1. **Problem Statement**

Create a program that can determine a Hamiltonian cycle of a graph using backtracking.

1. **Requirements**
   1. **Assumptions**

* Input File
  + Every input in a dataset is a non-negative integer
  + Items in a line are separated by a space
  + The input graph is simple, meaning there are no self-loops or parallel edges
  1. **Specifications**
* Must use a backtracking algorithm find the Hamiltonian cycle
* Output all data and actions to the screen and a file
  + Print the Hamiltonian cycle if one exists
    - If one doesn’t exist print that none exists

1. **Decomposition Diagram**

Main

Input file containing the graph datasets

Print all errors to the screen and to a file

Print all actions to the screen and to an output file

Determine if a Hamiltonian cycle exists using backtracking

Generate a graph of each dataset

Input

Output

Process

1. **Test Strategy**

* Valid Data  
  1.x
* File Handling  
  2.x

1. **Test Plan Version 1**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Strategy | Test Number | Description | Input | Expected Output | Actual Output | Pass/Fail |
| Data | 1.0 | Input file containing only one graph |  |  |  |  |
| Data | 1.1 | Input file containing more than one graph |  |  |  |  |
| Data | 1.2 | Graph that creates a Hamiltonian cycle |  |  |  |  |
| Data | 1.3 | Graph that does not create a Hamiltonian cycle |  |  |  |  |
| Data | 1.4 | Datasets that create a disjointed graph |  |  |  |  |
| File Handling | 2.0 | Use an input file that exists |  |  |  |  |
| File Handling | 2.1 | Use an input file that is empty |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

1. **Initial Algorithm**

* Graph class
  + Private attributes:
    - adjMatrix: 2-dimensional vector array representing a matrix
      * Public get and set access funcitons
  + Constructor:
    - Parameters:
      * Integer for the size of the graph
    - set the adjMatrix size to the input parameter amount
    - Set every value in adjMatrix equal to 0
  + addEdge function
    - Parameters:
      * Integer: v1
      * Integer: v2
    - If either of the two vertices are greater than the size of the adjMatrix
      * Print “[ERROR] One of the vertices in the edge do not exist, cannot create edge!”
    - Find the index of both vertices in the vertex vector (v1 and v2)
    - Set the float value of [v1][v2] in the adjMatrix equal to 1
    - Set the float value of [v2][v1] in the adjMatrix equal to 1
  + printGraph function:
    - if adjMatrix is not empty
      * for each row in the matrix
        + print each character in the row separated by 2 spaces
        + move to a new line
    - if the adjMatrix is empty
      * print “[ERROR] Graph is empty, cannot print!”
  + findHamiltonianCycle function:
    - Parameters: integer vector for the current Hamiltonian path, integer index for the current position in the vector
    - If index equals total vertex count
      * If the last vertex in the path vector has a path to the first vertex
        + Return true
      * Return false
    - For each vertex
      * If there is a path to the current indexed vertex and if the vertex is not already included in the path vector
        + Add the vertex to the path vector

Print “Adding vertex [v]”

* + - * + If findHamiltonianCycle returns true

Parameters: current path vector and current index + 1

Then Return true

* + - * + Else remove the most recent vertex from the path vector

Print “Removing vertex: [v]”

* + - Return false since no vertex can be added
  + hamiltonianCycle Function:
    - Parameters: none
    - Create a new vector for the path
    - set the first value of the path vector equal to 0 to indicate starting at the first vertex
    - call findHamiltonianCycle function and pass the path vector as a parameter
    - if findHamiltonianCycle returns false
      * print “Hamiltonian cycle does not exist”
    - else
      * print “Hamiltonian Cycle Exists: “
      * print the values of the path vector
* Main Function
  + Parameters: none
  + Ask the user to enter the name of the input file
  + Attempt to open the input file
    - If it does not exist
      * Print “[ERROR] File does not exist”
      * Return
    - If file is empty
      * Print “[ERROR] File is empty”
      * Return
  + Vector of graph objects
  + Integer for edge count equal to 0
  + for each line in the file
    - if edge count equals 0
      * create new graph with a size of the first integer on the line
      * set edge count equal to the second integer on the line
      * add the graph to the graph vector
    - else
      * add the edge to the current graph
        + first integer as vertex 1 and second integer as vertex 2
      * decrease edge count by 1
  + for each graph in the vector
    - call printGraph function
    - call hamiltonianCycle function

1. **Test Plan Version 2**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Strategy | Test Number | Description | Input | Expected Output | Actual Output | Pass/Fail |
| Data | 1.0 | Input file containing only one graph | Graphs1.txt | Reads the dataset for the graph and displays the Hamiltonian cycle |  |  |
| Data | 1.1 | Input file containing more than one graph | Graphs2.txt | Successfully reads in each dataset and generates a graph for each |  |  |
| Data | 1.2 | Graph that creates a Hamiltonian cycle | Graphs2.txt  Graph 1 | Cycle: 1,2,4,3 |  |  |
| Data | 1.3 | Graph that does not create a Hamiltonian cycle | Graphs2.txt  Graph 3 | No possible cycle |  |  |
| Data | 1.4 | Datasets that create a disjointed graph of with several vertices | Graphs2.txt  Graph 5 | No possible cycle |  |  |
| Data | 1.5 | Dataset that contains edge vertex that do not exist | Graphs2.txt  Graph 7 | Prints an error for non-existent edges and generates cycle of 1,2,3,4 |  |  |
| Data | 1.6 | Dataset that contains no edges | Graphs2.txt  Graph 8 | No possible cycle |  |  |
| Data | 1.7 | Graph that does not create a Hamiltonian cycle | Graphs2.txt  Graph 9 | No possible cycle |  |  |
| Data | 1.8 | Datasets that create a disjointed graph with only one disjointed vertex | Graphs2.txt  Graph 6 | No possible cycle |  |  |
| Data | 1.9 | Graph that creates a Hamiltonian cycle | Graphs2.txt  graph 2 | Cycle: 1,2,3,4,5,6 |  |  |
| Data | 1.10 | Graph that creates a Hamiltonian cycle | Graphs2.txt  graph 4 | Cycle: 1,4,3,7,5,2,6,8 |  |  |
| File Handling | 2.0 | Use an input file that exists | Graph1.txt  Graph2.txt | Successfully opens each file |  |  |
| File Handling | 2.1 | Use an input file that is empty | Empty.txt | Prints “[ERROR] file is empty” |  |  |
| File Handling | 2.2 | Use an input file that does not exist | Somefile.txt | Prints “[ERROR] file does not exist |  |  |

1. **Code**

A baseline for commenting is before any function add this:

//Description: What does the function do

//Pre-condition: What do input do you need for the function to work

//Post-condition: What is the end result of the function or what do you get out of the function

Also the beginning of your program should have these comments:

//Program Name:

//Programmer Name:

//Description:

//Date Created:

1. **Updated Algorithm**

* Graph class
  + Private attributes:
    - adjMatrix: 2-dimensional vector array representing a matrix
      * Public get and set access funcitons
  + Constructor:
    - Parameters:
      * Integer for the size of the graph
    - set the adjMatrix size to the input parameter amount
    - Set every value in adjMatrix equal to 0
  + addEdge function
    - Parameters:
      * Integer: v1
      * Integer: v2
    - If either of the two vertices are greater than the size of the adjMatrix
      * Print “[ERROR] One of the vertices in the edge do not exist, cannot create edge!”
    - Find the index of both vertices in the vertex vector (v1 and v2)
    - Set the float value of [v1][v2] in the adjMatrix equal to 1
    - Set the float value of [v2][v1] in the adjMatrix equal to 1
  + printGraph function:
    - if adjMatrix is not empty
      * for each row in the matrix
        + print each character in the row separated by 2 spaces
        + move to a new line
    - if the adjMatrix is empty
      * print “[ERROR] Graph is empty, cannot print!”
  + findHamiltonianCycle function:
    - Parameters: integer vector for the current Hamiltonian path, integer index for the current position in the vector
    - If index equals total vertex count
      * If the last vertex in the path vector has a path to the first vertex
        + Return true
      * Return false
    - For each vertex
      * If there is a path to the current indexed vertex and if the vertex is not already included in the path vector
        + Add the vertex to the path vector

Print “Adding vertex [v]”

* + - * + If findHamiltonianCycle returns true

Parameters: current path vector and current index + 1

Then Return true

* + - * + Else remove the most recent vertex from the path vector

Print “Removing vertex: [v]”

* + - Return false since no vertex can be added
  + hamiltonianCycle Function:
    - Parameters: none
    - Create a new vector for the path
    - set the first value of the path vector equal to 0 to indicate starting at the first vertex
    - call findHamiltonianCycle function and pass the path vector as a parameter
    - if findHamiltonianCycle returns false
      * print “Hamiltonian cycle does not exist”
    - else
      * print “Hamiltonian Cycle Exists: “
      * print the values of the path vector
* Main Function
  + Parameters: none
  + Ask the user to enter the name of the input file
  + Attempt to open the input file
    - If it does not exist
      * Print “[ERROR] File does not exist”
      * Return
    - If file is empty
      * Print “[ERROR] File is empty”
      * Return
  + Vector of graph objects
  + Integer for edge count equal to 0
  + for each line in the file
    - if edge count equals 0
      * create new graph with a size of the first integer on the line
      * set edge count equal to the second integer on the line
      * add the graph to the graph vector
    - else
      * add the edge to the current graph
        + first integer as vertex 1 and second integer as vertex 2
      * decrease edge count by 1
  + for each graph in the vector
    - call printGraph function
    - call hamiltonianCycle function

1. **Test Plan Version 3**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Strategy | Test Number | Description | Input | Expected Output | Actual Output | Pass/Fail |
| Data | 1.0 | Input file containing only one graph | Graphs1.txt | Reads the dataset for the graph and displays the Hamiltonian cycle |  |  |
| Data | 1.1 | Input file containing more than one graph | Graphs2.txt | Successfully reads in each dataset and generates a graph for each |  |  |
| Data | 1.2 | Graph that creates a Hamiltonian cycle | Graphs2.txt  Graph 1 | Cycle: 1,2,4,3 |  |  |
| Data | 1.3 | Graph that does not create a Hamiltonian cycle | Graphs2.txt  Graph 3 | No possible cycle |  |  |
| Data | 1.4 | Datasets that create a disjointed graph of with several vertices | Graphs2.txt  Graph 5 | No possible cycle |  |  |
| Data | 1.5 | Dataset that contains edge vertex that do not exist | Graphs2.txt  Graph 7 | Prints an error for non-existent edges and generates cycle of 1,2,3,4 |  |  |
| Data | 1.6 | Dataset that contains no edges | Graphs2.txt  Graph 8 | No possible cycle |  |  |
| Data | 1.7 | Graph that does not create a Hamiltonian cycle | Graphs2.txt  Graph 9 | No possible cycle |  |  |
| Data | 1.8 | Datasets that create a disjointed graph with only one disjointed vertex | Graphs2.txt  Graph 6 | No possible cycle |  |  |
| Data | 1.9 | Graph that creates a Hamiltonian cycle | Graphs2.txt  graph 2 | Cycle: 1,2,3,4,5,6 |  |  |
| Data | 1.10 | Graph that creates a Hamiltonian cycle | Graphs2.txt  graph 4 | Cycle: 1,4,3,7,5,2,6,8 |  |  |
| File Handling | 2.0 | Use an input file that exists | Graph1.txt  Graph2.txt | Successfully opens each file |  |  |
| File Handling | 2.1 | Use an input file that is empty | Empty.txt | Prints “[ERROR] file is empty” |  |  |
| File Handling | 2.2 | Use an input file that does not exist | Somefile.txt | Prints “[ERROR] file does not exist |  |  |

1. **Screenshots**
2. **Error Log**

|  |  |  |
| --- | --- | --- |
| Error Type | Cause of Error | Solution to Error |
|  |  |  |
|  |  |  |

1. **Status**