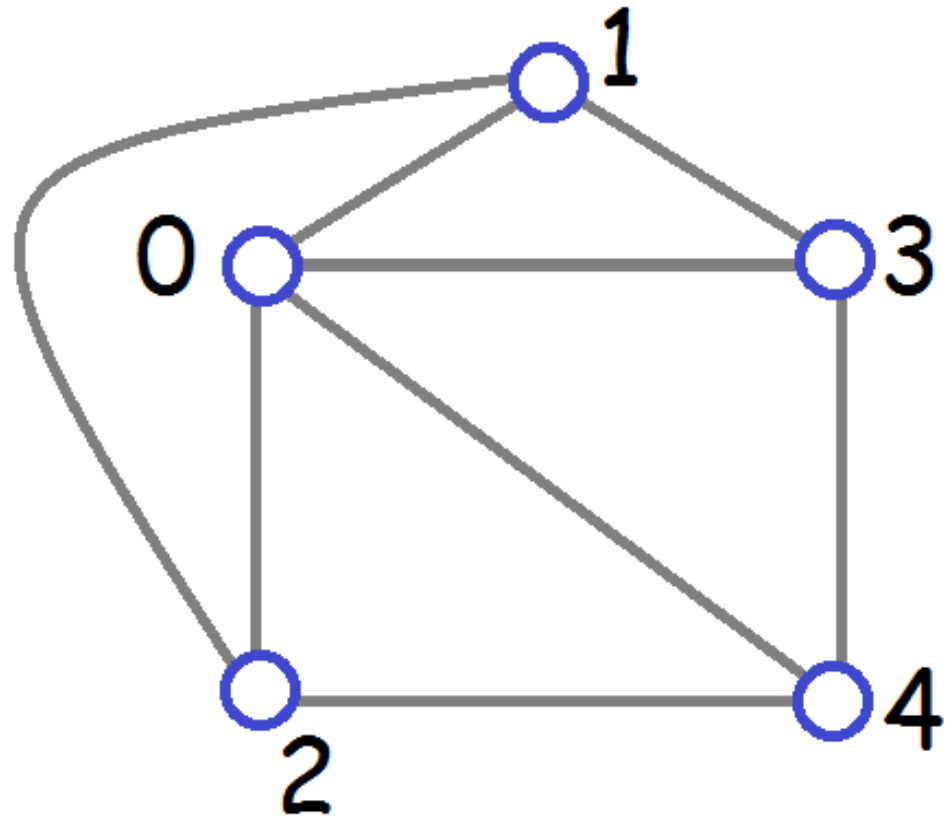


Show what you would type into the computer for the graph pictured using:

1. Upper triangular adjacency matrix format.
2. Adjacency list format (give a rotation system that represents the planar embedding).



Breadth-first search: method for traversing all the vertices/edges of a graph.

I've programmed this more than any other graph algorithm!

Some uses: graph traversal, finding connected components, identifying cut vertices, finding cycles, isomorphism testing for 3-connected planar graphs, finding bridges for a planarity testing algorithm, finding a maximum flow in a network, reordering vertices so algorithms for hard problems (clique, independent set, dominating set) perform better...

Queue (used for BFS)

http://devcentral.f5.com/weblogs/images/devcentral_f5_com/weblogs/Joe/WindowsLiveWriter/PowerShellABCsQisforQueues_919A/queue_2.jpg



<http://www.ac-nancy-metz.fr/enseign/anglais/Henry/bus-queue.jpg>

Queue data structure:

Items are:

Added to the rear of the queue.

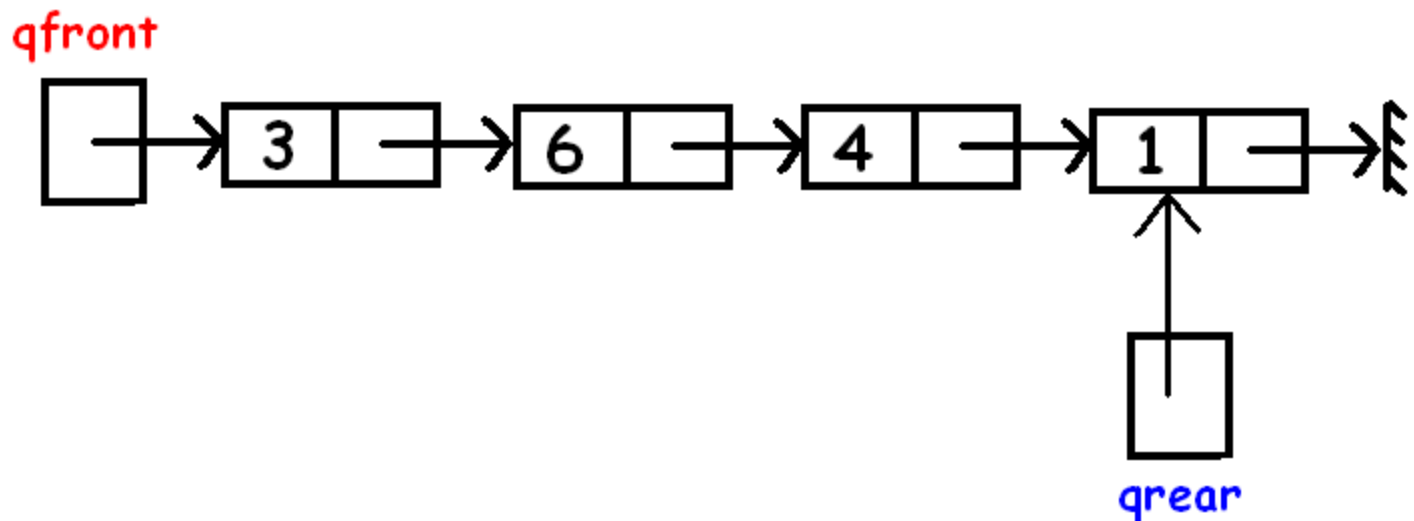
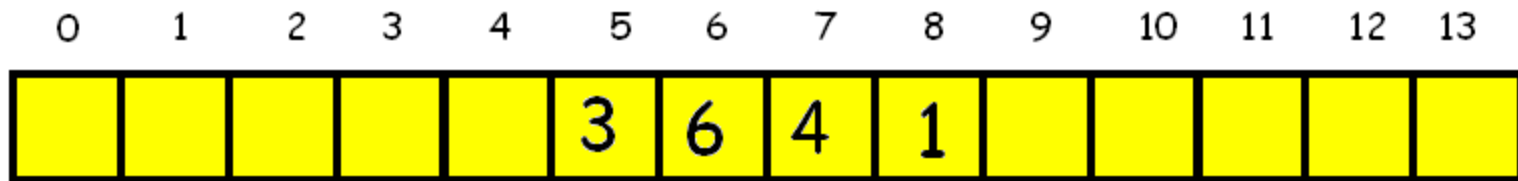
Removed from the front of the queue.



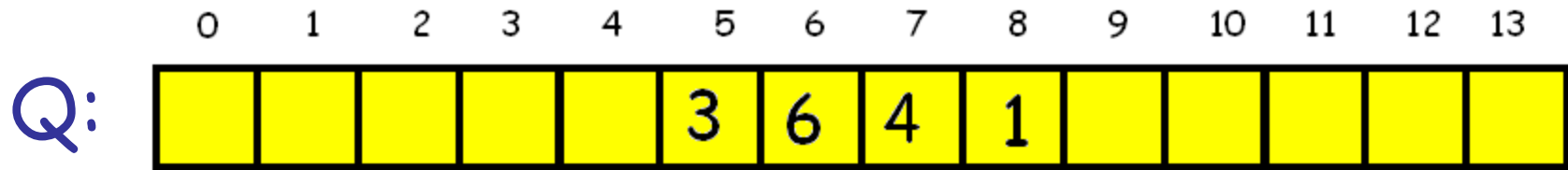
If you have an upper bound on the lifetime size of the queue then you can use an array:

$qfront=5$, $qrear=9$

($qrear$ is next empty spot in array)



$qfront=5, qrear=9$



To test if there is something in the queue:

if ($qfront < qrear$)

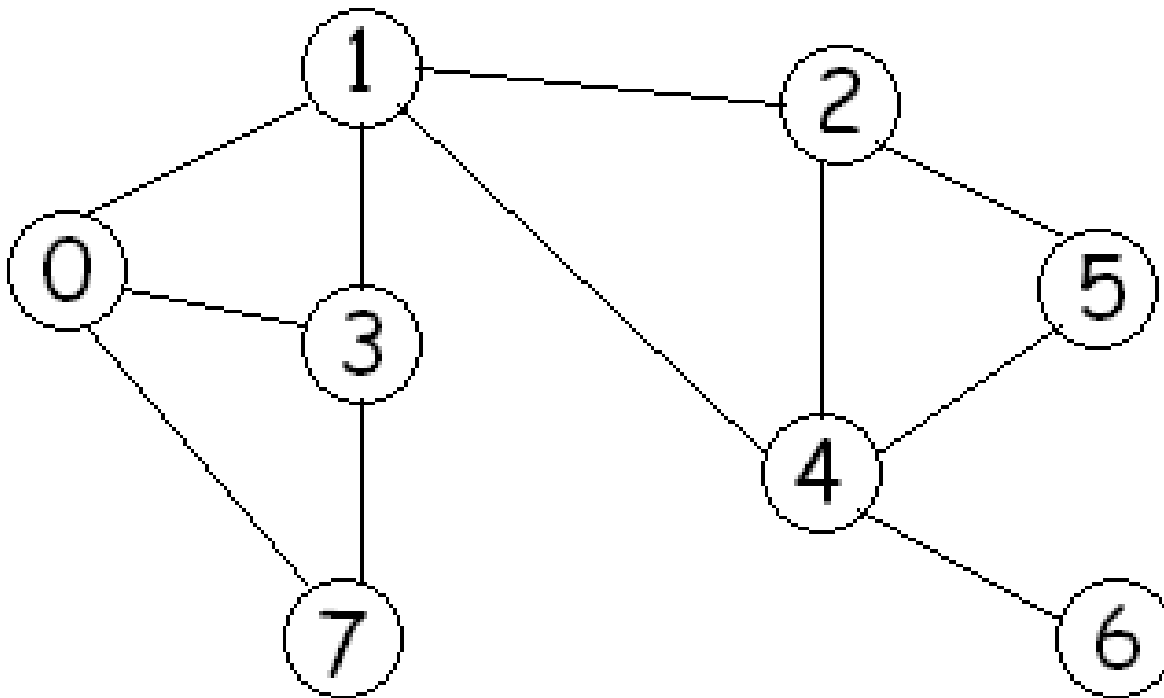
To add x to the queue:

$Q[qrear] = x; qrear++;$

To delete front element of the queue:

$x = Q[qfront]; qfront++;$

If the neighbours of each vertex are ordered according to their vertex numbers, in what order does a BFS starting at 0 visit the vertices?



BFS starting at a vertex s using an array for the queue:

Data structures:

A queue $Q[0..(n-1)]$ of vertices, $qfront$, $qrear$.
 $parent[i]$ = BFS tree parent of node i .

The parent of the root s is s .

To initialize:

```
// Set parent of each node to be -1 to indicate  
// that the vertex has not yet been visited.
```

```
for (i=0; i < n; i++) parent[i]= -1;
```

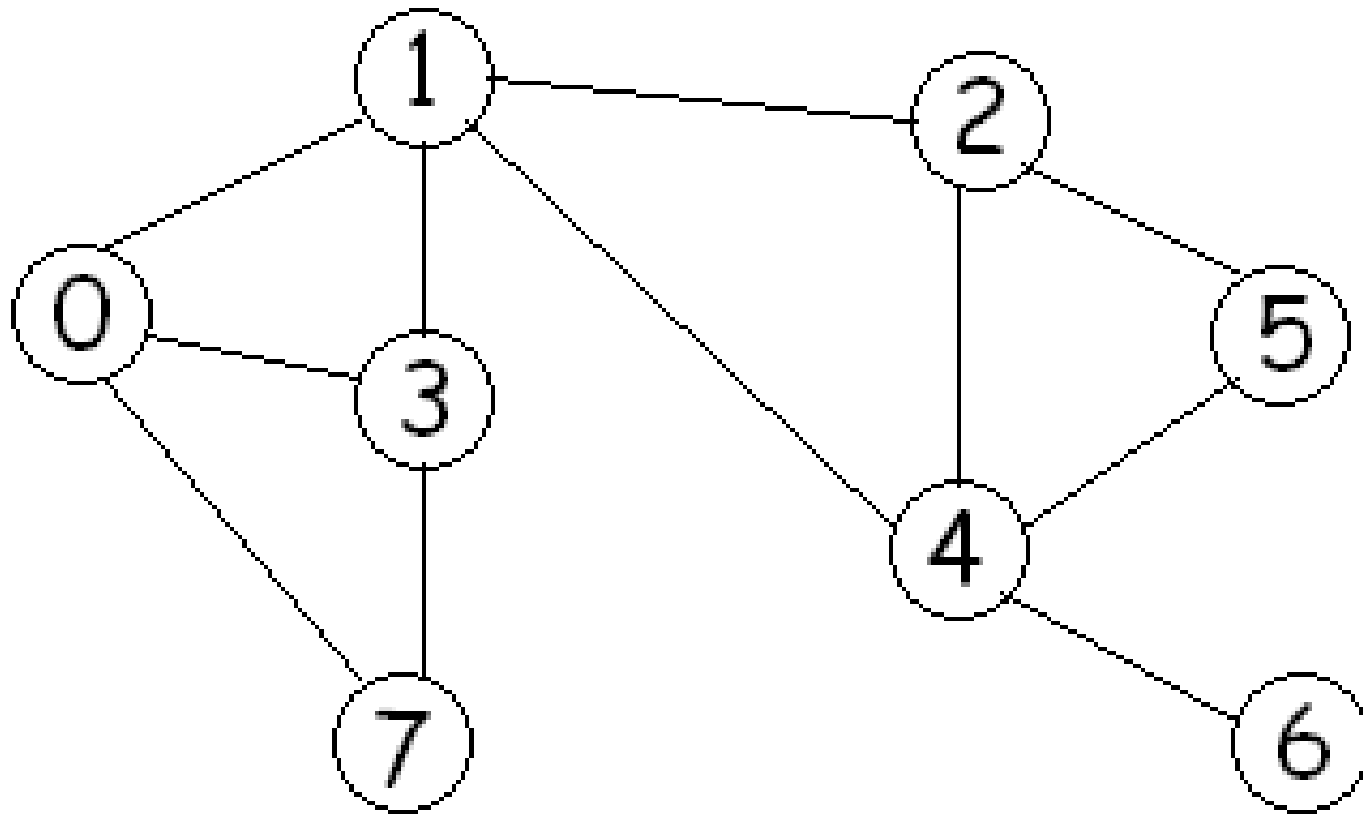
```
// Initialize the queue so that BFS starts at  $s$ 
```

```
qfront=0; qrear=1; Q[qfront]=  $s$ ;
```

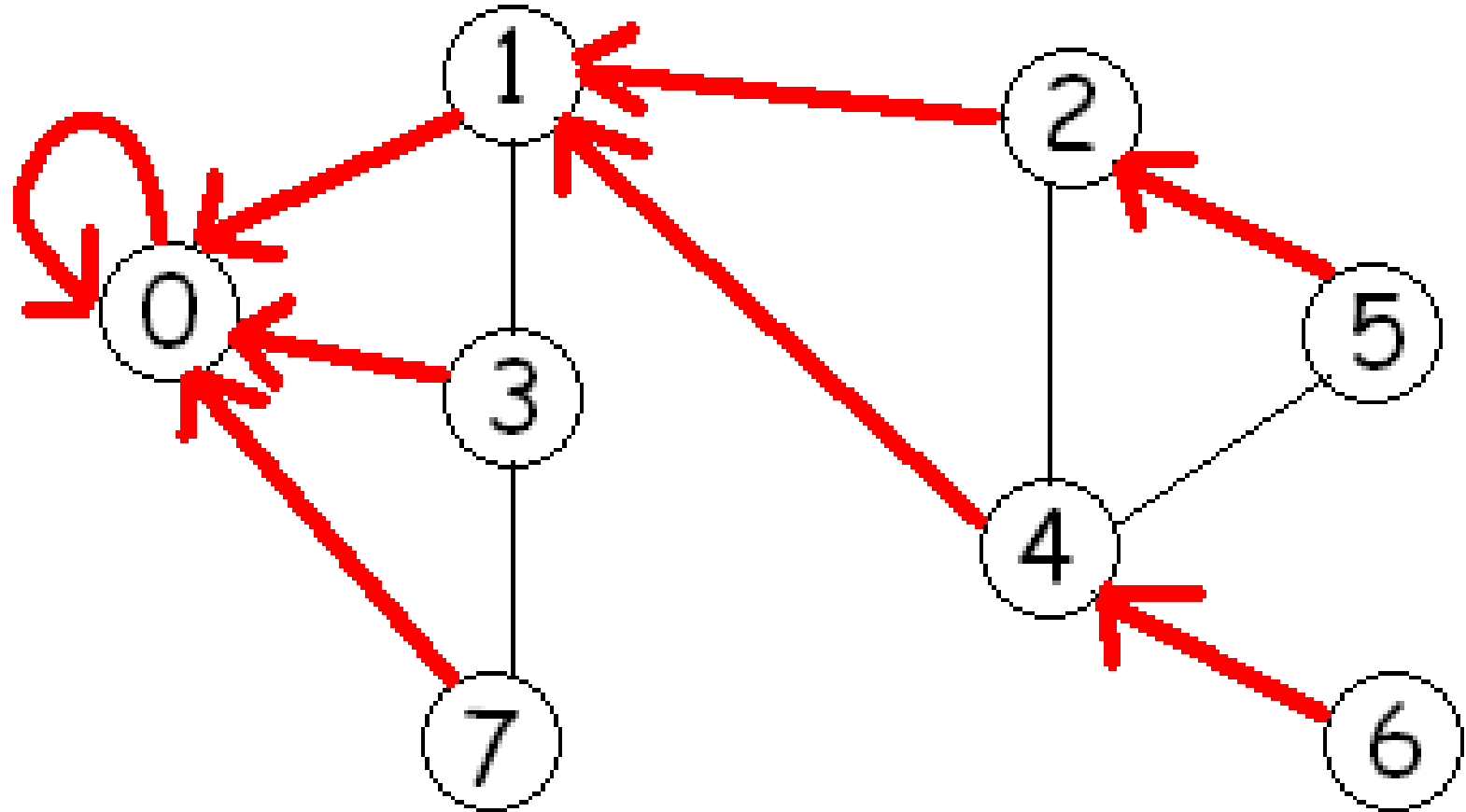
```
parent[ $s$ ]= $s$ ;
```

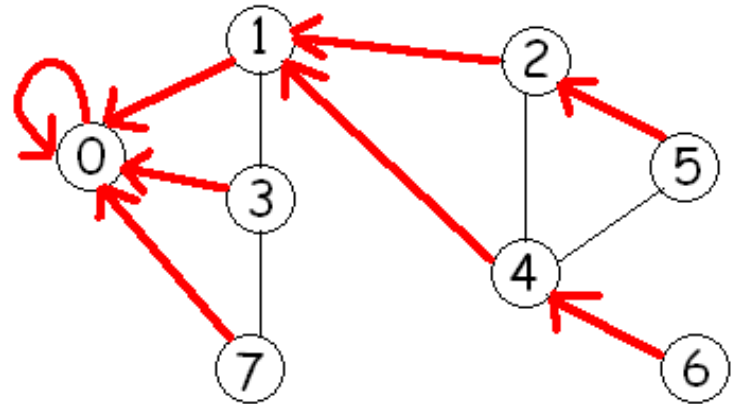


```
while (qfront < qrear) // Q is not empty
    u= Q[qfront]; qfront++;
    for each neighbour v of u
        if (parent[v] == -1) // not visited
            parent[v]= u;
            Q[qrear]= v; qrear++;
        end if
    end for
end while
```

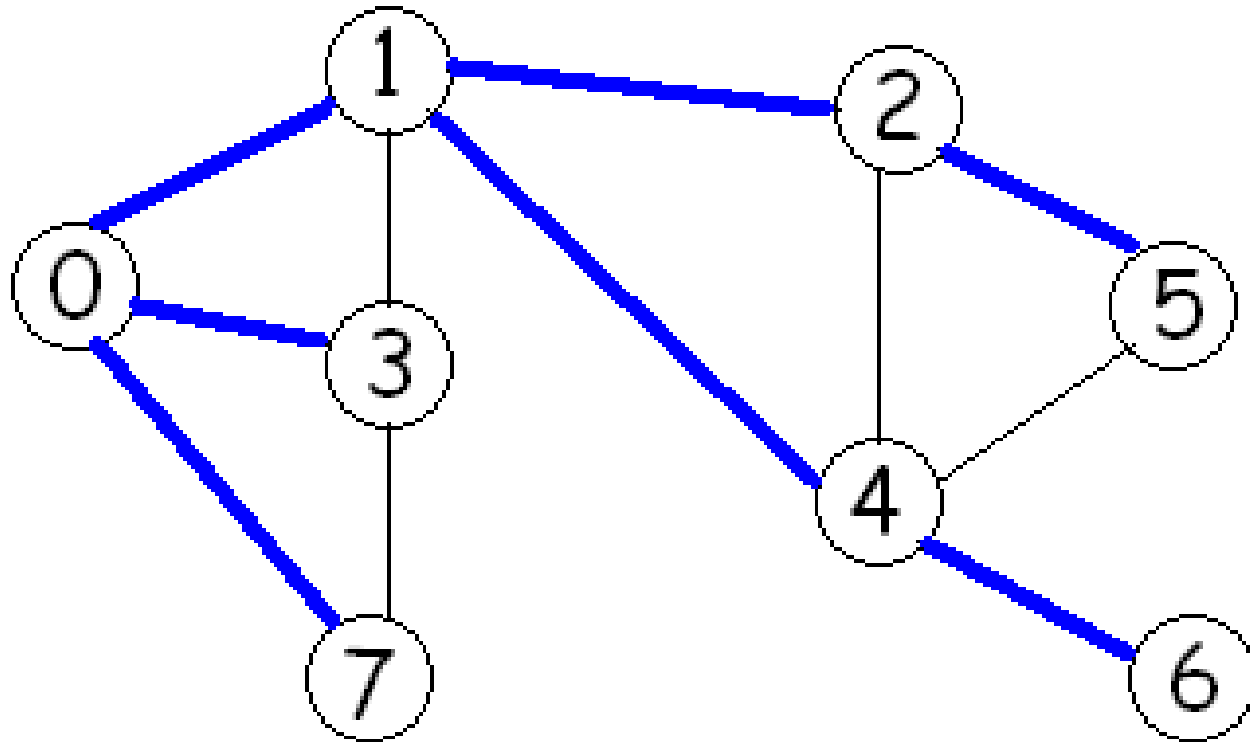


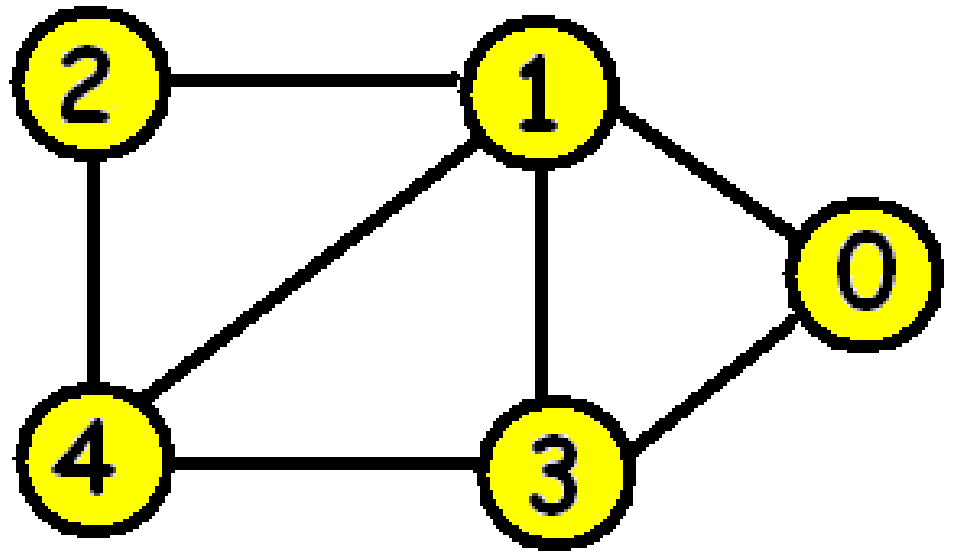
Red arcs represent parent information:





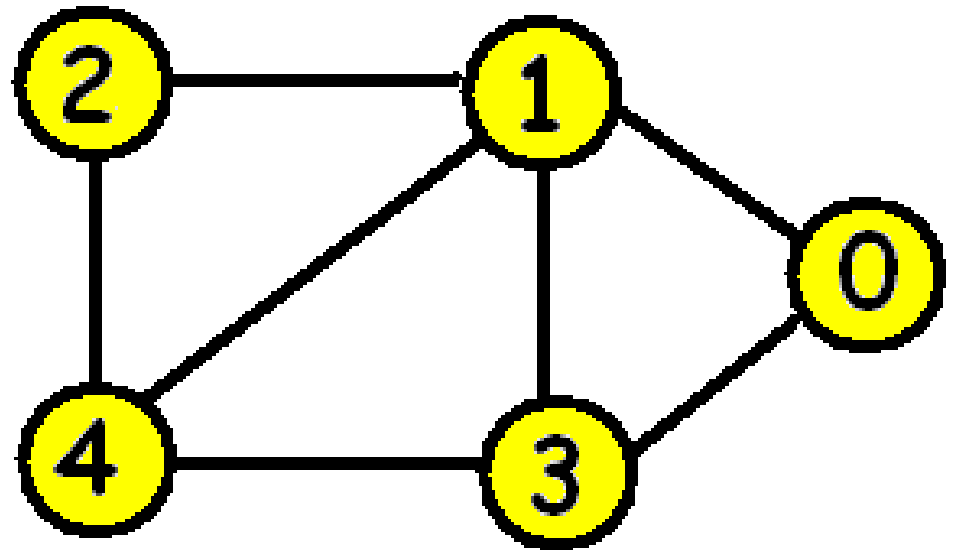
The blue spanning tree is the BFS tree.



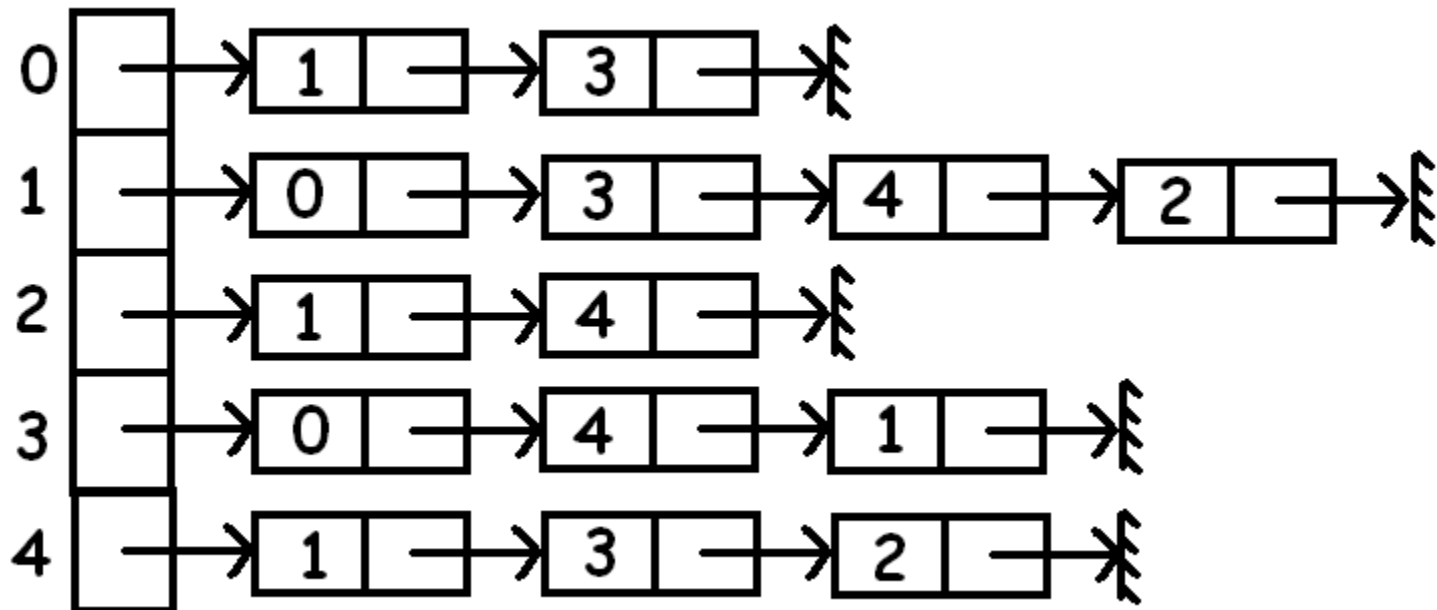


Adjacency matrix:

	0	1	2	3	4
0	0	1	0	1	0
1	1	0	1	1	1
2	0	1	0	0	1
3	1	1	0	0	1
4	0	1	1	1	0

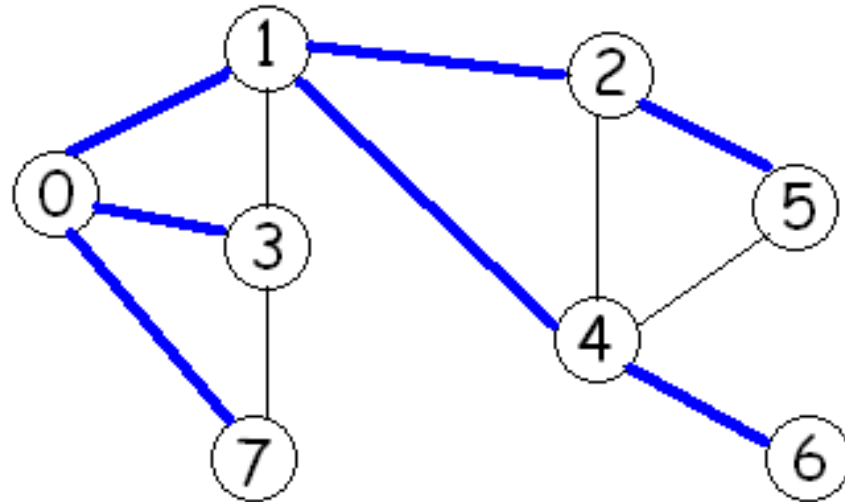


Adjacency list:



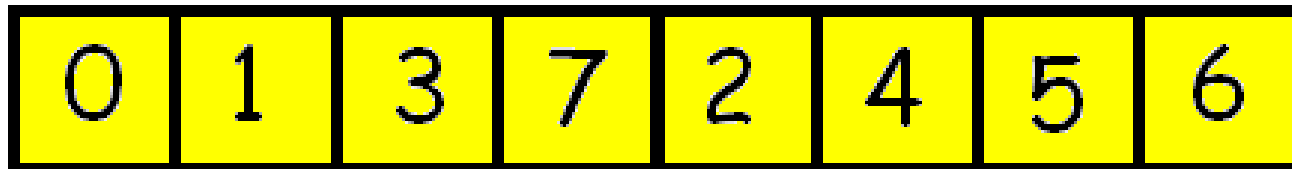
BFI[v]= Breadth first index of v
= step at which v is visited.

The BFI[v] is equal to v's position in the queue.



0 1 2 3 4 5 6 7

Q



To initialize:

```
// Set parent of each node to be -1 to indicate  
// that the vertex has not yet been visited.  
for (i=0; i < n; i++) parent[i]= -1;
```

```
// Initialize the queue so that BFS starts at s  
qfront=0; qrear=1; Q[qfront]= s;  
parent[s]=s;
```

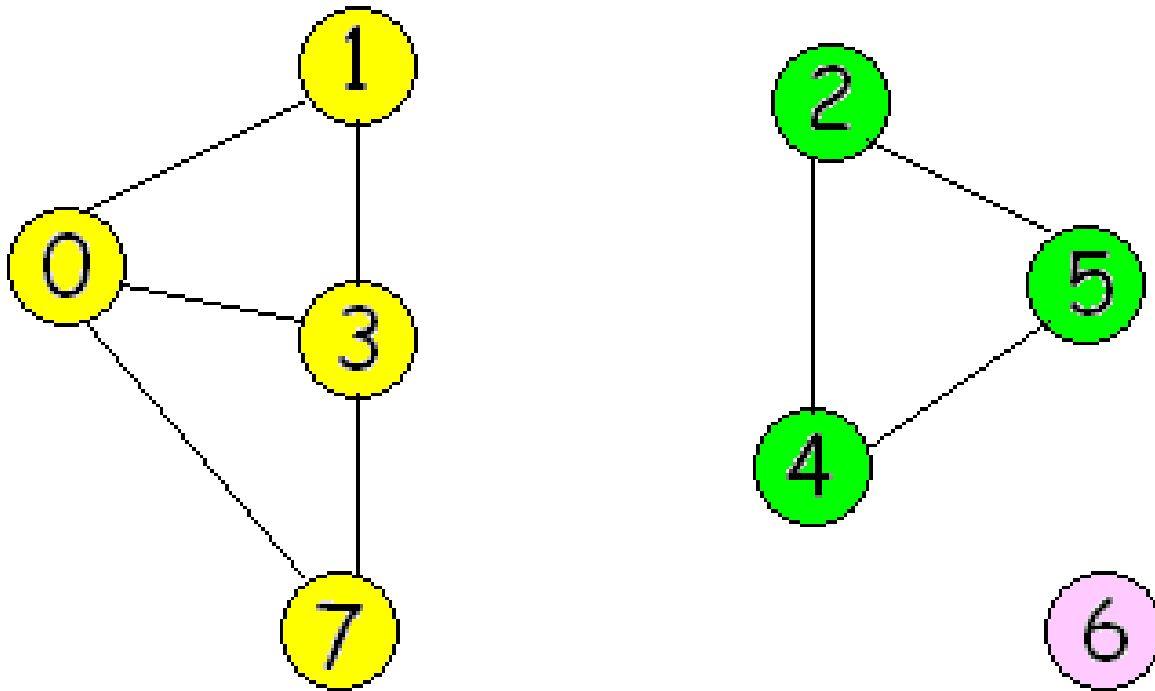
```
BFI[s]= 0;
```



```
while (qfront < qrear) // Q is not empty
    u= Q[qfront]; qfront++;
    for each neighbour v of u
        if (parent[v] == -1) // not visited
            parent[v]= u; BFI[v]= qrear;
            Q[qrear]= v; qrear++;
        end if
    end for
end while
```

One application:

How many connected components does a graph have and which vertices are in each component?



To find the connected components:

```
for (i=0; i < n; i++)
```

```
    parent[i]= -1;
```

```
nComp= 0;
```

```
for (i=0; i < n; i++)
```

```
    if (parent[i] == -1)
```

```
        nComp++;
```

```
        BFS(i, parent, component, nComp);
```

```
BFS(s, parent, component, nComp)
```

```
// Do not initialize parent.
```

```
// Initialize the queue so that BFS starts at s
```

```
qfront=0; qrear=1; Q[qfront]= s;
```

```
parent[s]=s;
```

```
component[s]= nComp;
```

```
while (qfront < qrear) // Q is not empty
```

```
    u= Q[qfront]; qfront++;
```

```
    for each neighbour v of u
```

```
        if (parent[v] == -1) // not visited
```

```
            parent[v]= u; component[v]= nComp;
```

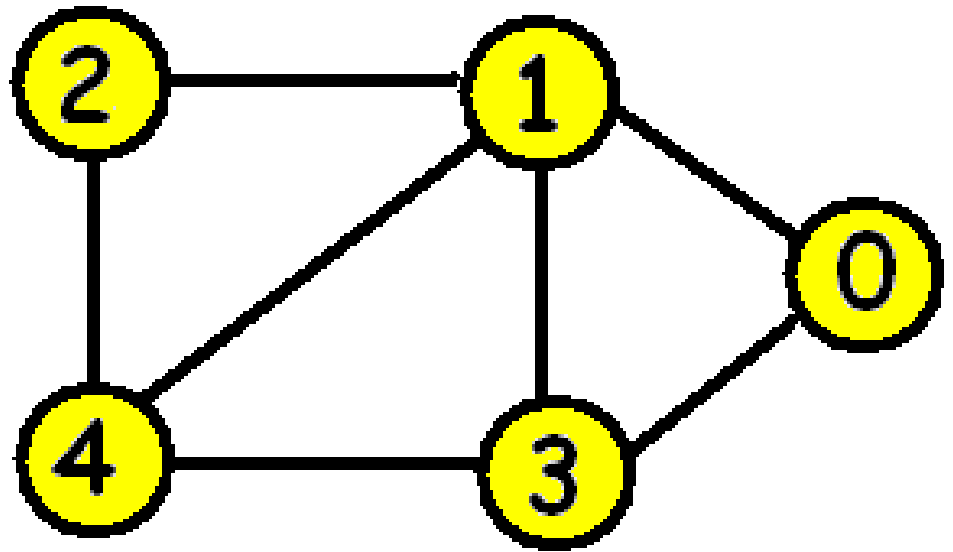
```
            Q[qrear]= v; qrear++;
```

```
        end if
```

```
    end for
```

```
end while
```

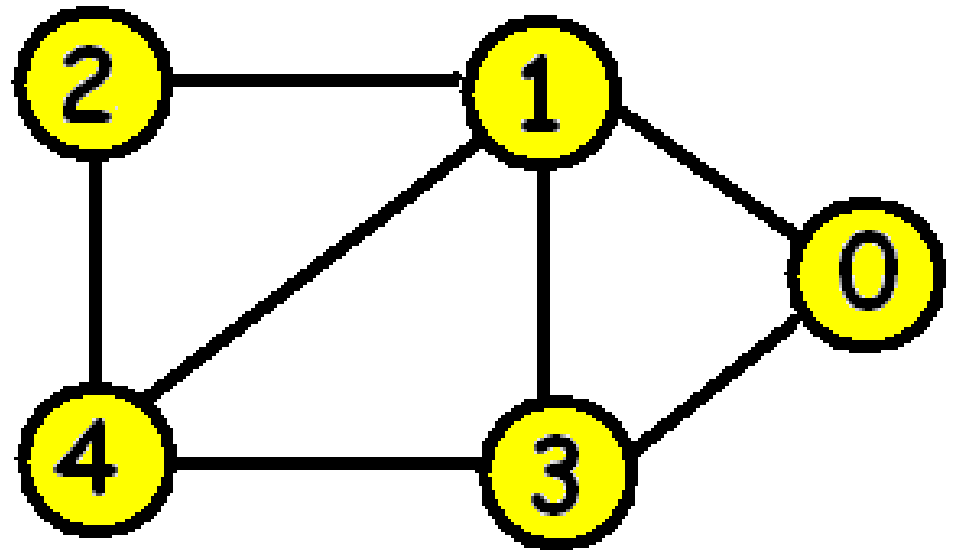
How much time does BFS take to identify the connected components of a graph when the data structure used for a graph is an adjacency matrix?



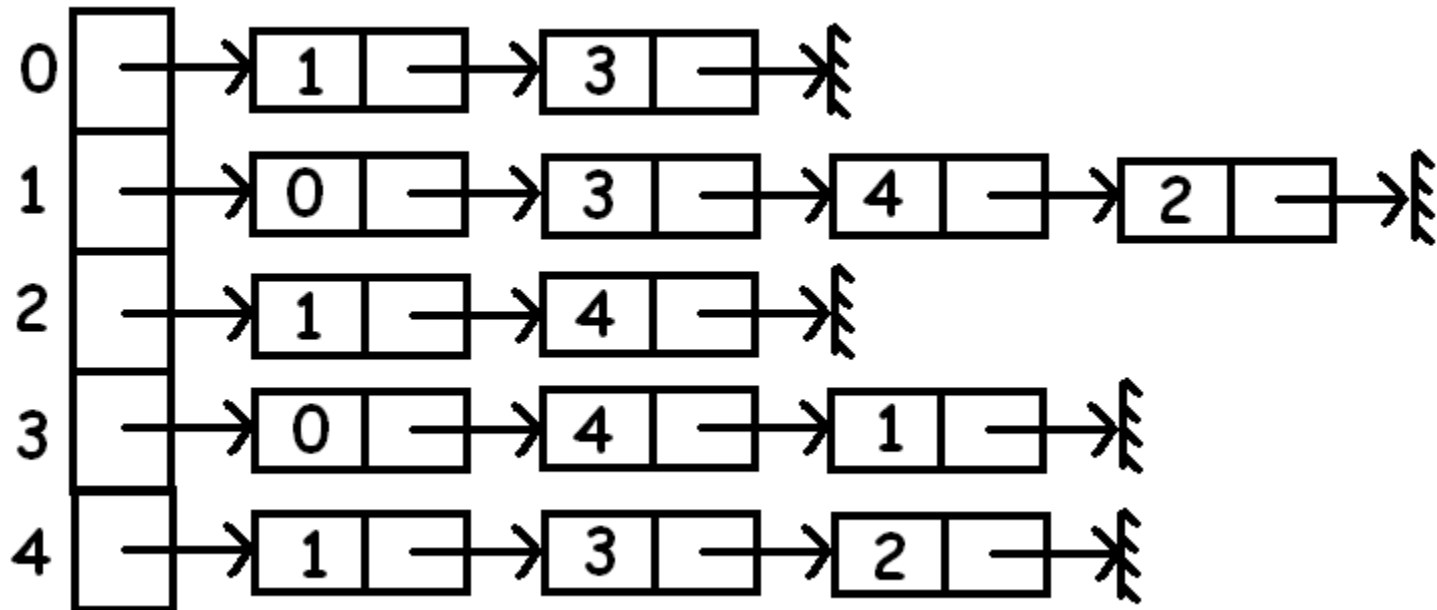
Adjacency matrix:

	0	1	2	3	4
0	0	1	0	1	0
1	1	0	1	1	1
2	0	1	0	0	1
3	1	1	0	0	1
4	0	1	1	1	0

How much time does BFS take to identify the connected components of a graph when the data structure used for a graph is an adjacency list?



Adjacency list:



How could you modify BFS to determine if v is a cut vertex?

.

A **bridge** with respect to a subgraph H of a graph G is either:

1. An edge $e=(u, v)$ which is not in H but both u and v are in H .
2. A connected component C of $G-H$ plus any edges that are incident to one vertex in C and one vertex in H plus the endpoints of these edges.

How can you find the bridges with respect to a cut vertex v ?