22-09-24

Breadth First Search

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

struct queue {
 int items[SIZE];
 int front;
 int rear;
};

struct node {
     int vertex;
     struct node* next;
 };

struct Graph {
     int numVertices;
     struct node** adjLists;
     int* visited;
};

void bfs(struct Graph* graph, int startVertex)
{
   struct queue* q = createQueue();

   graph->visited[startVertex] = 1;

   enqueue(q, startVertex);
```

```
while(!isEmpty(q))
{
  printQueue(q);
  int currentVertex = dequeue(q);
  printf("visited %d \n", currentVertex);

  struct node* temp = graph->adjLists
                      [currentVertex];

  while(temp){
    int adjVertex = temp->vertex;
    if(graph->visited[adjVertex]==0)
    {
      graph->visited[adjVertex] = 1;
      enqueue(q, adjVertex);
    }
    temp = temp->next;
  }
}
}

void addEdge(struct Graph* graph, int src,
             int dest)
{
  struct node* newNode = createNode
                         (dest);
  newNode->next = graph->adjLists[src
  graph->adjLists[dest] = newNode;
}
```

```
void enqueue(struct queue *q, int value)
{
    if (q->rear == size-1)
    {
      printf("Queue is full\n");
    }
    else {
      if (q->front == -1)
      {
        q->front=0;
      }

      q->rear++;
      q->items[q->rear]=value;
    }
int deque(struct queue *q)
{
    int item;
    if (isEmpty(q))
    {
      printf("Queue is Empty");
    }
    else
    {
      item=q->items[q->front];
      q->front++;
      if (q->front > q->rear)
      {
        printf("Resetting queue");
```

```
        q->front = q->rear = -1;
    }
}
return item;
}
```

Output:-

Adjacency list of vertex 0

2 -> 1 ->

Adjacency list of vertex 1

2 -> 0 ->

Adjacency list of vertex 2

3 -> 1 -> 0 ->

Adjacency list of vertex 3

2 ->

visited 2
visited 3
visited 1
visited 0.

DFS

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

struct node{
    int vertex;
    struct node* next;
};

struct Graph{
    int numvertices;
    int* visited;
    struct node** adjlists;
};

void DFS(struct Graph* graph, int vertex)
{
    struct node* adjlist = graph->adjlists[vertex];
    struct node* temp = adjlist;
    graph->visited[vertex] = 1;
    printf("visited %d \n", vertex);
    while(temp != NULL)
    {
        int connectedvertex = temp->vertex;
        if(graph->visited[connectedvertex] == 0)
        {
            DFS(graph, connectedvertex);
        }
        temp = temp->next;
    }
```

```
void addEdge(struct Graph* graph, int src, int dest)
{
    struct node * newNode = createNode(dest);
    newNode -> next = graph -> adjLists[src];
    graph -> adjLists[src] = newNode;

    newNode = createNode(src);
    newNode -> next = graph -> adjLists[src];
    graph -> adjLists[dest] = newNode;
}
void printGraph( struct Graph* graph)
{
    int v;
    for (v = 0; v < graph -> numVertices; v++)
    {
        struct node * temp = graph -> adjLists
        printf("\n Adjacency lists of vertex
                                        v);
        while (temp)
        {
            printf("%d -> ", temp -> vertex);
            temp = temp -> next;
        }
    printf("\n");
    }
}
```

output:-

Adjacency list of vertex 0.

2 -> 1 ->.

Adjacency list of vertex 1.

2 -> 0 ->

Adjacency list of vertex 2

3 -> 1 -> 0 ->

Adjacency list of vertex 3

2 ->

visited 2

visited 3

visited 1

visited 0.

26.02.24