|  |
| --- |
| **Computer Engineering Department - ITU** |
| **CE200L: Data Structures & Algorithms Lab** |

|  |  |
| --- | --- |
| **Course Instructor: Usama Bin Shakeel** | **Dated: 15/11/2022** |
| **Teaching Assistant: Muhammad Sufyan Ashraf** | **Semester: Fall 2022** |
| **Lab Engineer: Nadir Abbas** | **Batch: BSCE2021** |

# **Lab 12A. Basic Operations on Graph using Arrays**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Roll number** | **Report**  **(out of 100)** | **Scaled to 10** | **Total**  **(out of 10)** |
|  |  |  |  |  |

Checked on: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## **Objective**

The objective of this lab is to provide the knowledge of basic data structures and their implementations.

## **Equipment and Component**

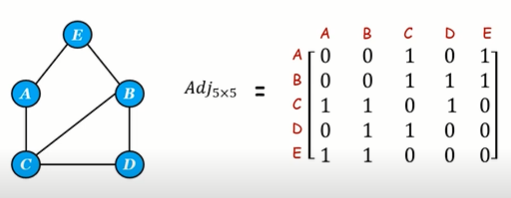
|  |  |  |
| --- | --- | --- |
| **Component Description** | **Value** | **Quantity** |
| Computer | Available in lab | 1 |

## **Conduct of Lab**

1. Students are required to perform this experiment individually.
2. In case the lab experiment is not understood, the students are advised to seek help from the course instructor, lab engineers, assigned teaching assistants (TA) and lab attendants.

## **Theory and Background**

In computer science, A **graph** data structure is a collection of nodes that have data and are connected to other nodes. Let's try to understand this through an example. On Facebook, everything is a node. That includes User, Photo, Album, Event, Group, Page, Comment, Story, Video, Link, note...anything that has data is a node. Every relationship is an edge from one node to another. Whether you post a photo, join a group, like a page, etc., a new edge is created for that relationship. More precisely, a graph is a data structure (V, E) that consists of a collection of vertices V and a collection of edges E, represented as ordered pairs of vertices (u, v).



A **pointer** is a variable that stores the address of another variable. Unlike other variables that hold values of a certain type, pointer holds the address of a variable. For example, an integer variable holds (or you can say stores) an integer value, however an integer pointer holds the address of a integer variable.

**Arrays** are used to store multiple values in a single variable, instead of declaring separate variables for each value. To declare an array, define the variable type, specify the name of the array followed by square brackets and specify the number of elements it should store: string cars[4];

**Templates** are a feature of the C++ programming language that allows functions and classes to operate with generic types. This allows a function or class to work on many different data types without being rewritten for each one.

**Lab Task**

**Task A**

As you have implemented the trees with arrays before, Now implement graph with arrays. Implement the following functions:

* Add elements (edges, nodes)
* Update elements (edges, nodes)
* Delete elements (edges, nodes)
* Check if the element is present in the graph
* Display graph

Make all necessary functions and handle all corner cases. Make a menu driven program.

|  |
| --- |
| // Paste your code here  **FUNCTION.H:**  #include <iostream> #include <iomanip>  using namespace std;  class graph { public:  int ver; //declaring  int \*\*mat;   graph(int v) {  ver = v; //copying  mat = new int \*[ver]; //allocating memory to rows  for (int i = 0; i < ver; i++) {  mat[i] = new int[ver]; //allocating memory to each column  for (int j = 0; j < ver; j++) {  mat[i][j] = 0; //at first storing zeros  }  }  }   void displayMat() {  cout << "\n\n" << setw(4) << " ";  for (int i = 0; i < ver; i++) {  cout << setw(3) << "( " << i << " )"; //displaying the nodes 0,1,2,3 etc.  }  cout << "\n\n";  for (int i = 0; i < ver; i++) {  cout << setw(3) << "( " << i << " )" << " "; //displaying the nodes  for (int j = 0; j < ver; j++) {  cout << setw(4) << mat[i][j]  << " "; //displaying that weather the nodes is present at that point or not by zero or 1  }  cout << "\n\n";  }  }   void addEdges() {  for (int i = 0; i < ver; i++) {  for (int j = 1; j < ver; j++) { //loops for adding an edge  if (i != j) { //if i and j are not equal.  cout << "ENTER 1 IF THE VERTEX " << i << " IS ADJACENT TO " << j << ",OTHERWISE ENTER 0";  cin  >> mat[i][j]; //entering zero and 1,1 if the nodes are present and zero if not  if (mat[i][j] != 1) {  if (mat[i][j] !=  0) { //checking if the user has enter a number other than 1 and zero  cout << "INVALID ARGUMENT." << endl;  exit(3);  }  }  mat[j][i] = mat[i][j]; //bcz the direction does not matter  } else {  mat[i][j] = 0; //if the i and j are equal then store zero  }  }  }  }   void deleteEdge(int v1, int v2) {   cout << "ENTER V1 = "; //taking vertexes  cin >> v1;  cout << "ENTER V2 = ";  cin >> v2;  if (v1 >= ver) { //checking if they are smaller than the vertex given by us in the start  cout << "VERTEX NOT PRESENT." << endl;  } else {  if (v2 >= ver) { //checking again  cout << "VERTEX NOT PRESENT." << endl;  } else {  mat[v1][v2] = 0; //then storing it to zero  }  }  }   void deleteNode(int node) {  cout << "ENTER NODE = ";  cin >> node; //taking node  if (node < ver) {  for (int i = 0; i < ver; i++) {  mat[i][node] = 0; //then putting every node to zero  for (int j = 0; j < ver; j++) {  mat[node][j] = 0;  }  }  }  }   void update(int v1, int v2) {   cout << "ENTER VERTEX V1 WHICH YOU WANT TO UPDATE = "; //taking vertexes  cin >> v1;  cout << "ENTER VERTEX V2 WHICH YOU WANT TO UPDATE = ";  cin >> v2;  if (v1 >= ver) {  cout << "VERTEX NOT PRESENT." << endl; //checking if small  } else {  if (v2 >= ver) {  cout << "VERTEX NOT PRESENT." << endl;  } else {  if (mat[v1][v2] == 1) {  mat[v1][v2] = 0; //checking if the value is 1 then putting it to zero  } else {  mat[v1][v2] = 1; //otherwise 1  }  }  }  }   void check(int v1, int v2) {  cout << "ENTER VERTEX V1 WHICH YOU WANT TO UPDATE = ";  cin >> v1;  cout << "ENTER VERTEX V2 WHICH YOU WANT TO UPDATE = ";  cin >> v2;  if (v1 >= ver) {  cout << "VERTEX NOT PRESENT." << endl;  } else {  if (v2 >= ver) {  cout << "VERTEX NOT PRESENT." << endl;  } else {  if (mat[v1][v2] == 1) { //if the value is 1 then it means the element is 1  cout << "THE ELEMENT IS PRESENT." << endl;  } else {  cout << "THE ELEMENT IS NOT PRESENT." << endl;  }  }  }   } };  **MAIN.CPP:**  #include <iostream> #include "Functions.h"  using namespace std;  int main() {  int v;  cout<<"ENTER THE NUMBER OF VERTEX = ";  cin>>v;  graph g(v);  int v1;  int v2; //declaring  int opt;  do {  cout << "CHOOSE OPTION." << endl; //displaying the options  cout << "1.ADD EDGE." << endl;  cout << "2.DELETE EDGE." << endl;  cout << "3.UPDATE EDGE." << endl;  cout << "4.UPDATE NODE." << endl;  cout << "5.CHECK EDGE." << endl;  cout << "6.DISPLAY." << endl;  cout << "7.EXIT." << endl;  cin >> opt;  if (opt == 1) {  g.addEdges();  g.displayMat();  }  if (opt == 2) {  g.deleteEdge(v1, v2);  g.displayMat();  }  if (opt == 3) {  g.update(v1, v2);  g.displayMat();  }  if(opt==4){  int node;  g.deleteNode(node);  g.displayMat();  }  if (opt == 5) {  g.check(v1, v2); //calling  }  if (opt == 6) {  g.displayMat();  }  if (opt == 7) {  cout << "YOU CHOOSE TO EXIT." << endl;  exit(4);  }  } while (opt >= 1 && opt <= 6);   return 0; }  // Paste your output here    A picture containing text  Description automatically generated  Text  Description automatically generated |

#### **Assessment Rubric for Lab**

**Method for assessment:**

Lab reports and instructor observation during lab sessions. Outcome assessed:

a. Ability to conduct experiments, as well as to analyze and interpret data (P) b. Ability to function on multi-disciplinary teams (A)

c. Ability to use the techniques, skills, and modern engineering tools necessary for engineering practice (P)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Performance metric** | **Task** | **CLO** | **Description** | **Max marks** | **Exceeds expectation** | **Meets expectation** | **Does not meet expectation** | **Obtained marks** |
| 1. Realization of experiment (a) | 1 | 1 | Functionality | 40 | Executes without errors excellent user prompts, good use of symbols, spacing in output. Through testing has been completed (35-40) | Executes without errors, user prompts are understandable, minimum use of symbols or spacing in output. Some testing has been completed (20-34) | Does not execute due to syntax errors, runtime errors, user prompts are misleading or non-existent. No testing has been completed (0-19) |  |
| 2. Teamwork (b) | 1 | 3 | Group Performance | 5 | Actively engages and cooperates with other group member(s) in effective manner (4-5) | Cooperates with other group member(s) in a reasonable manner but conduct can be improved (2-3) | Distracts or discourages other group members from conducting the experiment (0-1) |  |
| 3. Conducting experiment (a, c) | 1 | 1 | On Spot Changes | 10 | Able to make changes (8-10) | Partially able to make changes (5-7) | Unable to make changes (0-4) |  |
| 1 | 1 | Viva | 10 | Answered all questions (8-10) | Few incorrect answers (5-7) | Unable to answer all questions (0-4) |  |
| 4. Laboratory safety and disciplinary rules (a) | 1 | 3 | Code commenting | 5 | Comments are added and does help the reader to understand the code (4-5) | Comments are added and does not help the reader to understand the code (2-3) | Comments are not added (0-1) |  |
| 5. Data collection (c) | 1 | 3 | Code Structure | 5 | Excellent use of white space, creatively organized work, excellent use of variables and constants, correct identifiers for constants, No line-wrap (4-5) | Includes name, and assignment, white space makes the program fairly easy to read. Title, organized work, good use of variables (2-3) | Poor use of white space (indentation, blank lines) making code hard to read, disorganized and messy (0-1) |  |
| 6. Data analysis (a, c) | 1 | 4 | Algorithm | 20 | Solution is efficient, easy to understand, and maintain (15-20) | A logical solution that is easy to follow but it is not the most efficient (6-14) | A difficult and inefficient solution (0-5) |  |
| 7. Computer use (c) | 1 | 2 | Documentation & Github Submissions | 5 | Timely (4-5) | Late (2-3) | Not done (0-1) |  |
|  | Max Marks (total): | | | 100 | Obtained Marks (total): | | |  |

Lab Engineer Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_