HACETTEPE UNIVERSITY COMPUTER SCİENCE ENGINEERING DEPARTMENT

BIL-235 LABORATORY

4. EXPERIMENT REPORT

Name :Serdar
Surname :GÜL
Number :20421689
Subject :Graph,Stack,Shortest Path Algorithm
Language :JAVA
System :Windows
Platform :Eclipse 3.4.1. Ganymede and JRE 1.6
Advisors :Doc. Dr. Mustafa EGE, R.A. Önder KESKİN
Due Date :28.11.2008

SOFTWARE USAGE

It is used so easily.You can run this program by the command prompt or in the Eclipse platform of the programming.

Program takes 4 files for the arguments and it establish an output file which shows the results of the operations in the other files.

Our first argument is a file that gives the distances between the nodes.In this experiment we will use the “distances.txt”

Our second argument is a file that contains the packages of the nodes in our graph.In this experiment we will use the “package_list.txt”

Our third argument is a file that contains the vehicles’ informations and their tasks to accomplish.Vehicle is used to carry packages from one node to the target node by using the shortest path algorithm.I will explain them detailed in the innes sections of the report.In this experiment we will use “task_lists.txt” as the third parameter of the program.

And finally our fourth argument is a file that contains the commands which will be executed by the simulation program I made.In this experiment I will use “commands.txt” as the fourth parameter of the program.

At last program creates an automatic “output.txt” file which contains the results of the commands which are read and executed from the 4th parameter of the program.

And these files must be in the Project which we will design the code in it.

ERROR MESSAGES

“This is not a suitable command for this simulation”:In the command file , if there is a command that is not related with the experiment.User will take this error message.

SOFTWARE DESIGN NOTES

PROBLEM

In this experiment our main problem is using the stack data structure , designing a graph data structure and creating a shortest path algorithm.And the hardest part of the experiment is combining these in an experiment.

In the earlier steps of the program we can see these problems
-User can give more parameters
-User can give less parameters

-User can give the parameters with the wrong order
-User can give a file as a parameter which is not in the Project folder.

In the deeper parts of the experiment we can see these as a problem
-Using stack and graph combinely can be hard
-There will be too much controls to cope with this experiment.
-Creating a shortest path algorithm can be so hard and it can be failed sometimes.
-Printing the results to the output file can give us some difficulties.
-Because of the design of the simulation program there will be a lot of vectors and the other data structures.Control all of them and check that ,if they are complete true is so hard..

And finally the program may not give totally true results and may not write results to an output file sometimes for some commands..

SOLUTION

I read the first file “distances.txt” and I initialize the nodes and their edges..There will be a problem that our matrix data in the file must be at leat 4*4 and at must 10*10.If we will take an input file which is not in these boundaries we will not make operations about this input file.

After making the nodes and initialize their values I read the second file “package_list.txt” which contains the information of the packages the node will have.And in this input file the line’s first element is the number of the packages node will have information.When I read the number element I will just read that number element in the line , not more.

For example if my line will be like this

6 AB AC AD AE AF AT AY AH AJ AK

I will just read the

AB AC AD AE AF AT and I will send them to the node’s Package element.It will be a stack.Stack is a data structure that runs with the FIFO(First In First Out) algorithm.

After this I read the third file “task_lists.txt” and I will create the vehicle objects and I will initialize the elements.Also that I will read the tasks which the vehicles will have to do this and initialize their values too.

The operations until the reading of the command file(“command.txt”-> 4th argument) are the operations of the initializing and the creation of the elements which are strictly necessary to us to cope with this experiment.

And finally I read the fourth argument of the program “command.txt” and I will read the command.

Some of the commands take 2 parameters , and some of them take 3 parameters.I check the first argument of the input file’s line and after this I will take the parameters.

For example if the command is “get_node_graph” I will take a second parameter as the line number of the matrix.And after this I will print the whole egdes of the graph with no reputation..

Or if the command is “print_shortest_path” I will take 2 parameters that first is the source node and the second is the target node.With the shortest path algorithm I will print the shortest path from source node the the target node...

If the command is “print_node_package_lists” I will take a parameter that is the run time of the simulation. I have a function that named processAll(this returns the final form of the vehicle list) and processAll2(thie returns the final form of the node list).It runs the simulation by the given time and returns the nodes and the vehicles with their last form.

And after this I print the packages of the nodes at the given time.

Else if the command is “print_vehicle_package_lists” I will take a parameter that is the run time of the simulation.I have a function that named processAll(this returns the final form of the vehicle list) and processAll2(thie returns the final form of the node list).It runs the simulation by the given time and returns the nodes and the vehicles with their last form.

And after this I print the packages of the vehicles at the given time.

Or if the command is “print_vehicle_state” I will take 2 parameters.First one is the name of the vehicle and the second is the time of the state.So you can understand that I will call the processAll and processAll2 again to do this.And I will send the values of the state of the vehicle to the output file.

I will do this operations until the command input file will be ended and I will print the results to the “output.txt”

And also that I used the data structures which are enforced us to do this.These are like graph , stack etc.

But I use some of the data structures which are so useful to cope with this experiment like vectors etc.I will explain them in the MAIN DATA STRUCTURES part of the report detailed.

And I prefer a multi – classed solution fort his experiment.I gave a priority to make thie experiment with the object oriented programming.And I didn’t want to fulfil all the read ,write , algorithm operations in the main class (in the main function).So I defined an Operation class and I made all of the operations in them.As a result of this my main function and main class is so tidy and understandable.

SYSTEM CHARTS

INPUT	PROGRAM	OUTPUT
Distances.txt *	java source codes	output.txt
Package_list.txt *	Main.java	
Task_list.txt *		
Commands.txt *		

(*)->these are the prototypes of the files.First argument must give the information about the nodes and their distances between tehmselves.Second argument must give the package information of the nodes.Third file must give the information of the vehicles' and their tasks.And finally the fourth file will give information that will have the commands which will be executed by the simulation program I made.So it can take different names.I just give examples for this experiment.You can give a file that's name will be "onder.txt" but it is about the distances..

MAIN DATA STRUCTURES

I use lots of classes and data structures to cope with this experiment

At first I made a Node class that has the information of the node name , node length (it is very important to make a shortest path algorithm) ,

the edges list and the neighbor node list.And for the package operations it has a stack of the packages.

My Node class will be like this.

```
public class Node implements Comparable<Node>{

    public char NodeName ;
    public Vector<Edge> EdgeList ;
    public Vector<Node> Nodelist ;
    public Stack<Package> PackageStack ;
    public Node previous ;
    public int NodeLength = Integer.MAX_VALUE;

    public Node(char nodeName, Vector<Edge> edgeList,
Vector<Node> nodelist,
            Stack<Package> packageStack, int
nodeLength, int nodeNumber) {
        super();
        NodeName = nodeName;
        EdgeList = edgeList;
        Nodelist = nodelist;
        PackageStack = packageStack;
        NodeLength = nodeLength;
        NodeNumber = nodeNumber;
    }
}
```

And also that all of my classes have the getter and the setter methods.

I have an Edge class that has a name , source node , destination node and the edge distance informations and my Edge class will be like this.

```
    public class Edge {

        public String EdgeName ;
        public Node SourceNode ;
```

```
        public Node DestinationNode ;
        public int EdgeDistance ;

        public Edge(Node sourceNode, Node
destinationNode, int edgeDistance) {
            super();
            SourceNode = sourceNode;
            DestinationNode = destinationNode;
            EdgeDistance = edgeDistance;
        }
    }
```

I have a Package class that has just name information.

```
    public class Package {

        public String PackageName ;

        public Package(String packageName) {
            super();
            PackageName = packageName;
        }
    }
```

I have a Vehicle Class that has he informations of the numbers, a stack of the packages , a list of tasks , a list of the states acording to the execution of the simulation by a given time.

And my Vehicle class will be like this.

```
    public class Vehicle {

        public String VehicleName ;
        public Vector<State> StateList ;
        public Stack<Package> VehiclePackages ;
        public Vector <Task> TaskList ;
```

```
        public Vector <Node> VehiclePath ;



        public Vehicle(String vehicleName,
Vector<State> stateList,
                Stack<Package> vehiclePackages,
Vector<Task> taskList) {
            super();
            VehicleName = vehicleName;
            StateList = stateList;
            VehiclePackages = vehiclePackages;
            TaskList = taskList;
        }
    }
```

I have a State class that has the information of the state of the vehicle.Vehicle can be in an edge or in a node.Because of this , my state class has 2 constructor methods.And also that it has the information of the distance value..

```
    public class State {

        public Node StateNode ;
        public Edge StateEdge ;
        public int StateDistance ;

        public State(Edge stateEdge, int
stateDistance) {
            StateEdge = stateEdge;
            StateDistance = stateDistance;
        }

        public State(Node stateNode, int
stateDistance) {
            StateNode = stateNode;
            StateDistance = stateDistance;
        }
    }
```

I have a Task Class that has the information of the beginning node , the ending node and the number of the packages.With this class vehicle takes the given number of the packages by the given beginning nde and send them to the given ending node by using the shortest path algorithm.

My task class will be like this..

```
public class Task {

    public String Source ;
    public String Destination ;
    public Vector<Node> shortestPath ;
    public int PackageNumber ;

    public Task(String source, String
destination, Vector<Node> shortestPath,
            int packageNumber) {
        super();
        Source = source;
        Destination = destination;
        this.shortestPath = shortestPath;
        PackageNumber = packageNumber;
    }
}
```

I have a main class that just makes the argument control , creating an output file and sending the functions in the operation class functions to cope with the experiment.

And I have an Operation class that has an empty constructor and the whole input reading , output writing and the algorithm operations.

Also that I use some of the variables that are integer , string , vector , etc..

For example we can give a list of the variables which are used in the function of the getting graph info function.

```
int i = 0 , counter , decision;
Edge temp = new
Edge(NodeList.elementAt(i).EdgeList.elementAt(counter
).DestinationNode ,
NodeList.elementAt(i).EdgeList.elementAt(counter).Sou
rceNode ,
NodeList.elementAt(i).EdgeList.elementAt(counter).Edg
eDistance);
```

ALGORITHM

Take the command from the command line
If the argument number is more than the program want
    Give an error message and terminate the program
Else if the number of the arguments is less than the program want
    Give an error message and terminate the program
Else
    Create an output file named “output.txt” to print the results
    Take the first parameter as a distance file.
    Go to the readDistanceFile() function of the operation class
    Read the file line by line
    If the tokenizing elements number is not between 3 and 10

Give an error message and quit from the readdistancefile() function

Else

Read the values and if the value is not -1 or 0 it will be an edge.

Make this operations until the input file will be ended.(It initializes the nodes and their edges.)

Take the second parameter as a package file

Go to the readPackageFile() function of the operation class

Read the file line by line

Take the line’s first element as the number of the packages.

Read that number of the packages from the line and send them to the package stack of the node

Make that operations until the file will be ended.

Take the third parameter as the tasks file

Go to the readTaskFile() function of the Operations class

Read the first line

Take first line’s first element as a vehicle’s name and second is the number of the tasks

Read the task number of lines and take them as a task.line’s first item is beginning node , second is ending node and the last one is the package number which is taken from the source and send to the destination..

Make this operations until the task file will be ended

Take the fourth parameter as the commands file

Go to the readCommandFile() function of the Operations class

Open the file and read a line

Read the first element of the line

İf element is “get_graph_info”
Take the second parameter as a number
Go to the getGraphInfo() function of the Operations class (it prints the result to the output file)

Else if the element is “print_shortest_path”
Take the second parameter as the source node and take the third parameter as the destination node
Go to the printShortestPath() function of the Operation class(it goes to the related functions and prints the results to the output file)

Else if the element is “print_vehicle_package_lists”
Take the second parameter as the simulation run time
Go to the printVehiclePackages() function of the Operation class(it goes to the related functions and print the results to the output file.)

Else if the element is “print_node_package_lists”
Take the second parameter as the simulation run time
Go to the printNodePackages() function of the Operations class(it goes to the related functions and finally prints the result to the output file)

Else if the element is “print_vehicle_state”
Take the second parameter as the name of the vehicle and the third parameter as the simulation run time.
Go to the printVehicleState() function of the Operations class(it goes to the related functions and finally prints the resl-ult to the output file).

Else
Give an error message that is not a suitable command for this experiment.

Read the lines until the command file will be ended.

EXECUTION FLOW BETWEEN SUBPROGRAMS

There are some functions that are used combi nely.These are;

PrintVehiclePackageLists(NodeList,VehicleList,rea ltime,output); : This is used to print the packages of the vehicles to the output file with the given run time.It Works combinely with these functions
public Vector<Vehicle> processAll(Vector<Node> nodeList, Vector<Vehicle> vehicleList,int realtime) : it is used to process the nodes and the vehicles
private void initialize(Vector<Node> nodeList, Vector<Vehicle> vehicleList , int initialize , int element) : it is used to give the initial values to the temp lists
public Vector<Node> organize(Vector<Node> nodeList, Vector<Vehicle> vehicleList , int counter3 , int counter4) : it is used combinely with initialize and it is used to find the shortest path

public void PrintNodePackageLists(Vector<Node> nodeList, Vector<Vehicle> vehicleList, int realtime,

BufferedWriter output) throws IOException , EmptyStackException：it is used to print the packages of the nodes to the output file with the given run time..it Works with the same functions but processAll2 besides processAll function

public Vector<Node> processAll2(Vector<Node> nodeList, Vector<Vehicle> vehicleList,int realtime) : it returns a node list to print the packages of the nodes..

public void GetGraphInfo(Vector<Node> NodeList,Vector<Vehicle> vehicleList, int real, BufferedWriter output) throws IOException : it is used top rint the whole edges of the graph.

public void ReadCommandFiles(String FileName,Vector<Node> NodeList,Vector<Vehicle> VehicleList,BufferedWriter output) throws IOException , FileNotFoundException : it is used to read the command file and maket he operations

public Vector <Vehicle> ReadTaskListFile(String FileName) throws IOException , FileNotFoundException : it is used to read the task file and initialize the vehicles and the tasks

public void ReadPackageListFile(String FileName , Vector<Node> NodeList) throws IOException , FileNotFoundException : it is used to read the package information file and initialize the node’s packages values.

public Vector<Node> ReadDistancesFile(String FileName) throws IOException , FileNotFoundException : it is used to read the distances file and initialize the edges and the nodes of the graph.

public void PrintShortestPath(String source, String dest , Vector<Node> NodeList , BufferedWriter output) throws IOException : it is used top rint the shortest path from the source node to the destination node with giving the whole distance of the edges.

A couple of examples

```
    public void GetGraphInfo(Vector<Node> NodeList,Vector<Vehicle> vehicleList, int real, BufferedWriter output) throws IOException
    {
        int i = 0 , counter , decision;
        output.write("get_graph_info " + real);
        output.newLine();
        for(i = 0 ; i < NodeList.size() ; i++)
        {
            for(counter = 0 ; counter < NodeList.elementAt(i).EdgeList.size() ; counter++)
            {
                Edge temp = new Edge(NodeList.elementAt(i).EdgeList.elementAt(counter).DestinationNode , NodeList.elementAt(i).EdgeList.elementAt(counter).SourceNode , NodeList.elementAt(i).EdgeList.elementAt(counter).EdgeDistance);

    //output.write(temp.getSourceNode().getNodeName() + " " + temp.getDestinationNode().getNodeName() + " " + temp.getEdgeDistance());
                //output.newLine() ;
                decision = Search(temp,NodeList,i);
                if(decision == 1)
                {

                }
```

```
                else
                {

    output.write(NodeList.elementAt(i).EdgeList.eleme
ntAt(counter).SourceNode.getNodeName() + " " +
NodeList.elementAt(i).EdgeList.elementAt(counter).Des
tinationNode.getNodeName() + " " +
NodeList.elementAt(i).EdgeList.elementAt(counter).get
EdgeDistance());
                    output.newLine();
                }
            }
        }


    }//this function is used to give the edges of the
graph informations



public int Search(Edge temp, Vector<Node> nodeList,
int i) {
        int counter , counter2 ;
        int control = 0 ;
        for(counter = 0 ; counter < i ; counter++)
        {
            for(counter2 = 0 ; counter2 <
nodeList.elementAt(counter).EdgeList.size() ;
counter2++)
            {

    if(nodeList.elementAt(counter).EdgeList.elementAt
(counter2).getSourceNode().equals(temp.getSourceNode(
)) )
                {

    if(nodeList.elementAt(counter).EdgeList.elementAt
(counter2).getDestinationNode().equals(temp.getDestin
ationNode()) )
                    {

    if(nodeList.elementAt(counter).EdgeList.elementAt
```

```
(counter2).getEdgeDistance() ==
temp.getEdgeDistance())
                            {
                                control = 1 ;
                                break ;
                            }
                        }

                    }
                }
            }

            return control ;
        }// this function is used to search an edge in
graph if that is defined before...




public void ReadCommandFiles(String
FileName,Vector<Node> NodeList,Vector<Vehicle>
VehicleList,BufferedWriter output) throws IOException
, FileNotFoundException
    {
        String line , number , command , source ,
dest , vehicle , time = new String();
        int real , realtime;
        BufferedReader input = new BufferedReader(new
FileReader(FileName));
        while((line = input.readLine()) != null)
        {
            StringTokenizer token = new
StringTokenizer(line);
            command = token.nextToken() ;
            if(command.equals("get_graph_info"))
            {
                number = token.nextToken();
                real = Integer.parseInt(number);

    GetGraphInfo(NodeList,VehicleList,real,output);
            }//if the command is about getting the
information of the graph
```

```
            else
if(command.equals("print_shortest_path"))
            {
                source = token.nextToken() ;
                dest = token.nextToken() ;

    PrintShortestPath(source,dest,NodeList,output);
            }//if the command is printing the
shortest path by the source to the destination

            else
if(command.equals("print_vehicle_package_lists"))
            {
                time = token.nextToken() ;
                realtime = Integer.parseInt(time);

    PrintVehiclePackageLists(NodeList,VehicleList,rea
ltime,output);
            }//if the command is printing the
packages of the vehicles

            else
if(command.equals("print_node_package_lists"))
            {
                time = token.nextToken() ;
                realtime = Integer.parseInt(time);

    //PrintNodePackageLists(NodeList,VehicleList,real
time,output);
            }//if the command is printing the
packages of the nodes

            else
if(command.equals("print_vehicle_state"))
            {
                vehicle = token.nextToken() ;
                time = token.nextToken() ;
                realtime = Integer.parseInt(time);

    //PrintVehicleState(VehicleList,NodeList,vehicle,
realtime,output);
            }//if the command is printing the state
of the vehicle
```

```
            else
            {
                output.write("This is not a suitable
command for this simulation.");
                output.newLine();
            }//if the command is not in a suitable
format for that experiment

        }//end of while
    }//end of the reading commands from the command
file
```

SOFTWARE TESTING NOTES

SOFTWARE RELIABILITY

It is a good program that has too little bugs.If you use the third option so much and after this you run the fourth option program runs collectly.But in some commands it may not run excellently.I give you guarantee that graph_info and shortest_path algorithm runs correctly.But the other commands may give exception.Because of this I take them as an

explanation part of the program.But you can take a look at in the source code.

And if user enters a wrong number program controls this and prints the menu again but it cleasr the screen so you may not see the error messages…

Except this , this program runs correctly and it has lots of controls that user can give wrong expression.
It controls these events.
-User can give less parameters to the command line while executing the program.(This program will run in 3 arguments for the files and 4 arguments for the commands which are written to the command line)
-User can give more parameters.(User can give unnecessary parameters to the command line..)
-User can give a wrong input or output file..

SOFTWARE EXTENDIBILITY AND UPRADIBILITY

This program can be extended with some small changes and we can use it with the operations we can cope with another data structures like vectors , arraylists and we can make our own library to this programming language.

And with this changes we will make an excellent processor and we can make a task manager from this source code..

And we can give a visual form with using the graphical interface of the JAVA GUI Object oriented programming language.
If the graphical interface will be good , a lot of people will use this program.

And we can change the subject of the program and we can make a lot of programs with a huge variety..

PERFORMANCE CONSİDERATIONS

This program runs rapidly and with some small changes we can make it faster then all.
And it gives the true results which are expected from us in a very little time.

COMMENTS

I think that this experiment will be so useful for us now and after these days.
For example I learn JAVA language again and I don’t forget it after this experiment.
And it was so useful to us to finding great ideas for the programming and it causes us to study much more than.
Briefly it was a so useful and good experiment.
And I take the Bil-341 course,because of this , program gives false results in sometimes rarely .So I apologise from you.And while you were giving the not for the experiment if you think about this I will be so appreciated to you..
Regards..

RESOURCES

Thinking in JAVA 2rd Edition by BRUCE ECKEL
How To Learn JAVA by DEITEL
Data Structures and Algorithms in JAVA

These are the e-books I used for dealing with this experiment.

Web Sites

www.javasun.com
some of the forum sites
www.programlama.com