Assignment 6 – Surfin' U.S.A.

Nithin Duvvuru

CSE 13S – Winter 24

Purpose

Audience for this section: Pretend that you are working in industry, and write this paragraph for your boss. You are answering the basic question, "What does this thing do?". This section can be short. A single paragraph is okay.

Do not just copy the assignment PDF to complete this section, use your own words.

Questions

Please answer the following questions before you start coding. They will help guide you through the assignment. To make the grader's life easier, please do not remove the questions, and simply put your answers below the text of each question.

- What benefits do adjacency lists have? What about adjacency matrices?

  Adjacency lists are more memory efficient since they only store information about edges that exist and iterate faster over the neighboring vertexes. Adjacency matrices are better for dense graphs since they check whether an edge exists between two vertices constantly. But they do consume more memory especially for big graphs.

- Which one will you use. Why did we choose that (hint: you use both)?

  It is better to use a mix of both because adjacency lists can be used for small graphs and adjacency matrices for big graphs, balancing memory usage and time.

- If we have found a valid path, do we have to keep looking? Why or why not?

  We only have to continue if we are looking for all possible paths for finding the shortest path, if not we can stop.

- If we find 2 paths with the same weights, which one do we choose?

  I would choose depending on path length, simplicity, or efficiency.

- Is the path that is chosen deterministic? Why or why not?

  If the algorithm is deterministic, then the chosen path will also be deterministic.

- What type of graph does this assignment use? Describe it as best as you can.

  I think this assignment will use a weighted directed graph where each edge has a weight or cost.

- What constraints do the edge weights have (think about this one in context of Alissa)? How could we optimize our DFS further using some of the constraints we have?

  Edge weights in the context of Alissa represent distances or costs between locations. DFS optimization involves cutting through branches when the path weight exceeds a threshold and using techniques to avoid redundancy in the code.

Testing

List what you will do to test your code. Make sure this is comprehensive. [1] Be sure to test inputs with delays and a wide range of files/characters.

Test small input files with known paths and weights. Test edge cases like empty files and single nodes. Test larger input files. Test inputs with multiple valid paths. Test inputs with invalid paths.

How to Use the Program

Audience: Write this section for the user of your program. You are answering the basic question, "How do I use this thing?". Don't copy the assignment exactly; explain this in your own words. This section will be longer for a more complicated program and shorter for a less complicated program. You should show how to compile and run your program. You should also describe any optional flags or inputs that your program uses, and what they do.

To show "code font" text within a paragraph, you can use `\lstinline{}`, which will look like this: `text`.

For a code block, use `\begin{lstlisting}` and `\end{lstlisting}`, which will look like this:

```
Here is some code in lstlisting.
```

And if you want a box around the code text, then use `\begin{lstlisting}[frame=single]` and `\end{ lstlisting}`

which will look like this:

```
Here is some framed code (lstlisting) text.
```

Want to make a footnote? Here's how.[2]

Do you need to cite a reference? You do that by putting the reference in the file `bibtex.bib`, and then you cite your reference like this[1][2][3].

Run the program and provide an input file as the argument. Can use -i to read input file given as argument, and -o to write to output file given as argument. -d treats all graphs are directed. -h prints a help message.The program will output errors to stderr if any and output according to the used command line options.

Program Design

Audience: Write this section for someone who will maintain your program. In industry you maintain your own programs, and so your audience could be future you! List the main data structures and the main algorithms. You are answering the basic question, "How is this thing organized so that I can have a chance of fixing it?". This section will be longer for a more complicated program and shorter for a less complicated program.

[1] This question is a whole lot more vague than it has been the last few assignments. Continue to answer it with the same level of detail and thought.

[2] This is my footnote.

Main Data Structures: Graph (represented using adjacency lists or matrices), Stack (used for DFS), Path (represents path taken in the graph).

Main Algorithms: DFS (used to traverse graph), Pathfinding (finds the most optimal path between two vertices in graph), Graph traversing (algorithm to explore all vertices and edge in a graph), Error handling

Pseudocode

Give the reader a top down description of your code! How will you break it down? What features will your code have? How will you implement each function.

main function in tsp.c: Combines all pieces of code to finish task.
Graph.c/Graph.h: Provides function to create, modify, and access graph elements. Uses adjacency lists/matrices.
Stack.h/Stack.h: implements stack data structures. Provides functions like push, pop, and stack initialization. Used for DFS.
Path.c/Path.h: Defines path data structure. Stores information about path taken when traversing graph. Includes functions for path initialization, appending vertices, calculating path length, and determining if the path is valid.
Vertices.h: Specifies the starting vertex for the path.

Function Descriptions

For each function in your program, you will need to explain your thought process. This means doing the following

- The inputs of every function (even if it's not a parameter)
- The outputs of every function (even if it's not the return value)
- The purpose of each function, a brief description about a sentence long.
- For more complicated functions, include pseudocode that describes how the function works
- For more complicated functions, also include a description of your decision making process; why you chose to use any data structures or control flows that you did.

Do not simply use your code to describe this. This section should be readable to a person with little to no code knowledge. **DO NOT JUST PUT THE FUNCTION SIGNATURES HERE. MORE EXPLANATION IS REQUIRED.**

```
Function: main
Outputs: exit status and prints errors to stderr
Purpose: Calls necessary functions to find optimal path in graph.

GRAPH:

Function: add_edge
Inputs: Two vertices and the weight connecting them
Purpose: Adds an edge between two vertices with specified weight.

Function: remove_edge
Inputs: Two vertices
Purpose: Removes edge between given vertices

Function: get_neighbors
Inputs: a vertex
Outputs: List of neighboring vertices
```

```
Purpose: Retrieves neighboring vertices of given vertex

STACK:

Function: push
Inputs: an item
Purpose: Pushes item onto stack

Function: pop
Outputs: Item popped from stack
Purpose: Removes and returns the top item from stack

PATH:

Function: initalize_path
Inputs: Starting vertex
Purpose: Initializes a path from given vertex

Function: append_vertex
Inputs: Adds vertex to path
Purpose: Appends vertex to path

Function: calculate_path_length
Outputs: Length of path
Purpose: Calculates and returns length of path

Function: is_valid_path
Outputs: Boolean
Purpose: Checks if path is valid and returns a boolean based on its calculations
```

Results

Follow the instructions on the pdf to do this. In overleaf, you can drag an image straight into your source code to upload it. You can also look at `https://www.overleaf.com/learn/latex/Inserting_Images`

References

[1] Wikipedia contributors. C (programming language) — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/C_(programming_language), 2023. [Online; accessed 20-April-2023].

[2] Robert Mecklenburg. *Managing Projects with GNU Make, 3rd ed.* O'Reilly, Cambridge, Mass., 2005.

[3] Walter R. Tschinkel. Just scoring points. *The Chronicle of Higher Education*, 53(32):B13, 2007.