Asgn4 Design Doc

Elliott Jensen

October 2021

1 Introduction

Given a set of vertices this function finds the shortest Hamiltonian path. The user is able to specify if the graph is undirected or if the user wants the program to exhaustively print every Hamiltonian path

2 Test File: tsp.c

This file includes main and dfs (depth first search) and calls all of the functions of the program. It uses getopt to parse through the the command line and allows the user to specify an input and output file (stdin and stdout by default) and allows the user to decide if they want verbose printing, -v, or undirected graphs, -u.

I use getopt to set the input/output file and to set booleans for whether the program will run with verbose printing or undirected graphs. I use fscanf to set important variables like this:

```
fscanf(infile, "%d\n", &num_cities);
```

I use this method to create the an array with the cities names in it and to create the vertices for the graph. I create the vertices with the following for loop:

```
for i in num_cities:
    fscanf(infile,"%d %d %d", &i, &j, &k);
    graph_add_edge(i,j,k);
```

After generating all of the edges and variables I call dfs. In dfs I first add my current node to my path and check if my current node has an edge. I then recursively iterate through those edges as shown in the following loop:

```
graph_mark_visited()
path_push_vertex()
for edge in graph_vertices:
    if !graph_has_edge():
        continue
    if !graph_visited:
```

```
    dfs();
elif edge == 0 and graph_vertices == path_vertices:
    path_push_vertex()
    if path_length(curr) < path_length(shortest) || path_length(shortest == 0))
        path_copy(shortest,curr);
        path_print();
```

3 stack.c, path.c graph.c

These files each contain their respective structs and functions to interact with the struct. They are implemented exactly like they are specified in the asgn4 pdf, so I will not restate them here.