Texas Tech University Whitacre College of Engineering

CS 2413/5401 – Data Structures
Spring 2022
Lab Assignment 8

Acknowledge your collaborators or source of solutions, if any. **Online submission is required.**

While designing your programs or answering items, you are free to come up with your own assumptions based upon concepts and material learned in the course, if every potential specification is not given to you. Just be reasonable and document your assumptions. Such assumptions should not conflict with concepts and material learned in the course.

Your compliance with the "PROGRAMMING STYLE GUIDELINE" for CS 2413/5401 will affect your actual grade. All assignments will be checked for academic misconduct (cheating, plagiarism, collusion, falsifying academic records, misrepresenting facts, violations of published professional ethics/standards, and any act or attempted act designed to give unfair academic advantage to oneself or another student) defined by "OP 34.12: Grading Procedures, Including Academic Integrity" of TTU. Special software will be used to uncover such attempts.

A subset of answers submitted in this lab may be graded.

Objective: Practice graphs

Tasks:

1. You may work on this assignment by yourself, or you may work with one other student in this course as a team to complete this lab assignment.
   - **a.** Teams larger than 2 people will incur a 25% penalty off the total lab points per extra person.
   - **b.** It is expected that each team member contributes equitably and participates in coding and design ideas.
   - **c.** If a team member is dissatisfied with the performance of the other team member, you are allowed to dissolve the team and continue individually.
     - **i.** Whatever code each team member has contributed may be taken with that team member.
     - **ii.** Try not to let such a decision wait for the day the assignment is due as no extensions will be given if a team is dissolved.
2. Write a C program, problem1.c, to build the adjacency list for a directed and undirected graph (the program should handle both) from a graph file and run either a breadth-first search or a depth-first search on the graph.
   - **a.** Graph file format (you may presume the file is correct)
     - **i.** Number of vertices
     - **ii.** Flag for an undirected (0) or directed (1) graph
     - **iii.** Vertex character labels (consecutive letters starting with A and going through Z)
     - **iv.** List of edges and weights
     - **v.** Example
       1. 6
       2. 0
       3. A B C D E F
       4. A B 7
       5. A C 6
       6. A D 1
       7. B C 8
       8. C D 5
       9. C E 3
       10. D E 4

11. D F 5
12. E F 2

b. Build a dynamically allocated adjacency list to store the graph. Print the adjacency list out. Output should be similar to:

```
   i. Adjacency List:
  ii.   Number of Vertices:  6
 iii.   A:  (B,7) -> (C,6) -> (D,1) -> (nil)
  iv.   B:  (A,7) -> (C,8) -> (nil)
   v.   C:  (A,6) -> (B,8) -> (D,5) -> (E,3) -> (nil)
  vi.   D:  (A,1) -> (C,5) -> (E,4) -> (F,5) -> (nil)
 vii.   E:  (C,3) -> (D,4) -> (F,2) -> (nil)
viii.   F:  (D,5) -> (E,2) -> (nil)
```

c. Implement either the breadth first search or the depth first search from the chapter 13 lecture slides. Run the chosen search on at least two vertices. Output should be similar to:

```
   i. Breadth-First Search Results:
  ii.   Q: A B C D E F
 iii.   O:   A A A C D
  iv.
   v. Breadth-First Search Results:
  vi.   Q: C A B D E F
 vii.   O:   C C C C D
viii.
  ix. Depth-First Search Results:
   x.   V: A D F E C B
  xi.
 xii. Depth-First Search Results:
xiii.   V: C E F D B A
```

d. Hints:
   i. use a define macro constant at the top for the filename (the use of graph.txt would be appreciated)
   ii. utilize a head and rear pointer for each vertex list in the adjacency list

e. The main function should be a driver to call other functions to perform the required tasks.

f. No global variables should be used but define macro constants and typedef's may be used.

g. Report: In the comments at the end of the program, give the following information:
   i. Team Member Names
      1. For each team member, detail the work on the program concerning specific work, test cases, and code and functions designed, implemented, and modified, such as
         a. Name
            i. void build_adj (char filename[], adj_list_t *); - designed/implemented/modified
            ii. created graph file test cases
            iii. …
      2. A grade penalty of up to 25% of the total assignment points, which will be in addition to any other penalties, may be considered if inequitable contributions are made, not enough detail is present to be convincing, and/or all team members have the same list; i.e., all team members allegedly did the same thing (if both team members do the same thing – one team member is not needed).
   ii. Test Cases and Status
      1. Undirected graph file – passed/failed
      2. Directed graph file – passed/failed

3. …

iii. Graph Analysis

1. Big O of building the adjacency list in terms of number of vertices V and edges E
2. Big O of the storage requirements for the adjacency list
3. Big O of the storage needs for the breadth or depth first search as given in the chapter 13 lecture slides.

Learning Outcomes:

- Understand C program concepts, such as structs and pointers
- Understand how to implement graphs

Grading: 50 points

- Standard Deductions - 14 points
- Problem 1 – 36 points (problem1.c, graph test case text files)
    - Graph – 12 points
    - Report – 12 points
    - Test Cases – 12 points

Due Date:

4/1/2022, 11:59pm (submitted on Blackboard by ONE team member)