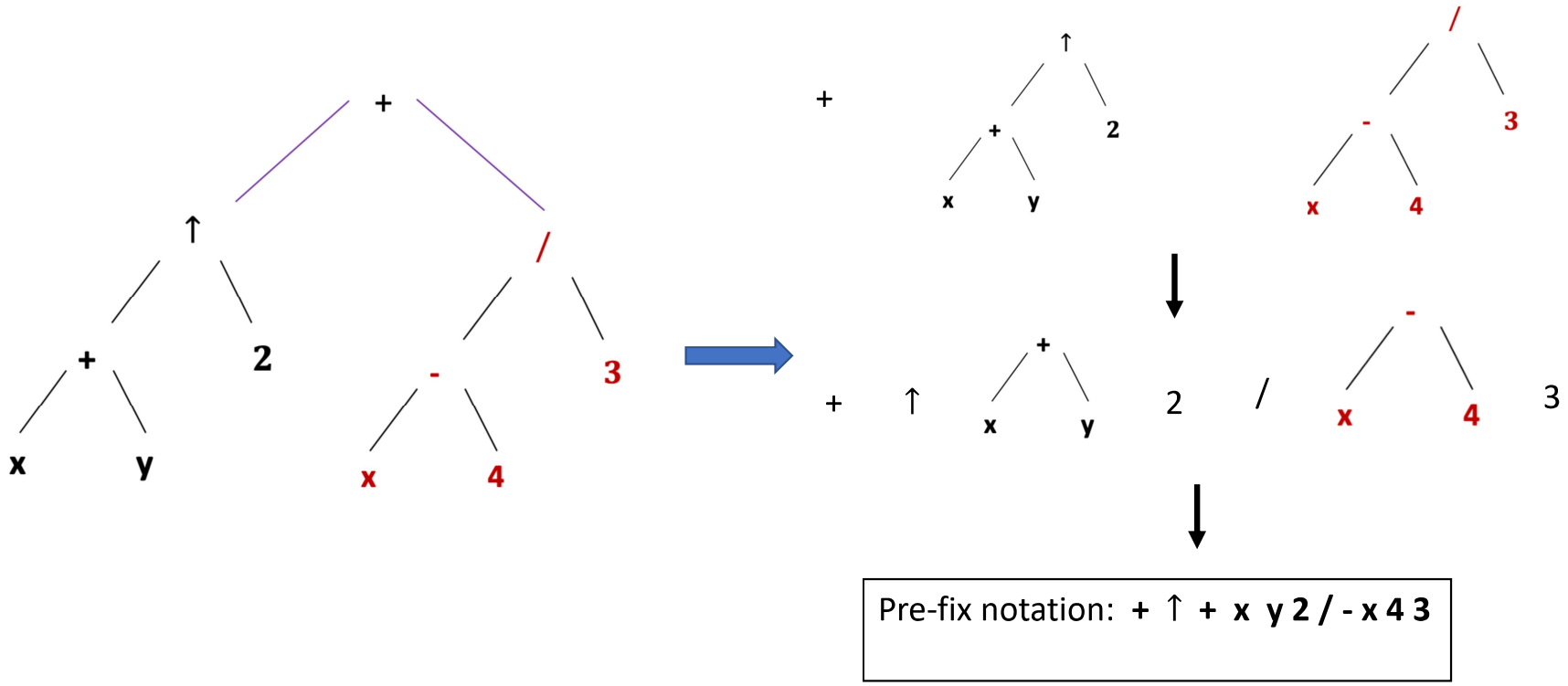




Tree – Represent the expression

prefix notation: root – left –right

$$((x + y) \uparrow 2) + ((x - 4)/3)$$

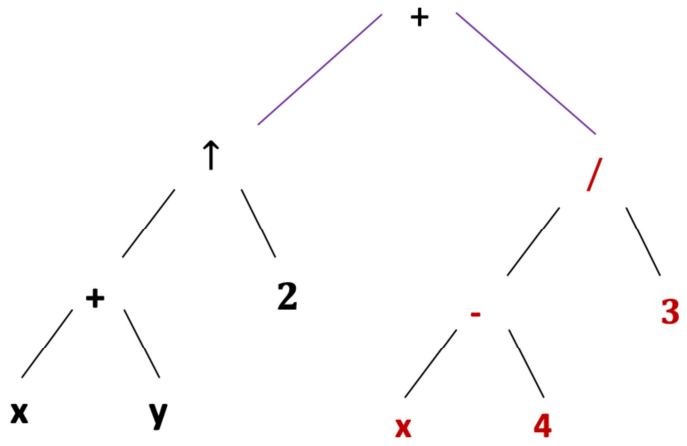


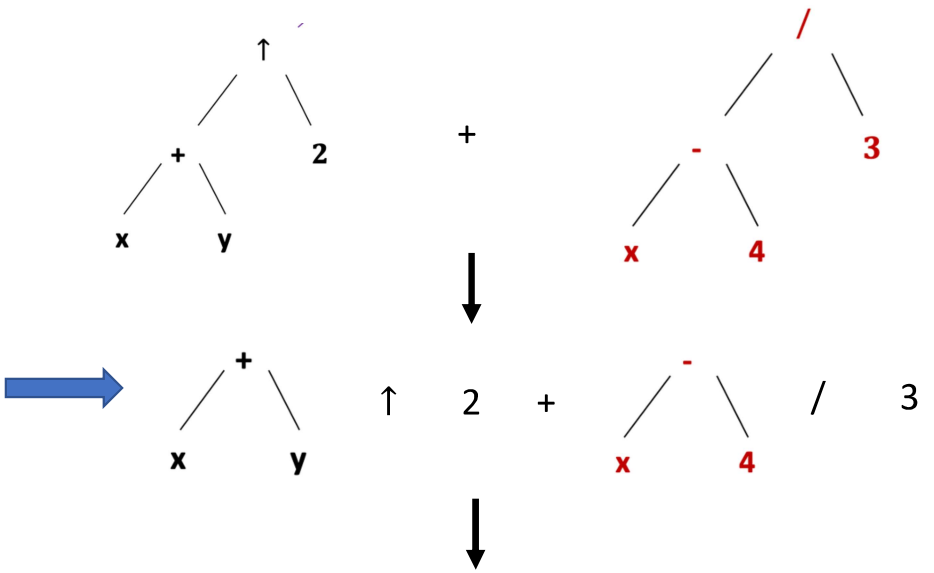
Pre-fix notation: **+ ↑ + x y 2 / - x 4 3**

Tree – Represent the expression

infix notation: left – root -right

$$((x + y) \uparrow 2) + ((x - 4)/3)$$



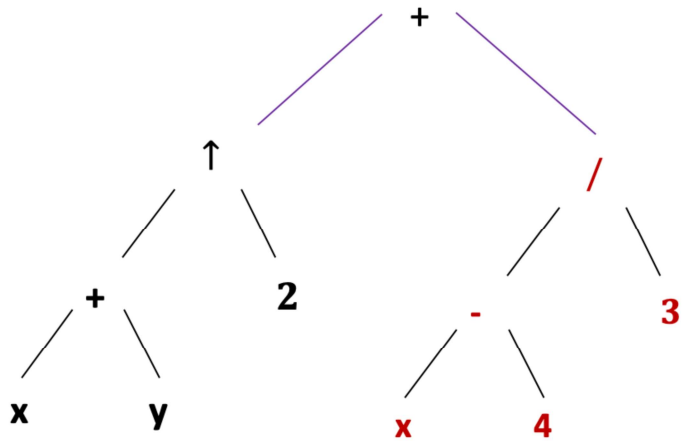


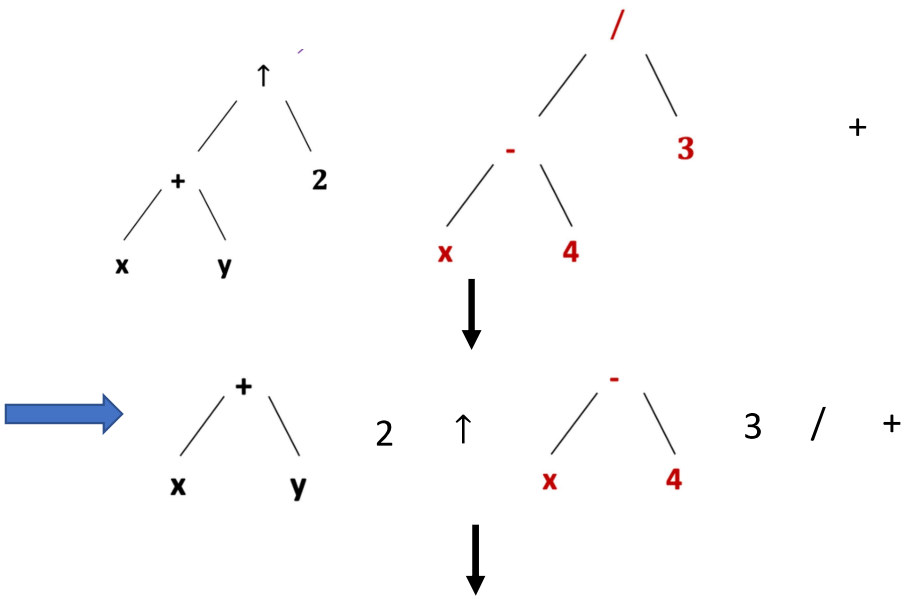
Infix notation: **x + y ↑ 2 + x – 4 / 3**

Tree – Represent the expression

postfix notation: left – right - root

$$((x + y) \uparrow 2) + ((x - 4)/3)$$



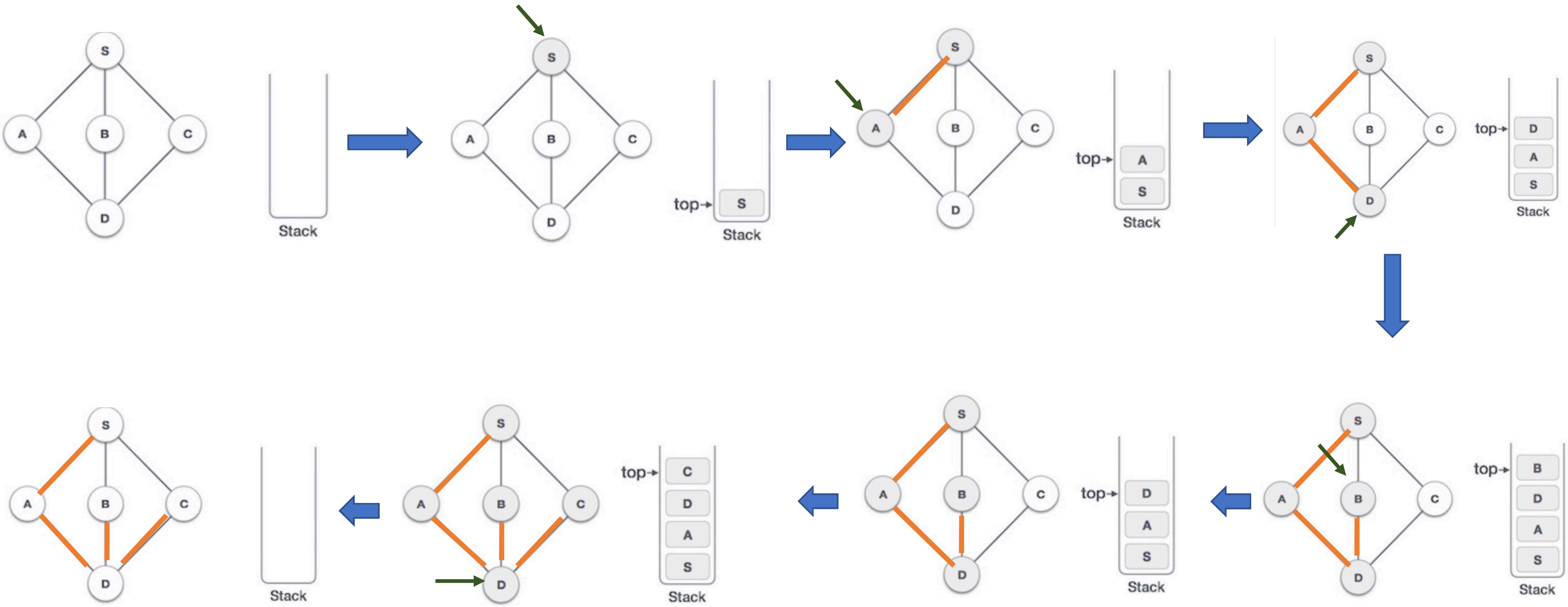


Postfix notation: **x y + 2 ↑ x 4 - 3 / +**

Tree – DFS and BFS

Question: Produce the spanning tree for the given graph using DFS / BFS

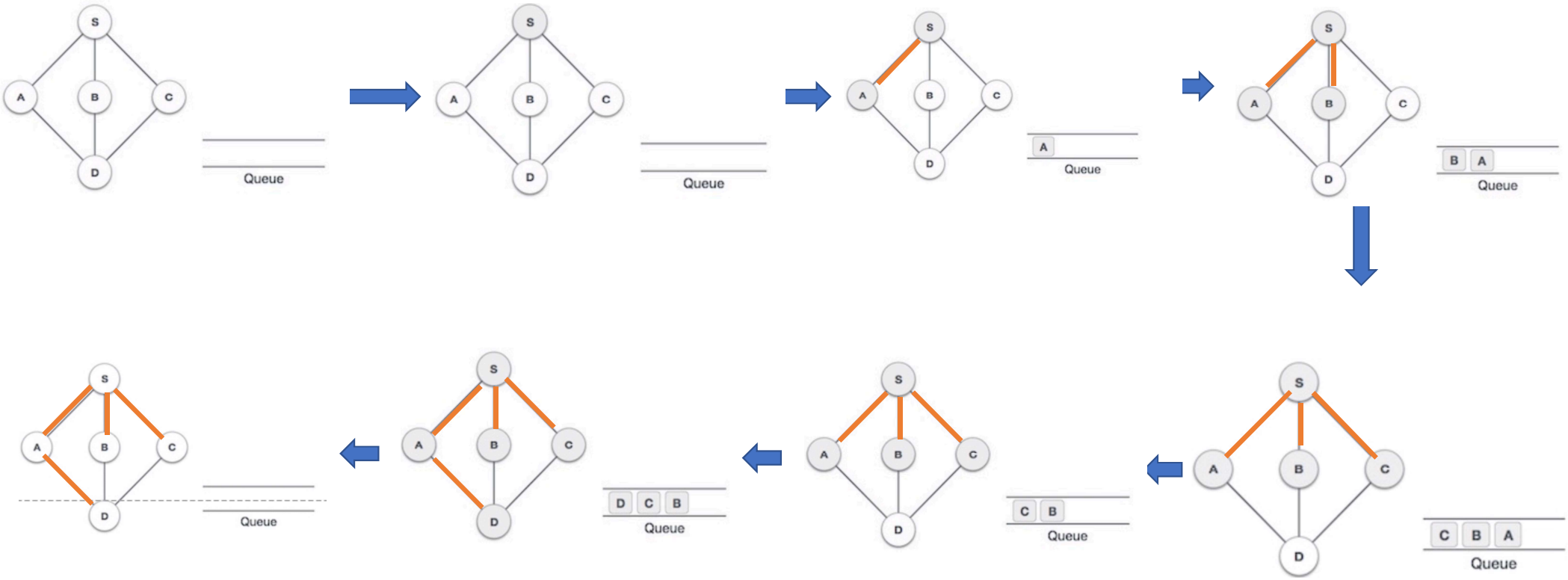
Deep-first Search



Tree – DFS and BFS

Question: Produce the spanning tree for the given graph using DFS / BFS

Breadth -first Search

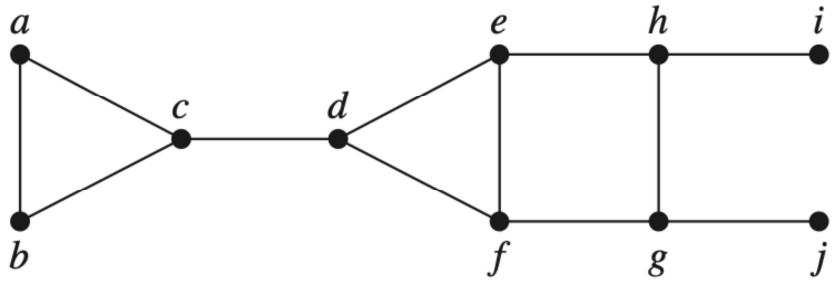


Tree – DFS and BFS

Use the DFS and BFS to produce the spanning tree for the given graph. Choose “a” as the root of the spanning tree and assume that the vertices are ordered alphabetically

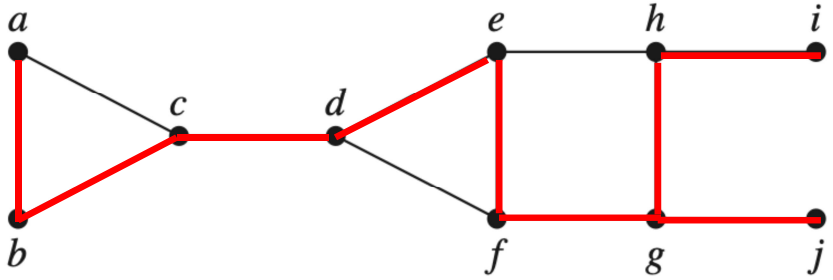
Note:

- If a rooted note is not defined, you can choose any vertice to become a rooted note.
- “Assume that the vertices are ordered alphabetically” -> choose the adjacent vertices based on alphabetical order



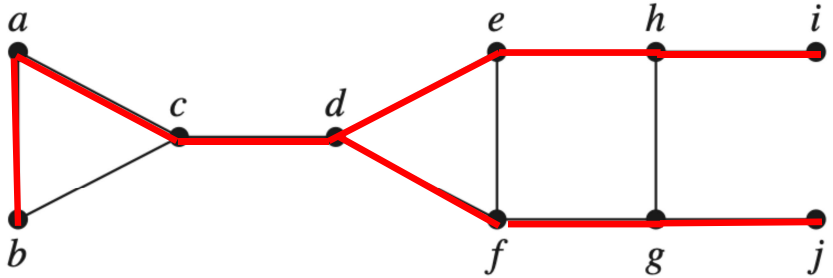
Tree – DFS and BFS

Deep-first Search



i
h j
g
f
e
d
c
b
a

Breadth -first Search



j
i
g
h
f
e
d
c
b
a