Assignment 5 – Surfin' USA

Collin Wen

CSE 13S – Fall 2023

Purpose

The purpose of this assignment is to create a program that is able to find the shortest path to go through multiple places. In this case Alissa is trying to find the most time efficient way to visit all the beaches mentioned in the song Surfin' USA starting at Santa Cruz and ending up back at Santa Cruz. To do this we create a program that finds the shortest path by checking all the possible paths.

How to Use the Program

To use this program the user must run the executable with ./tsp. The user also must pass the argument -i in the command line for the input file which should be a graph to read from. -o is the argument for the outfile which defaults to stdout. -d makes all graphs to be treated as directed, without -d all graphs are assumed to be undirected. -h prints out a help message.

An example of how this works would be user inputs into the command line:

```
./tsp -d -i maps/basic.graph
```

This runs the executable tsp with -d meaning the graph will be directed and -i choosing the file maps/basic.graph as the input file.

Say the file maps/basic.graph contains:

```
2
Home
The Beach
2
0 1 1
1 0 2
```

Then the output of the program would be printed to stdout as:

```
Alissa starts at:
Home
The beach
Home
Total Distance: 3
```

This output is the shortest possible path that Alissa can take to visit Home, The beach, and arrive back Home with the total distance the path takes.

This example is taken from asgn5.pdf.

Program Design

The design of this program is to use 3 different structs explained below and a main program "tsp" which uses the structs to print out the shortest path in a given graph file. The structs are used to record the different possible paths. In the main program a dfs is used to find the shortest path and print an output message with the shortest path given.

Data Structures

The 3 structs used in this program are,

Graph:

This struct is used to create an adjacency matrix with the different destinations as vertices and records the weights between destinations.

Its members are:

```
uint32_t vertices;
bool directed;
bool *visited;
char **names;
uint32_t **weights;
```

Stack:

This struct is just a Stack implementation with the members:

```
uint32_t capacity;
uint32_t top;
uint32_t *items;
```

It is used in the Struct paths to record the indexes of the different destinations.

Path:

This struct is used to record a path and the destinations on it as well as the total weight of the path.

Its members are:

```
typedef struct path {
uint32_t total_weight;
Stack *vertices;
} Path;
```

Algorithms

The algorithm used in this program is a depth first algorithm to find the shortest path of a given graph.

The pseudocode of this algorithm is:

```
def dfs(node n, graph g):
    mark n as visited
    for every one of n's edges:
        if (edge is not visited):
            dfs(edge, g)
        mark n as unvisited
```

Taken from asgn5.pdf.

This algorithm traverses through all the possible paths that can be created from a graph and finds the one with the shortest weight.