BFS

Week 9

22/2/24

```
#include<stdio.h>
#define MAX_VERTICES 10
int n, i, j, visited[MAX_VERTICES], queue[MAX_VERTICES],
    front = 0 , rear = 0;
int adj[MAX_VERTICES][MAX_VERTICES];
void bfs(int v)
 {
  visited[v] = 1 ;
  queue[rear ++]= v;
  while (front < rear)
      {
       int current = queue [front ++];
       printf (" %d\t", current);

       for (int i=0; i<n ; i++)
         {
           if(adj[current][i] && !visited[i])
              {
                visited[i] = 1;
                queue [rear ++]= i ;
              }
         }
      }
int main( )
 {
   int v;
   printf("Enter the number of vertices: ");
   scanf(" %d", &n);
   for (i = 0; i<n; i++)
      {
        visited[i]=0 ;
      }
   printf ("Enter graph data in matrix form:\n");
```

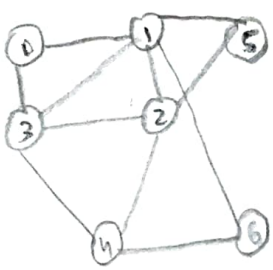
```
for (i=0;  i<n;  i++)
    for (j=0;  j<n;  j++)
        scanf("%d", &adj[i][j]);
printf("Enter the starting vertex: ");
scanf("%d", &v);
bfs(v);

for (i=0; i<n; i++)
  {
    if (!visited[i])
      {
        printf("\n BFS is not possible. Not all
                nodes are reachable.\n");
        return 0;
      }
  }
return 0;
}
```

o/p

```
Enter the number of vertices : 7
Enter graph data in matrix form:
0 1 0 1 0 0 0
1 0 1 1 0 1 1
0 1 0 1 1 1 0
0 0 1 1 0 0 1
0 1 1 0 0 0 0
0 1 0 0 1 0 0
Enter the starting vertex : 4
4 2 3 6 1 5 0
```

4, 2, 3, 6, 1, 5, 0

visited 4, 2, 3, 6



4 -> 2, 3, 6

2 -> 1, 3, 4, 5, 6

1 -> 0, 2, 3, 5, 6

```
Enter the number of vertices: 7
Enter graph data in matrix form:
0 1 0 1 0 0 0
1 0 1 1 0 1 1
0 1 0 1 1 1 0
1 1 1 0 1 0 0
0 0 1 1 0 0 1
0 1 1 0 0 0 0
0 1 0 0 1 0 0
Enter the starting vertex: 4
4       2       3       6       1       5       0
Process returned 0 (0x0)   execution time : 194.033 s
Press any key to continue.
```

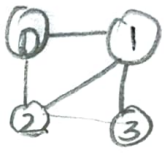
dfs :

```
#include <stdio.h>
#include<stdbool.h>

#define MAX_VERTICES 10

int n, i, j, visited[MAX_VERTICES];
int adj[MAX_VERTICES][MAX_VERTICES];

void dfs(int v)
  {
   visited[v]=1;
   for(int i=0; i<n; i++)
     {
      if(adj[v][i] && !visited[i])
        {
          dfs(i);
        }
     }
  }
int main()
 {
  int v;
  printf("Enter the number of vertices: ");
  scanf("%d", &n);
  for(i=0; i<n; i++).
   {
     visited[i] = 0;
   }
  printf("Enter graph data in matrix form: \n");
  for(i=0; i<n; i++)
     for(j=0; j<n; j++)
        scanf("%d", &adj[i][j]);
```

```
printf("Enter the starting vertex : ");
scanf("%d", &v);
dfs(v);

for(i=0; i<n; i++){
    if(!visited[i])
    {
        printf("\n The graph is not connected.\n");
        return 0;
    }
}
printf("\n The graph is connected.\n");
return 0;
}
```

O/P:

Enter the number of vertices: 4
Enter the graph data in matrix form:



0 1 1 0
1 0 0 1
1 0 0 0
0 1 0 0

Enter the ~~vertex~~ starting vertex : 0
The graph is connected.

Sp.f
22/2/24.

C:\Users\BMSCE\OneDrive\De

```
Enter the number of vertices: 4
Enter graph data in matrix form:
0 1 1 0
1 0 0 1
1 0 0 0
0 1 0 0
Enter the starting vertex: 0

The graph is connected.
```

~~test~~
leetcode 4

Rotate List

> Given the head of a linked list, rotate the list to the right by K places.
>

```
struct ListNode* rotateRight(struct ListNode *head, int k)
{
    if(head == NULL || head->next == NULL || k == 0)
        return head;
    int length = 1;
    struct ListNode *tail = head;
    while(tail->next != NULL)
    {
        length++;
        tail = tail->next;
    }
    k = k % length;
    if(k == 0)
        return head;
    struct ListNode *new_tail = head;
    for(int i=0; i<length-k-1; i++)
    {
        new_tail = new_tail->next;
    }
    struct ListNode *new_head = new_tail->next;
    new_tail->next = NULL;
    tail->next = head;
    return new_head;
}
```

o/p :-

Case 1

head =
[1, 2, 3, 4, 5]

k =
2

case 2

head =
[0, 1, 2]

k =
4