BFS

```
#include <stdio.h>
void bfs(int a[10][10], int n, int u){
  int f, r, q[10], v;
  int s[10] = {0};
  printf("The nodes visited from %d:", u);
  f = 0;
  r = -1;
  q[++r] = u;
  s[u] = 1;
  printf("%d", u);
  while (f<=r){
    u = q[f++];
    for (v=0; v<n; v++){
      if((a[u][v] == 1) && (s[v] == 0)){
        printf("%d", v);
        s[v] = 1;
        q[++r] = v;
      }
    }
  }
  printf("\n");
}
void main(){
  int n, a[10][10], source, i, j;
  printf("\n Enter no of nodes:");
  scanf("%d", &n);
  printf("\n Enter the adjacency matrix:\n");
  for(i=0; i<n; i++){
    scanf("%d", &a[i][j]);
  for(j=0; j<n; j++){
```

```
        scanf("%d", &a[i][j]);
    }
}

for (source = 0; source < n; source++){
        bfs(a, n, source);
    }
}
```

Output:

Enter no of nodes : 4
Enter the adjacency matrix :
1 1 0 0
1 1 1 0
0 0 1 1
0 0 1 0

The node visited from 0 : 0 1 2 3
The node visited from 1 : 1 0 2 3
The node visited from 2 : 2 3
The node visited from 3 : 3 2

DFS

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

#define Max_vertices 20

int graph [max_vertices] [Max_vertices];
int visited [Max_vertices];
int n;

void dfs ( int start){
    int i;
    visited [start] = 1;
    printf ("visited %d \n", start);

    for (i=0; i<n ; i++) {
        if ( graph [start][i] && ! visited [i]){
            dfs(i);
        }
    }
}
int main() {
    int i, j ; int start;
    printf ("Enter the number of vertices:");
    scanf ("%d", &n);

    printf ("Enter the adjacency matrix:\n");
    for (i=0 ; i<n; i++) {
        for (j=0; j<n; j++) {
            scanf ("%d", & graph [i][j]);
        }
    }
```

```
printf("Enter the starting vertex for DFS:");
scanf("%d", &start);
for(i=0; i<n; i++){
    visited[i] = 0;
}
dfs(start);
return 0;
}
```

Output:

```
Enter the number of vertices: 3
Enter the adjacency matrix:
1 0 1
0 0 1
1 1 1
Enter the starting vertex for DFS: 1
visited 1
visited 2
visited 0
```

SURYA Gold
Date Page

Delete Node in BST:

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

struct Node {
    int data;
    struct Node *left, *right;
};

struct Node* createNode(int value) {
    struct Node* newNode = (struct Node*)
    malloc(sizeof(struct Node));
    newNode->data = value;
    newNode->left = newNode->right = NULL;
    return newNode;
}

struct Node* minValueNode(struct Node* node) {
    struct Node* current = node;
    while (current && current->left != NULL)
        current = current->left;
    return current;
}

struct Node* deleteNode(struct Node* root,
    int key) {
    if (root == NULL)
        return root;

    if (key < root->value)
        root->left = deleteNode(root->left, key);
```

```
else if (key > root->val)
    root->right = deleteNode(root->right, key);
else {
    if (root->left == NULL) {
        struct Node* temp = root->right;
        free(root);
        return temp;
    } else if (root->right == NULL) {
        struct Node* temp = root->left;
        free(root);
        return temp;
    }

    struct Node* temp = minValueNode(root->right);
    root->val = temp->val;
    root->right = deleteNode(root->right,
                             temp->val);
    }
    return root;
}

void inorder(struct Node* root) {
    if (root != NULL) {
        inorder(root->left);
        printf("%d", root->val);
        inorder(root->right);
    }
}
```

26.02.24