Computational Lab - Assignment 3: Directed Acyclic Graph (DAG) with DOT visualization

By: Pankaj Azad, Roll Number - 14M517, M.Tech Ist Sem

Enter number of vertices...13

Enter the edges (E.g. 4 3 means a directed edge from 4 to 3). -1 -1 as input stops the process

0 5

0 1

0 6

2 0

2 3

3 5

5 4

6 4

6 9

7 6

8 7

9 11

9 12

9 10

11 12

-1 -1

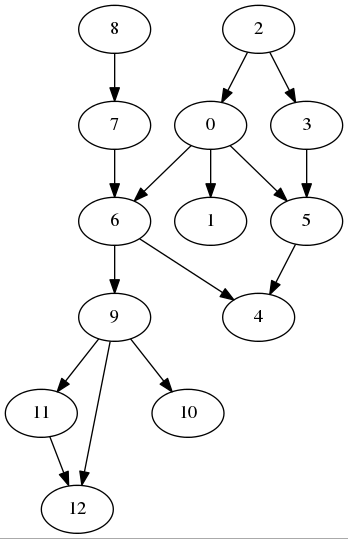
Enter any of the following available choices

1 - Add Edge(s)

2 - Delete Edge(s)

3 - Print Adjacency list and draw graph (generates a png image)

4 - Depth First Search (DFS)

5 - Detect existence of cycle

9 - Exit

3

DAG: 13 Vertices and 15 Edges

[0] => 5 1 6

[1] =>

[2] => 0 3

[3] => 5

[4] =>

[5] => 4

[6] => 4 9

[7] => 6

[8] => 7

[9] => 11 12 10

[10] =>

[11] => 12

[12] =>

Enter any of the following available choices

1 - Add Edge(s)

2 - Delete Edge(s)

3 - Print Adjacency list and draw graph (generates a png image)

4 - Depth First Search (DFS)

5 - Detect existence of cycle

9 - Exit

4

Enter the source node for DFS...2

DFS from 2 : 2, 0, 5, 4, 1, 6, 9, 11, 12, 10, 3,

Enter any of the following available choices

1 - Add Edge(s)

2 - Delete Edge(s)

3 - Print Adjacency list and draw graph (generates a png image)

4 - Depth First Search (DFS)

5 - Detect existence of cycle

9 - Exit

5

No Cycle found!!

Enter any of the following available choices

1 - Add Edge(s)

2 - Delete Edge(s)

3 - Print Adjacency list and draw graph (generates a png image)

4 - Depth First Search (DFS)

5 - Detect existence of cycle

9 - Exit

1

Enter the edges (E.g. 4 3 means a directed edge from 4 to 3). -1 -1 as input stops the process

12 0

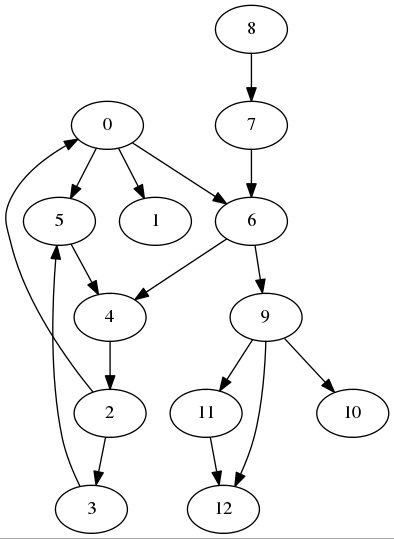
-1 -1

Enter any of the following available choices

1 - Add Edge(s)

2 - Delete Edge(s)

3 - Print Adjacency list and draw graph (generates a png image)

4 - Depth First Search (DFS)

5 - Detect existence of cycle

9 - Exit

5

Edge 12,0 is involved in cycle

Cycle Exits!!

Enter any of the following available choices

1 - Add Edge(s)

2 - Delete Edge(s)

3 - Print Adjacency list and draw graph (generates a png image)

4 - Depth First Search (DFS)

5 - Detect existence of cycle

9 - Exit

3

DAG: 13 Vertices and 16 Edges

[0] => 5 1 6

[1] =>

[2] => 0 3

[3] => 5

[4] =>

[5] => 4

[6] => 4 9

[7] => 6

[8] => 7

[9] => 11 12 10

[10] =>

[11] => 12

[12] => 0

Enter any of the following available choices

1 - Add Edge(s)

2 - Delete Edge(s)

3 - Print Adjacency list and draw graph (generates a png image)

4 - Depth First Search (DFS)

5 - Detect existence of cycle

9 - Exit

2

Enter the edges (E.g. 4 3 means a directed edge from 4 to 3). -1 -1 as input stops the process

0 6

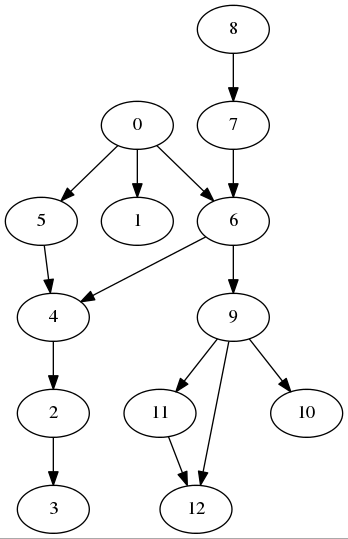
-1 -1

Enter any of the following available choices

1 - Add Edge(s)

2 - Delete Edge(s)

3 - Print Adjacency list and draw graph (generates a png image)

4 - Depth First Search (DFS)

5 - Detect existence of cycle

9 - Exit

3

DAG: 13 Vertices and 15 Edges

[0] => 5 1

[1] =>

[2] => 0 3

[3] => 5

[4] =>

[5] => 4

[6] => 4 9

[7] => 6

[8] => 7

[9] => 11 12 10

[10] =>

[11] => 12

[12] => 0

Enter any of the following available choices

1 - Add Edge(s)

2 - Delete Edge(s)

3 - Print Adjacency list and draw graph (generates a png image)

4 - Depth First Search (DFS)

5 - Detect existence of cycle

9 - Exit

9

SOURCE CODE

dag.h

#include <iostream>

#include <list>

#include <vector>

**using** **namespace** std**;**

enum Color **{** WHITE**,** GREY**,** BLACK **};**

class Dag

**{**

public**:**

Dag**(**int vertices**);**

**~**Dag**();**

int vertices**()** **{** **return** numberOfVertices**;** **}**

int edges**()** **{** **return** numberOfEdges**;** **}**

bool addEdge**(**int src**,** int dst**);**

void addEdges**();**

bool deleteEdge**(**int src**,** int dst**);**

void deleteEdges**();**

vector**<**int**>&** performDfsFromGivenSource**();**

void dfs**(**int sourceVertex**);**

bool doesCycleExist**();**

bool visit**(**int vertex**);**

friend ostream **&operator** **<<** **(**ostream **&**out**,** const Dag **&**dag**);**

private**:**

int numberOfEdges**;**

int numberOfVertices**;**

list**<**int**>\*** adjList**;**

char**\*** colorList**;**

vector**<**int**>** dfsOutputList**;**

**};**

Dag.cpp

#include "dag.h"

#include <fstream>

#include <cstdlib>

#include <string>

#include <sstream>

**using** **namespace** std**;**

Dag**::**Dag**(**int vertices**)**

**{**

numberOfVertices **=** vertices**;**

numberOfEdges **=** 0**;**

adjList **=** **new** list**<**int**>[**numberOfVertices**];**

colorList **=** **new** char**[**numberOfVertices**];**

**}**

Dag**::~**Dag**()**

**{**

**delete** adjList**;**

**delete** colorList**;**

**}**

bool Dag**::**addEdge**(**int src**,** int dst**)**

**{**

bool ret **=** **false;**

**if(** src **<** numberOfVertices **&&** dst **<** numberOfVertices**)**

**{**

adjList**[**src**].**push\_back**(**dst**);**

numberOfEdges**++;**

ret **=** **true;**

**}**

**return** ret**;**

**}**

bool Dag**::**deleteEdge**(**int src**,** int dst**)**

**{**

bool ret **=** **false;**

**if(** src **<** numberOfVertices **&&** dst **<** numberOfVertices**)**

**{**

adjList**[**src**].**remove**(**dst**);**

numberOfEdges**--;**

ret **=** **true;**

**}**

**return** ret**;**

**}**

ostream **&operator** **<<** **(**ostream **&**out**,** const Dag **&**dag**)**

**{**

static int dotfileNumber**=**0**;**

dotfileNumber**++;**

out **<<** "DAG: " **<<** dag**.**numberOfVertices **<<** " Vertices and " **<<** dag**.**numberOfEdges **<<** " Edges " **<<** endl**;**

ofstream dotfile**;**

ostringstream dotfileName**;**

dotfileName **<<**"dag" **<<** dotfileNumber **<<** ".dot"**;**

dotfile**.**open**(**dotfileName**.**str**().**c\_str**(),**ofstream**::**out**);**

dotfile **<<** "digraph G {" **<<** endl**;**

**for** **(**int index **=** 0**;** index **<** dag**.**numberOfVertices**;** index**++)**

**{**

out **<<** "["**<<** index **<<** "] => " **;**

**for** **(**list**<**int**>::**iterator it**=**dag**.**adjList**[**index**].**begin**();** it **!=** dag**.**adjList**[**index**].**end**();** **++**it**)**

**{**

out **<<** **\***it **<<** " " **;**

dotfile **<<** "\t "**<<** index **<<** " -> " **<<** **\***it **<<** ";" **<<** endl**;**

**}**

cout **<<** endl**;**

**}**

dotfile **<<** "}" **<<** endl**;**

dotfile**.**close**();**

ostringstream dagGenerationCommand**;**

dagGenerationCommand **<<** "dot -Tpng dag" **<<** dotfileNumber **<<** ".dot -o dag" **<<** dotfileNumber **<<** ".png"**;**

system**(**dagGenerationCommand**.**str**().**c\_str**());**

ostringstream imageOpeningCommand**;**

imageOpeningCommand **<<** "ristretto dag" **<<** dotfileNumber **<<** ".png" **<<** "&"**;**

system**(**imageOpeningCommand**.**str**().**c\_str**());**

**return** out**;**

**}**

void Dag**::**addEdges**()**

**{**

int src**,** dst**;**

cout **<<** "Enter the edges (E.g. 4 3 means a directed edge from 4 to 3). -1 -1 as input stops the process" **<<** endl**;**

**do**

**{**

cin **>>** src **>>** dst**;**

**if(**src**!=-**1 **&&** **!**addEdge**(**src**,**dst**))**

cout **<<**"Failed to add this edge, (Try again with vertex numbers between range (0,TotalVertices -1))" **<<** endl**;**

**}while(**src**!=-**1**);**

**}**

void Dag**::**deleteEdges**()**

**{**

int src**,** dst**;**

cout **<<** "Enter the edges (E.g. 4 3 means a directed edge from 4 to 3). -1 -1 as input stops the process" **<<** endl**;**

**do**

**{**

cin **>>** src **>>** dst**;**

**if(**src**!=-**1 **&&** **!**deleteEdge**(**src**,**dst**))**

cout **<<**"Failed to add this edge, (Try again with vertex numbers between range (0,TotalVertices -1))" **<<** endl**;**

**}while(**src**!=-**1**);**

**}**

vector**<**int**>&** Dag**::**performDfsFromGivenSource**()**

**{**

int src**;**

cout **<<** "Enter the source node for DFS..."**;**

cin **>>** src**;**

dfsOutputList**.**clear**();**

**for(**int i**=**0**;**i**<**numberOfVertices**;**i**++)**

**{**

colorList**[**i**]** **=** WHITE**;**

**}**

dfs**(**src**);**

cout **<<** "DFS from " **<<** src **<<** " : "**;**

**for(**int i**=**0**;**i**<**dfsOutputList**.**size**();**i**++)**

cout **<<** dfsOutputList**[**i**]** **<<** ", "**;**

cout **<<** endl**;**

**return** dfsOutputList**;**

**}**

void Dag**::**dfs**(**int sourceVertex**)**

**{**

colorList**[**sourceVertex**]** **=** BLACK**;**

dfsOutputList**.**push\_back**(**sourceVertex**);**

**for(**list**<**int**>::**iterator it **=** adjList**[**sourceVertex**].**begin**();** it **!=** adjList**[**sourceVertex**].**end**();** **++**it**)**

**{**

**if** **(** colorList**[\***it**]** **!=** BLACK **)**

dfs**(\***it**);**

**}**

**}**

bool Dag**::**doesCycleExist**()**

**{**

**for(**int i**=**0**;**i**<**numberOfVertices**;**i**++)**

**{**

colorList**[**i**]** **=** WHITE**;**

**}**

**for(**int i**=**0**;**i**<**numberOfVertices**;**i**++)**

**{**

**if(** colorList**[**i**]** **==** WHITE **)**

**{**

**if** **(** visit**(**i**)** **)**

**return** **true;**

**}**

**}**

**return** **false;**

**}**

bool Dag**::**visit**(**int v**)**

**{**

int u**;**

colorList**[**v**]** **=** GREY**;**

**for** **(**list**<**int**>::**iterator it **=** adjList**[**v**].**begin**();** it **!=** adjList**[**v**].**end**();** **++**it**)**

**{**

u **=** **\***it**;**

**if** **(** colorList**[**u**]** **==** GREY**)**

**{**

cout **<<** "Edge " **<<** v **<<** "," **<<** u **<<** " is involved in cycle" **<<** endl**;**

**return** **true;**

**}**

**else** **if** **(**colorList**[**u**]** **==** WHITE**)**

**{**

**if** **(** visit**(**u**)** **)**

**return** **true;**

**}**

**}**

colorList**[**v**]** **=** BLACK**;**

**return** **false;**

**}**

main.cpp

#include <iostream>

#include <cstdlib>

#include <cstdio>

#include "dag.h"

#ifdef \_\_linux\_\_

#define CLEAR\_SCREEN system("clear")

#elif \_WIN32

#define CLEAR\_SCREEN system("cls")

#endif

**using** **namespace** std**;**

int main**()**

**{**

int input**=**0**,**vertices**=**0**;**

CLEAR\_SCREEN**;**

cout **<<** "\nComputational Lab - Assignment 3: Directed Acyclic Graph (DAG) with DOT visualization"**;**

cout **<<** "\nBy: Pankaj Azad, Roll Number - 14M517, M.Tech Ist Sem\n\n\n" **<<** endl**;**

cout **<<** "Enter number of vertices..."**;**

cin **>>** vertices**;**

Dag dag**(**vertices**);**

dag**.**addEdges**();**

**do**

**{**

cout **<<** "\n\n\n"**;**

cout **<<** "Enter any of the following available choices" **<<** endl**;**

cout **<<** "1 - Add Edge(s)" **<<** endl**;**

cout **<<** "2 - Delete Edge(s)" **<<** endl**;**

cout **<<** "3 - Print Adjacency list and draw graph (generates a png image)" **<<** endl**;**

cout **<<** "4 - Depth First Search (DFS)" **<<** endl**;**

cout **<<** "5 - Detect existence of cycle" **<<** endl**;**

cout **<<** "9 - Exit" **<<** endl**;**

cin **>>** input**;**

**switch(**input**)**

**{**

**case** 1**:**

dag**.**addEdges**();**

**break;**

**case** 2**:**

dag**.**deleteEdges**();**

**break;**

**case** 3**:**

cout **<<** dag **<<** endl**;**

**break;**

**case** 4**:**

dag**.**performDfsFromGivenSource**();**

**break;**

**case** 5**:** cout **<<** **(** dag**.**doesCycleExist**()?** "\nCycle Exits!!\n"**:** "\nNo Cycle found!!\n"**)** **<<** endl**;**

**break;**

**case** 9**:** exit**(**0**);**

**break;**

**}**

**}while(**input**!=**9**);**

**return** 0**;**

**}**