**CS311 Yoshii HW6 - Intro to Graphs (based on Notes-10A)**

**DUE: Week 11 Saturday**

**Total: 24 points Your score is: Fix and Resubmit with HW7?**

**Your Name: Kyungbin Lee**

**Date Turned In: April 11, 2020**

**IMPORTANT REMINDERS:**

1. **Be able to draw the most crowded undirected graph (complete graph).**
2. **Be able to draw the most crowded directed graph (complete graph).**
3. **Be able to draw the least crowded connected graphs (directed/undirected).**
4. **Be able to say how many edges are there for 1-3 with N vertices.**
5. **Be able to do DFS and BFS of any tree.**

**Review Questions from Notes-7A:[6pts] Your score:**

1. **\*Inter4\* How many edges does a tree have (Given N nodes)? Why? (Relate this to the statement above.)[1]**

It should be N-1 edges. Node is vertex, so if there are three nodes, then tree will have 2 edges.

1. **In preparation for the next question, give the adjacency list tables to store the following graphs.**

**a) A ------ B b) A ----****B**

**| | | |**

**| | \|/ \|/**

**C D C D**

**N = 4 in both cases**

**E = 3 in both cases**

**Table for a) [1] Table for b) [1]**

A: [2] 🡪 B,C A:[2] 🡪 B,C

B:[2] 🡪 A,D B:[1] 🡪 D

C:[1] 🡪 A C:[0]

D:[1] 🡪B D:[0]

**3. For an adjacency list representation, G is the table:**

**\*Inter9\* If you have N vertices and E edges in your graph:**

* **how many array slots are needed in G?[1]** N
* **how many linked list nodes are required altogether in terms of E:**
  1. **for directed G?** E **Why?** Minimum number of edges in directed graph is N **[1]**
  2. **for undirected G?** E+1 **Why?** Minimum number of edges in undirected graph is N-1**[1]**

**Program: Graph Class Implementation (Adjacency Lists) [2+16=18pts] Your score:**

**Header:**

**Implementation:**

**Client:**

**Test results:**

**Total 16 points: Q’s 2 points:**

**Q) The state of the program statement [2pts]**

* **Does your program compile without errors?** Yes
* **List any bugs you are aware of, or state “No bugs”: No bugs**

**Make sure your HW3 files are perfect by running the slistTest.**

**dgraph.h, dgraph.cpp and hw6client.cpp must be used.**

Create a **directed graph** **class** which has the following data member:

**Gtable[20]**(an array) which contains the following (struct)in each slot:

**Struct Gvertex: (this is declared outside the class)**

* a vertex name (char)
* the mark/visit number (int)
* the out degree (int)
* a linked list object for adjacent vertices **(HW3 slist but with char element type)**

**And the following methods/member functions:**  **dgraph Constructor** - initializes the table entries

[ Make sure all names are initialized to be ‘ ‘ and visit number is 0] **dgraph Destructor** - destroys the table

|  |  |
| --- | --- |
| [ Does this call the list destructor automatically? If not, you have to destroy the lists. Test | |
| and see. **]** |  |

**displayGraph()** - displays the table content in a very readable table format

But make sure you do not display unused slots

**fillTable()** - reads the specified input file to fill the table Open and close the input file in here.

**int findOutDegree(char)** - returns the out degree of the vertex whose name is given as an argument.

**Must map the vertex name to the slot number.** A=0, B=1, etc.

usuing a formula.

May throw an exception if not found.

**slist findAdjacency(char)** - returns the linked list of adjacent vertices of the vertex whose name is given as an argument. **Must map the vertex name to the slot number**. A=0, B=1, etc.

usuing a formula.

May throw an exception if not found.

|  |  |
| --- | --- |
| [This one calls your HW3 copy constructor automatically because a list is being | |
| returned. Thus your HW3 copy construcor needs to be working.] |  |

**Note that the mark/visit number is not being used yet by the client. It will be used in the next HW.**

**Note that the linked list of adjacent vertices is of type slist and thus, you can use any of the slist member functions on it.**

**table.txt** **has the following format with vertices in the alphabetical order. Each line is name out-degree a-list-of-its-adjacent-vertices-separated-by-blanks e.g.**

**A 2 B F**

**I have provided you with the input file based on the notes.**

**You must use it to test your program thoroughly [test with vertices such as Z ]**  **Test.txt**

**Client program (hw6client.cpp) should: (Check hw6.out first but do not use contrlC for your program)**

1. **fillTable()**
2. **displayGraph()**
3. **LOOP until the user wants to stop (do not use control-C):**
   * 1. **the user will specify which vertex**
     2. **findOutDegree(char) findAdjacency(char)**
     3. **displays the returned results (hint: use HW3 function) And catches exceptions to display error messages but does not exit.**

**You must make HW6 is perfect because it will be used in HW7.**

**SUBMIT THESE 5 FILES: All files must be commented well!!**

**1. This assignment sheet with your answers**

# dgraph.h 3. dgraph.cpp 4. hw6client.cpp

**5. Test.txt – compilation and the results of thorough test cases (script) using my table.txt**