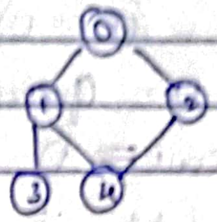
# BFS traversal

$v = 5;$ ← [nodes]

[Graph] $$M = \left[\begin{array}{c|ccccc} & 0 & 1 & 2 & 3 & 4 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 2 & 1 & 0 & 0 & 0 & 1 \\ 3 & 0 & 1 & 0 & 0 & 0 \\ 4 & 0 & 1 & 1 & 0 & 0 \end{array}\right]$$



source = 0 [Assumption]

bfs (Graph, 5, 0)

**function call:-**

```
int queue;
front = 0 , rear = 0 , u , i ; visited

for (i=0 ; i<5 ; i++)
  { visited [0] = 0 ;
for (i=1 ; i<5 ; i++)
    visited [1] = 0 ;
for (i=2 ; i<5 ; i++)
    visited [2] = 0 ;
for (i=3 ; i<5 ; i++)
    visited [3] = 0 ;
for (i=4 ; i<5 ; i++)
    visited [4] = 0 ;
for (i=5 ; i<5 ; i++) X
```

$$\therefore \begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \\ \hline 0 & 0 & 0 & 0 & 0 \end{array}$$

```
queue[rear[0]] = 0
    visited[0] = 1                          0 1 2 3 4
                                            1 0 0 0 0
cout - - -                              rear front


while (0 <= 0) ✓
    u = queue[f]->0;
    cout == 0                     -> [0]
    f++ = 1
~~while (1 <= 0)~~ X
    for (i=0; i<5; i++)
        if (M[0][0] == 1 && visited[0] == 0) X
    for (i=1; i<5; i++)
        if (M[0][1] == 1 && visited[1] == 0) ✓
                                            0 1 2 3 4
            visited[1] = 1;                 1 1 0 0 0
            rear++ = 1                      ~~0 1 0 0 0~~
            queue[rear] = 1;                fro   rear
    for (i=2; i<5; i++)
        if (M[0][2] == 1 && visited[2] == 0) ✓
            visited[2] = 1;                 0 1 2 3 4
            rear++ = 2                      1 1 1 0 0 0
            queue[rear] = 2;                front rear
    for (i=3; i<5; i++)
        if (M[0][3] == 1 && visited[3] == 0) X
    for (i=4; i<5; i++)
        if (M[0][4] == 1 && visited[4] == 0) X
    for (i=5; i<5; i++) X
    }
while (1 <= 2) ✓
    u = queue[front]->1
    cout = 1                      -> [0 1]
    front++ -> 2
```

```
for(i = 0; i < 5; i++)
    if (M[i][0] == 1 && visited[0] == 0) X
        ~~visited~~
for (i = 1; i < 5; i++)
    if(M[i][1] == 1 && visited[i] == 0) X
for (i = 2; i < 5; i++)
    if (M[i][2] == 1 && visited [2] == 0) X
for (i = 3; i < 5; i++)
    if (M[i][3] == 1 && visited[3] == 0) ✓
        visited[3] = 1;                      0 1 2 3 4
        rear++ = 3                           1 1 1 1 0
        queue[rear] = 3;                         front  rear
for (i = 4; i < 5; i++)
    if (M[i][4] == 1 && visited[4] == 0) ✓
        visited [4] = 1;                     0 1 2 3 4
        rear++ = 4                           1 1 1 1 1
        queue [rear] = 4;                        front  rear
for (i = 5; i < 5; i++) X

while (2 <= 4) ✓
  v = queue[front] = 2
  cout = 2        →   | 0 1 2 |
  front ++ ⇒ 3
   for (i = 0; i < 5; i++)
       if (M [2] [0] == 1 && visited (0) == 0) X
   for ( i = 1; i < 5; i++)
       if (M[2][0] == 1 && visited(0) == 0) X
   for ( i = 2; i < 5; i++)
       if (M[2][0] == 1 && visited (0) == 0) X
   fo

   for (i = 5; i < 5; i++) X               0 1 2 3 4
                                           1 1 1 1 1
                                                 front rear
```

while (3 <= 4) ✓

u = queue(front) → 3

Cout . 3 → $\boxed{0\ 1\ 2\ 3}$

front ++ → 4

for (i=0 ; i < 5; i++) .

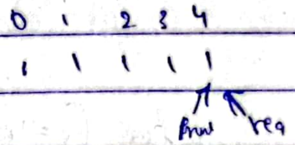
    if (M[3][0] == 1 && visited[0]==0) X

for (i=1 ; i<5; i++)

    :

    :

for (i=5; i<5; i++) X



while (4 <= 4) ✓

u = queue [front] → 4

cout = 4 → $\boxed{0\ 1\ 2\ 3\ 4}$ //

front ++ → 5

for (i=0 ; i<5 ; i++)

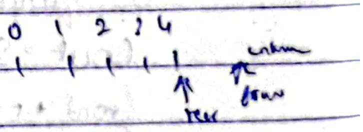
    if (M[4][0] == 1 && visited[0] = = 0) X

for (i=1 ; i<5 ; i++)

    :

    :

for (i=5 ; i<=5 ; i++) X

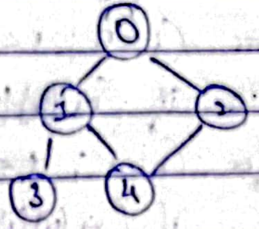


while (5 <= 4) X

∴ BFS = 0 1 2 3 4 //

## DFS traversal:

$$M = \begin{array}{c|ccccc} & 0 & 1 & 2 & 3 & 4 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 2 & 1 & 0 & 0 & 0 & 1 \\ 3 & 0 & 1 & 0 & 0 & 0 \\ 4 & 0 & 1 & 1 & 0 & 0 \end{array}$$



```
v = 5
int source
for (i=0; i<5; i++)
        visited[0] = 0;
  for (i=1; i<5; i++)
        visited[1] = 0;
  for (i=2; i<5; i++)
        visited[2] = 0;
    for (i=3; i<5; i++)
        visited[3] = 0;
    for (i=4; i<5; i++)
        visited[4] = 0;
    for (i=5; i<5; i++) X        =>  0 1 2 3 4
                                     0 0 0 0 0

source = 0
print  ------------------>  [0]
dfs (M, v, source)
dfs (M, 5, 0)
```

```
void dfs (n, v, source){
    visited [0] = 1;  ⟶        0 1 2 3 4
                                1 0 0 0 0
    for (i=0; i<5; i++)
        if (n[0][0] == 1 && visited[0]==0) ✗
    for (i=1; i<5; i++)
        if (n[0][1] == 1 && visited[1]==0) ✓
            cout = 1            ⟶    | 0 1 |
            dfs (n, 5, 1)       ⟶    0 1 2 3 4
                                      1 0 0 0 0
                                        ↑
                                      source

void dfs (n, 5, 1)
    visited [1] = 1;   ⟶    0 1 2 3 4
                             1 1 0 0 0
    for (i=0; i<5; i++)
        if (n[1][0] == 1 && visited[0]==0) ✗
    for (i=1; i<5; i++)
        if (n[1][1] == 1 && visited[1]==0) ✗
    for (i=2; i<5; i++)
        if (n[1][2] == 1 && visited[2]==0) ✗
    for (i=3; i<5; i++)
        if (n[1][3] == 1 && visited[3]==0) ✓
            cout = 3          ⟶    | 0 1 3 |
            dfs (n, 5, 3)     ⟶    0 1 2 3 4
                                    1 1 0 0 0
                                          ↑
                                        source

void dfs (n, 5, 3)
    visited [3] = 1    ⟶    0 1 2 3 4
                             1 1 0 1 0
    for (i=0; i<5; i++)
        if (n[3][0] == 1 && visited[0] ==0) ✗
    for (i=1; i<5; i++)
        if (n[3][1] == 1 && visited[1] == 0) ✗
    for (i=2; i<5; i++)
        if (n[3][2] == 1 && visited[2] ==0) ✗
```

```
for (i=3; i<5; i++)
    if (m[3][3] == 1 && visited[3]==0) X
for (i=4; i<5; i++)
    if (m[3][4] ==1 && visited[4]==0) X
for (i=5; i<5; i++) X
```

∴ Backtrack    source = 1

```
void dfs (m, 5, 1)
    visited [1] = 1;    -->    0 1 2 3 4
                               1 1 0 1 0
    for (i=0; i<5; i++)
        if (m[1][0] == 1 && visited[0] == 0) X
    for (i=1; i<5; i++)
        if (m[1][1] == 1 && visited [1] == 0) X
    for (i=2; i<5; i++)
        if (m[1][2] == 1 && visited [2] == 0) X
    for (i=3; i<5; i++)
        if (m[1][3] == 1 && visited [3] == 0) X
    for (i=4; i<5; i++)
        if (m[1][4] == 1 && visited [4] == 0) ✓
            cout = 4    -->    | 0 1 3 4 |
            dfs (m, 5, 4)  -->    0 1 2 3 4
                                  1 1 0 1 0
                                          ↑
                                        source

void dfs (m, 5, 4)
    visited [4] = 1;    -->    0 1 2 3 4
                               1 1 0 1 1
    for (i=0; i<5; i++)
        if (m[4][0] == 1 && visited[0] == 0) X
    for (i=1; i<5; i++)
        if (m[4][1] == 1 && visited[1] == 0) X
    for (i=2; i<5; i++)
        if (m[4][2] == 1 && visited[2] == 0) ✓
            cout : 2    -->    | 0 1 3 4 2 |
            dfs (m, 5, 2)
```

void. dfs (H, 5, 2)

visited [2] = 1 ⟶

| 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 |

∴ All the nodes are visited

∴ All the conditions will be false

∴ Dfs = 0 1 3 4 2 ///