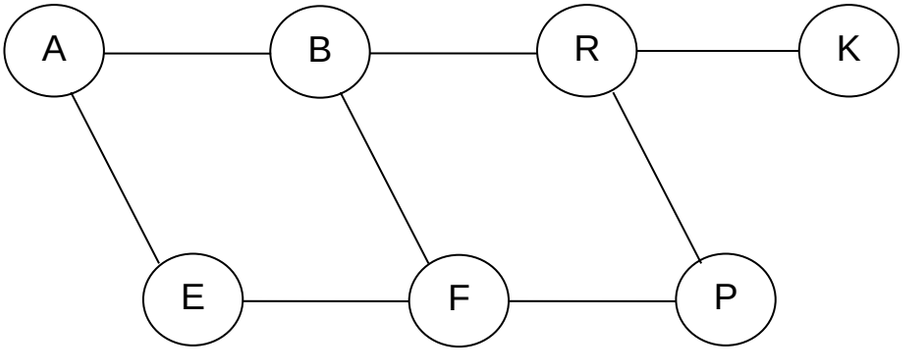
Technical Assignment 2

Part II. Data Structure and Algorithms

In the following graph, nodes are represented as alphabets.



Depth First Search (DFS)

A stack is used. PUSH and POP operations of a stack are used.

PATH 1:

Step 1:

Visited :

Result : -

K	R	B	A	E	F	P
0	0	0	0	0	0	0

-
-
-
-
-

Stack

Step 2:

Visited :

Result : K

K	R	B	A	E	F	P
1	0	0	0	0	0	0

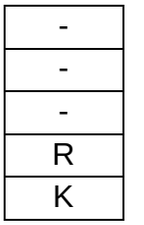
-
-
-
-
K

Stack

Step 3:

Visited : Result : K R

K	R	B	A	E	F	P
1	1	0	0	0	0	0

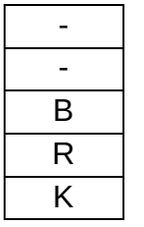


Stack

Step 4: P and B are the adjacent nodes of R. For this PATH, we consider B.

Visited : Result : K R B

K	R	B	A	E	F	P
1	1	1	0	0	0	0

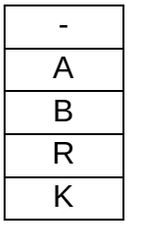


Stack

Step 5: F and A are the adjacent nodes of B. For this PATH, we consider A.

Visited : Result : K R B A

K	R	B	A	E	F	P
1	1	1	1	0	0	0

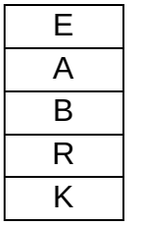


Stack

Step 6: B and E are the adjacent nodes of A. For this PATH, we consider E as B has already been visited.

Visited : Result : K R B A E

K	R	B	A	E	F	P
1	1	1	0	1	0	0



Stack

Result of PATH 1 - K R B A E

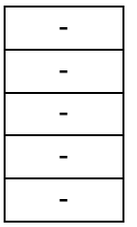
PATH 2:

Step 1:

Visited :

Result : -

K	R	B	A	E	F	P
0	0	0	0	0	0	0



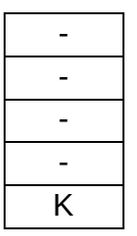
Stack

Step 2:

Visited :

Result : K

K	R	B	A	E	F	P
1	0	0	0	0	0	0



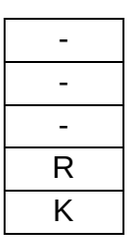
Stack

Step 3:

Visited :

Result : K R

K	R	B	A	E	F	P
1	1	0	0	0	0	0



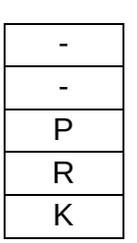
Stack

Step 4: P and B are the adjacent nodes of R. For this PATH, we consider P as we have already considered B in PATH 1.

Visited :

Result : K R P

K	R	B	A	E	F	P
1	1	0	0	0	0	1



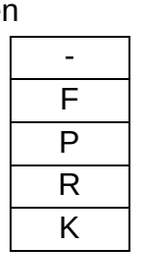
Stack

Step 5: R and F are the adjacent nodes of P. For this PATH, we consider F as R has already been visited.

Visited :

Result : K R P F

K	R	B	A	E	F	P
1	1	0	0	0	1	1



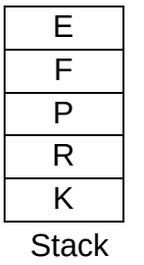
Stack

Step 6: B and E are the adjacent nodes of F. For this PATH, we consider E as it is the final destination.

Visited :

Result : K R P F E

K	R	B	A	E	F	P
1	1	0	0	1	1	1



Stack

Result of PATH 2 - K R P F E

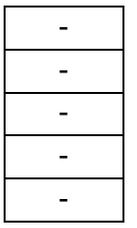
PATH 3:

Step 1:

Visited :

Result : -

K	R	B	A	E	F	P
0	0	0	0	0	0	0



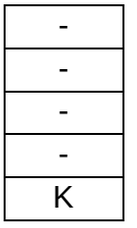
Stack

Step 2:

Visited :

Result : K

K	R	B	A	E	F	P
1	0	0	0	0	0	0



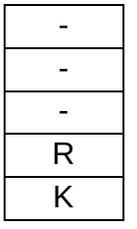
Stack

Step 3:

Visited :

Result : K R

K	R	B	A	E	F	P
1	1	0	0	0	0	0



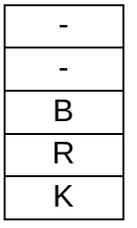
Stack

Step 4: P and B are the adjacent nodes of R. For this PATH, we consider B.

Visited :

Result : K R B

K	R	B	A	E	F	P
1	1	1	0	0	0	0



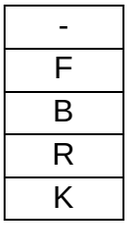
Stack

Step 5: F and A are the adjacent nodes of B. For this PATH, we consider F as we have already used A in PATH 1.

Visited :

Result : K R B F

K	R	B	A	E	F	P
1	1	1	0	0	1	0



Stack

Step 6: B and E are the adjacent nodes of A. For this PATH, we consider E as B has already been visited.

Visited :

Result : K R B F E

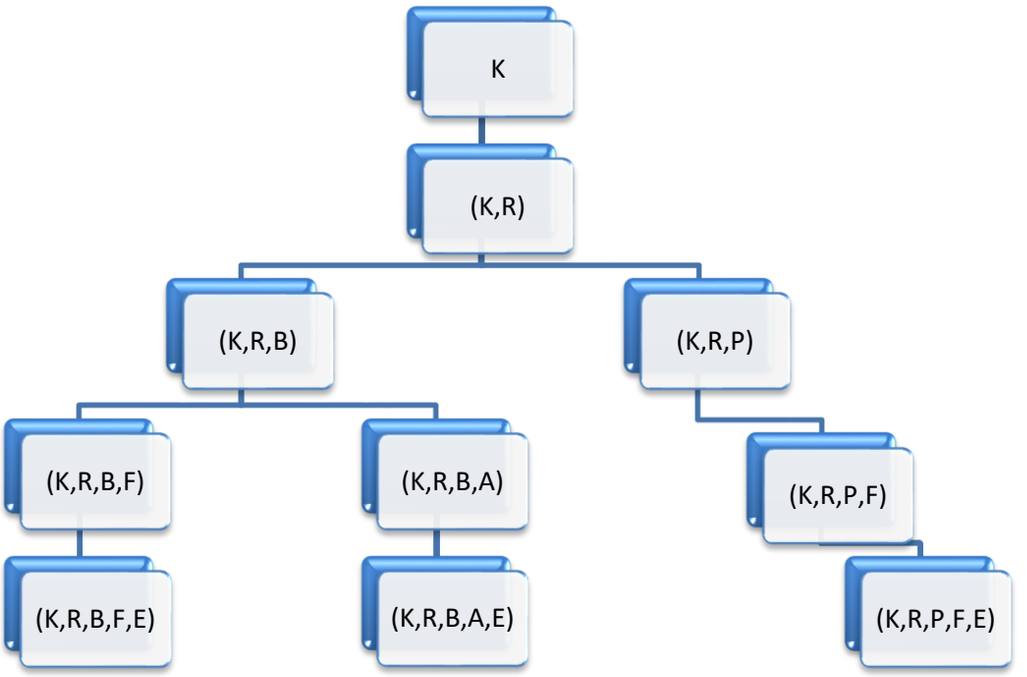
K	R	B	A	E	F	P
1	1	1	0	1	1	0

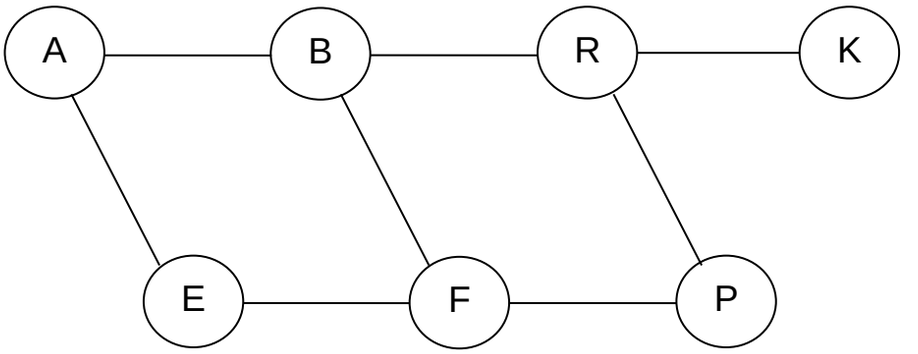
E
F
B
R
K

Stack

Result of PATH 3 - K R B F E

DFS Tree





Breadth First Search (BFS)

A Queue is used. ENQUEUE and DEQUEUE are the operations used.

Consider the **RED** colored element in the QUEUE as DEQUEUE element and the one in **GREEN** to be the current node.

Step 1 :

Result : K

Queue :

K

Step 2 :

Result : K R

Queue :

K
R

Step 3 :

Result : K R B

Queue :

K
R
B

Step 4 :

Result : K R B P

Queue :

K
R
B
P

Step 5 :

Result : K R B P A

Queue :

K
R
B
P
A

Step 6 :

Result : K R B P A F

Queue :

K
R
B
P
A
F

Step 7 :

Result : K R B P A F

Queue :

K
R
B
P
A
F
E

Step 8 :

Result : K R B P A F

Queue :

K
R
B
P
A
F

Step 9 :

Result : K R B P A F E

Queue :

K
R
B
P
A
F

Step 10 :

K
R
B
P
A

Result : K R B P A F E

Queue : F

Final BFS - K R B P A F E

BFS Tree