Assignment 4 DESIGN.pdf

Description of Program

This program calculates the quickest path passing through a 2d array of interconnected vertices and visits each vertice only one time while ending back at the original vertice. A graph function is used to store a matrix of values that denote the weight associated with each path connecting vertice to vertice. This program will attempt to utilize a recursive pathing method to find the quickest path with a given file input of number of vertices, name of vertices and a matrix of vertice and weight values.

graph.c: implements ADT to track the moves that vertices have made as well as create the graph matrix that stores the values given by the input file

tsp.c: main function that runs the Hamiltonian paths and takes in from the command line

-h Prints the help message

-v Enables printing of all the Hamiltonian paths found and the total number of recursive calls

-u specifies the graph to be undirected

-i infile: takes the input file name and prints out the path taken from city to city and edges of a graph

-o outfile specify the output file path to print to

stack.c: contains an ADT that implements a tracking method for path in order to retrace and store steps of path. The Stack will contain the index of the empty slot, capacity of the array and an array of items

Makefile:

• CC = clang must be specified.
• CFLAGS = -Wall -Wextra -Werror -Wpedantic must be specified.
• make must build the tsp executable, as should make all and make tsp.
• make clean must remove all files that are compiler generated.
• make format should format all your source code, including the header files.

README.md: It describes how to use your program and Makefile. It also lists and explains any command-line options that your program accepts.

DESIGN.pdf: Describes the program design as well and the design process in great detail. Provides pseudocode and notes describing the general functionality.

tsp.c source code

main()

Takes argument from command line using get opt

Switch (opt)

Case i prints argument and opens file with fopen to read mode ‘r’

Case o prints argument and open file with fopen to writing mode ‘w’

Character buffer 1024 spaces to get the first line

graph.c source code

Initialize struct Graph with unsigned vertices, undirected bool, veristed bool array Unsigned matrix [VERTICES][VERTICES]

Graph_create function takes in vertices and undirected
Initializes Graph*G and allocates memory to sizeof (Graph)
Sets vertices and undirected of G to parameters

graph _delete frees *G and sets it to NULL

graph_vertices returns vertices of the parameter Graph *G

Boolean graph_add_edge that returns if it is able to add weight parameter k to parameter i,j

graph_has_edge has parameter i,j of vertices and return of their are in bound of matrix

graph_edge_weight If either i or j aren’t within bounds, or if an edge doesn’t exist, return 0.

graph _visted returns if vertex v has been visited

graph_mark_visited sets vertex v as visited if in bound

graph_mark_unvisted if vertex is within bounds and is unvisted

graph _print

Prints out the matrix in order to debug code