DFS

#include <stdio.h>

#include <stdlib.h>

int time = 1;

struct node

{

  int vertex;

  struct node \*next;

};

struct node \*createNode(int v)

{

  struct node \*newNode = malloc(sizeof(struct node));

  newNode->vertex = v;

  newNode->next = NULL;

  return newNode;

}

struct Graph

{

  int numVertices;

  struct node \*\*adjLists;

};

struct Graph \*createGraph(int vertices)

{

  struct Graph \*graph = malloc(sizeof(struct Graph));

  graph->numVertices = vertices;

  graph->adjLists = malloc(vertices \* sizeof(struct node \*));

  int i;

  for (i = 0; i < vertices; i++)

    graph->adjLists[i] = NULL;

  return graph;

}

void addEdge(struct Graph \*graph, int src, int dest)

{

*// Add edge from src to dest*

  struct node \*newNode = createNode(dest);

  newNode->next = graph->adjLists[src];

  graph->adjLists[src] = newNode;

}

void printGraph(struct Graph \*graph)

{

  int v;

  for (v = 0; v < graph->numVertices; v++)

  {

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

    printf("\n Adjacency list of vertex %d\n ", v);

    while (temp)

    {

      printf("%d -> ", temp->vertex);

      temp = temp->next;

    }

    printf("NULL");

    printf("\n");

  }

}

void dfs\_visit(struct Graph \*graph, char \*color, int \*pi, int \*d, int \*f, int v)

{

  color[v] = 'g';

  d[v] = time++;

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

  while (temp)

  {

    int u = temp->vertex;

    if (color[u] == 'w')

    {

      pi[u] = v;

      dfs\_visit(graph, color, pi, d, f, u);

    }

    temp = temp->next;

  }

  color[v] = 'b';

  f[v] = time++;

}

void dfs(struct Graph \*graph, char \*color, int \*pi, int \*d, int \*f, int source)

{

  dfs\_visit(graph, color, pi, d, f, source);

  for (int v = 0; v < graph->numVertices; v++)

  {

    if (color[v] == 'w')

    {

      dfs\_visit(graph, color, pi, d, f, v);

    }

  }

}

void printpath(struct Graph \*graph, int v, int \*pi)

{

  if (pi[v] == -1)

    printf(" %d", v);

  else

  {

    printpath(graph, pi[v], pi);

    printf(" -> %d", v);

  }

}

int main()

{

  int numVertices;

  printf("Enter the number of vertices: ");

  scanf("%d", &numVertices);

  struct Graph \*graph = createGraph(numVertices);

  int src, dest;

  printf("Enter edges (source destination) [Enter -1 -1 to stop]:\n");

  while (1)

  {

    scanf("%d %d", &src, &dest);

    if (src >= numVertices || dest >= numVertices)

    {

      printf("Invalid edge!\n");

      continue;

    }

    if (src == -1 && dest == -1)

      break;

    addEdge(graph, src, dest);

  }

  printGraph(graph);

  char color[graph->numVertices];

  int d[graph->numVertices], f[graph->numVertices], pi[graph->numVertices];

  for (int v = 0; v < graph->numVertices; v++)

  {

    color[v] = 'w';

    d[v] = pi[v] = f[v] = -1;

  }

  printf("\nEnter the source vertex: ");

  int source;

  scanf("%d", &source);

  printf("\nDepth First Search\n");

  dfs(graph, color, pi, d, f, source);

  printf("\nVertex\tColor\tDiscovery Time\tFinish Time\tPredecessor\n");

  for (int v = 0; v < graph->numVertices; v++)

    printf("%d\t%c\t%d\t\t%d\t\t%d\n", v, color[v], d[v], f[v], pi[v]);

  printf("\nPredecessor Subgraph\n");

  for (int v = 0; v < graph->numVertices; v++)

  {

    if (pi[v] != -1 && v != source)

    {

      printf("Path from %d to %d:", source, v);

      printpath(graph, v, pi);

      printf("\n");

    }

  }

  return 0;

}