Assignment 4 Report

Problem statement:

This assignment involves the design of a software to control a system of traffic lights at an intersection. An intersection consists of two streets that cross at right angles. For simplicity, each street has a single lane in each direction (no lanes designated for left turn). The software controls the 4 traffic lights (2 direction x 2 roads) as shown in the figure below.



Gathering info:

The green timing for each traffic light is proportional to the traffic flow rate reported

for the same road, according to the following equation:

Where is the green time for the ith traffic light, represents the traffic flow (number of vehicles per

hour) crossing the ith traffic light, represents the total traffic flow passing through the intersection, and

𝐶 represents the cycle length in seconds.

The system has the following components:

1. Traffic semaphores (signal lights): these are standard semaphores with three lights: red, yellow,

and green.

2. Traffic sensors that are embedded in each lane near the intersection to record the traffic flow for all

roads (4 sensors generating 4 traffic rate values when four traffic lights are used). The sensors save

the traffic rate information into a file (average number of vehicles per hour passing through a

particular road in one direction).

3. The signals operate in a conventional fashion. Traffic is allowed to move on one road, say R1, and

then the next (R2), alternatively across the four roads of the intersection. Assume that the four

traffic lights are represented as L1, L2, L3, and L4. The system operates as follows:

a. Traffic light (L1) is green for a duration calculated based on the traffic flow rate in road

R1, the other traffic lights (L2, L3, and L4) are red.

b. L1 becomes yellow for X seconds (X being a constant value). The Department of

Transportation's traffic manual recommends that yellow lights are between 3 and 6 seconds

long. Other traffic lights (L2, L3, and L4) remain in red state.

c. Then, traffic light L2 becomes green for a duration calculated based on the traffic flow rate

in road R2. Meanwhile, L1, L3, and L4 are red.

d. Traffic light L2 becomes yellow for X seconds (X being a constant value). Other traffic

lights (L1, L3, and L4) remain in red state.

e. Then, traffic light L3 becomes green for a duration calculated based on the traffic flow rate

in road R3. Meanwhile, traffic lights L1, L2, and L4 are red.

f. Traffic light L3 becomes yellow for X seconds (X being a constant value). Other traffic

lights (L1, L2, and L4) remain in red state.

g. Then, traffic light L4 becomes green for a duration calculated based on the traffic flow rate

in road R4. Meanwhile, traffic lights L1, L2, and L3 are red.

h. Traffic light L4 becomes yellow for X seconds (X being a constant value). Other traffic

lights (L1, L2, and L3) remain in red state.

i. The next cycle starts with traffic light L1 becoming green again, and so on.

4. The green timings for the traffic lights are updated regularly based on traffic flow. The program

assumes that the traffic information is stored in a file (cycle time and traffic flow rates). Every

specific duration (say 24 hours), the data is read from the file, the green timings are updated based

on the latest traffic condition, and the control proceeds with the updated green timings.

Input/Output:

Output: Simulation of the traffic intersection

Calculating green light times…

ID: 1

State: Active

Green light time: 9.6 seconds

ID: 2

State: Active

Green light time: 14.4 seconds

ID: 3

State: Active

Green light time: 19.2 seconds

ID: 4

State: Active

Green light time: 16.8 seconds

T0: green

T1: red

T2: red

T3: red

\*Green light time delay of 9.6s\*

T0: yellow

T1: red

T2: red

T3: red

(keeps going)

Test Cases:

Case 1: Testing the simulation and the printLightInfo function

Cycle time = 60s

T1 flowrate = 20 T1 greenlight= 9.6s

T2 flowrate = 30 T2 greenlight= 14.4s

T3 flowrate = 40 T3 greenlight= 19.2s

T4 flowrate = 35 T4 greenlight= 16.8s

Total flow = 125

Calculating green light times…

ID: 1

State: Active

Green light time: 9.6 seconds

ID: 2

State: Active

Green light time: 14.4 seconds

ID: 3

State: Active

Green light time: 19.2 seconds

ID: 4

State: Active

Green light time: 16.8 seconds

T1: green

T2: red

T3: red

T4: red

\*9.6 second delay\*

T1: yellow

T2: red

T3: red

T4: red

\*6 second delay\*

T1: red

T2: green

T3: red

T4: red

\*14.4 second delay\*

T1: red

T2: yellow

T3: red

T4: red

\*6 second delay\*

T1: red

T2: red

T3: green

T4: red

\*19.2 second delay\*

T1: red

T2: red

T3: yellow

T4: red

\*6 second delay\*

T1: red

T2: red

T3: red

T4: green

\*16.8 second delay\*

T1: red

T2: red

T3: red

T4: yellow

\*6 second delay\*

(keeps looping)

Case 2: Simulation then update green times after a 5 minute interval (For testing purposes, usually the interval would be 24 hours)

Initial flowrates:

Cycle time = 60s

T1 flowrate = 20 T1 greenlight= 9.6s

T2 flowrate = 30 T2 greenlight= 14.4s

T3 flowrate = 40 T3 greenlight= 19.2s

T4 flowrate = 35 T4 greenlight= 16.8s

Total flow = 125

updated flowrates:

Cycle time = 70s

T1 flowrate = 40 T1 greenlight= 13.65s

T2 flowrate = 50 T2 greenlight= 17.1s

T3 flowrate = 70 T3 greenlight= 23.9s

T4 flowrate = 45 T4 greenlight= 15.37s

Total flow = 205

Calculating green light times…

ID: 1

State: Active

Green light time: 9.6 seconds

ID: 2

State: Active

Green light time: 14.4 seconds

ID: 3

State: Active

Green light time: 19.2 seconds

ID: 4

State: Active

Green light time: 16.8 seconds

T1: green

T2: red

T3: red

T4: red

\*9.6 second delay\*

T1: yellow

T2: red

T3: red

T4: red

\*6 second delay\*

T1: red

T2: green

T3: red

T4: red

\*14.4 second delay\*

T1: red

T2: yellow

T3: red

T4: red

\*6 second delay\*

T1: red

T2: red

T3: green

T4: red

\*19.2 second delay\*

T1: red

T2: red

T3: yellow

T4: red

\*6 second delay\*

T1: red

T2: red

T3: red

T4: green

\*16.8 second delay\*

T1: red

T2: red

T3: red

T4: yellow

\*6 second delay\*

(Keeps looping until 5 minutes have passed)

Updating timings...

ID: 1

State: Active

Green light time: 13.67 seconds

ID: 2

State: Active

Green light time: 17.1 seconds

ID: 3

State: Active

Green light time: 23.9 seconds

ID: 4

State: Active

Green light time: 15.37seconds

T1: green

T2: red

T3: red

T4: red

\*13.65 second delay\*

T1: yellow

T2: red

T3: red

T4: red

\*6 second delay\*

T1: red

T2: green

T3: red

T4: red

\*17 second delay\*

T1: red

T2: yellow

T3: red

T4: red

\*6 second delay\*

T1: red

T2: red

T3: green

T4: red

\*23 second delay\*

T1: red

T2: red

T3: yellow

T4: red

\*6 second delay\*

T1: red

T2: red

T3: red

T4: green

\*15.37 second delay\*

T1: red

T2: red

T3: red

T4: yellow

\*6 second delay\*

(Keeps looping)

Case 3: Checking if the droplight and addlight function works

Adding an extra Trafficlight 5 using addlight

After one loop of the simulation Trafficlight 2 will be removed using droplight

Case 4: Testing the setoff and setlightarrayoff functions (if a traffic light stops working)

Set Trafficlight 2 to inactive

Algorithm:

Trafficlight.h

Define YELLOWTIME as 6

Define class Trafficlight.h

Private members

ID as integer

State as integer

greentime as double

Public members

Trafficlight()

Assign ID to 1

Assign State to 1

Assign greentime to 0

Increment ID

Trafficlight(Greentime)

Assign ID to 1

Assign State to 1

Assign greentime to Greentime

Increment ID

Wait()

If State is equal to 3

Then the program will pause for greentime seconds

Otherwise If State is equal to 2

Then the program will pause for YELLOWTIME seconds

setgreentime(time)

Assign greentime to time

printLightInfo()

Print “ID: “,ID

If State is not equal to 0

Then print “State: Active”

Otherwise

Print “State: Inactive”

Print “Green light time: “,greentime,”seconds”

DisplayLightState(i)

If State is equal to 1

Print “T”,i,”: red”

Otherwise If State is equal to 2

Print “T”,i,”: yellow”

Otherwise If State is equal to 3

Print “T”,i,”: green”

Otherwise If State is equal to 0

Print “T”,i,”: Inactive”

Setoff()

Assign State to 0

Setred()

Assign State to 1

Setyellow()

Assign State to 2

Setgreen()

Assign State to 3

Intersection.h

Define MAXLIGHTS as 12

Define CURRENTLIGHTS as 4

Define class Intersection

Private member

Define Totalflow as double and assign to 0 Define Cycle\_length as double and assign to 0

Define Q[MAXLIGHTS] as an array of doubles

Define NumOfTrafficlights as integer

Define lightarray[MAXLIGHTS] as an array of objects of class Trafficlight

Define currentlights as an integer and assign to CURRENTLIGHTS +1

Public member

Intersection()

Assign NumOfTrafficlights to 0

Assign all members of Q[MAXLIGHTS-1] to 0

Assign Cycle\_length to 0

Assign all members of lightarray[MAXLIGHTS-1] to 0

printIntersectionInfo()

Print “Number of traffic lights: “,NumOfTrafficights,

Print “Cycle length: “,Cycle\_length,

Print “Total flow: “,Totalflow

Addlight(i)

Assign lightarray[i] to Trafficlight(i)

Increment NumOfTrafficlights

DropLight(ID)

Assign j to 1

While j is less than currentlights

lightarray[j]=lightarray[j+1]

Increment j

Decrement NumOfTrafficLights

Decrement currentlights

getcurrentlights()

return the integer currentlight

getNumOfTrafficLights()

return the integer NumOfTrafficLights

run()

Print "Calculating green light timings..."

Print “”

Call updateTiming function of class Intersection for object I1

Assign l to 1

While l is less than call getcurrentlights function of class Intersection for object

Call printLightInfo(l) function of class Intersection for object I1

Increment l

Assign start to the current time

While //infinite loop

Assign i to 1

While i is less than currentlights

If state is greater than 0

call setgreen function of class Trafficlight for object lightarray[i]

Assign j to 1

While j is less than currentlights

call DisplayLightState function of class Trafficlight for object lightarray[i]

Increment j

Print “”

Print “”

Call wait function of class Trafficlight for object

If state is greater than 0

lightarray[i]Call setyellow function of class Trafficlight for object lightarray[i]

Assign k to 1

While k is less the currentlights

Call DisplayLightState function of class Trafficlight for object lightarray[k]

Increment k

Call wait function of class Trafficlight for object lightarray[i]

If state is greater than 0

Call setred function of class Trafficlight for object lightarray[i]

Print “”

Print “”

Increment i

Assign end to the current time

Assign elapsed\_time to end – start

If elapsed\_time is greater than or equal to REFRESHTIME

Then call updateTiming function of class Intersection for object I1

Print “Updating timings…”

Print “”

Assign m to 1

While m is less than call getcurrentlights function of class Intersection for I1

Call Addlight(m) function of class Intersection for object I1

Increment m

Assign start to current time

getlightarray(i)

Return object of Trafficlight class from array lightarray[i]

Setlightarraystateoff(i)

Call setoff function on Trafficlight object lightarray[i]

readTrafficData()

Open infile “in.txt”

If infile fails to open

Then Print File could not be opened

Exit the program

Assign input from infile to Cycle\_length

Assign i to 1

While i is less than currentlights

Assign input from infile to Q[i]

Increment Totalflow by Q[i]

Increment i

updateTiming()

Call readTrafficData function of class Intersection

Define time as double

Assign i to 1

While i is less than currentlights

If state is equal to 0

Call setgreentime(0) function on lightarray[i]

Otherwise

Assign time to (Q[i]/Totalflow)\*Cycle\_length

Call setgreentime(time) function of class Trafficlight for object lightarray[i]

Increment i

Assign Totalflow to 0

Source.cpp

Define REFRESHTIME as 86400

Main

Define I1 as an object of class Intersection

Assign i to 1

While i is less than MAXLIGHTS

Call Addlight(i) function of class Intersection for object I1

Increment i

Call run function of class Intersection for object I1

Code:

Trafficlight.h

#pragma once

#include <iostream>

#include <Windows.h>

#include <time.h>

#include <fstream>

#define YELLOWTIME 6 //The amount of time the yellow light stays on

using namespace std;

class Trafficlight {

private: //Private variable declaration

int ID, State;

double greentime;

public:

Trafficlight() { //Default constuctor for the Trafficlight class

ID = 1;

State = 1;

greentime = 0;

}

Trafficlight(int id) { //Non-default constuctor for the Trafficlight class

ID = id;

State = 1;

greentime = 0;

}

void wait() { //Function used for the time delay in the simulation

if (State == 3) {

Sleep(greentime \* 1000); //This function pauses the program for a duration of the inputed time in milliseconds, so we mutiply by 1000 for seconds

}

else if(State == 2){

Sleep(YELLOWTIME \* 1000);

}

}

int getstate() {

return State;

}

void setgreentime(double time) { //Used to set the green time

greentime = time;

}

void printLightInfo() { //Prints all the information of a Trafficlight object

cout << "ID: " << ID << endl;

if (State != 0) {

cout << "State: Active" << endl;

}

else {

cout << "State: Inactive" << endl;

}

cout << "Green light time: " << greentime << " seconds" << endl;

}

void DisplayLightState(int i) { //Used to display the light state in the simulation

if (State == 1) {

cout << "T" << i << ": red" << endl;

}

else if (State == 2) {

cout << "T" << i << ": yellow" << endl;

}

else if (State == 3) {

cout << "T" << i << ": green" << endl;

}

else if (State == 0) {

cout << "T" << i << ": Inactive" << endl;

}

}

void setoff() { //Used to set the state to off

State = 0;

}

void setred() { //Used to set the state to red

State = 1;

}

void setyellow() { //Used to set the state to yellow

State = 2;

}

void setgreen() { //Used to set the state to green

State = 3;

}

};

Intersection.h

#pragma once

#include <iostream>

#include <Windows.h>

#include <time.h>

#include <fstream>

#include "Trafficlight.h"

#define MAXLIGHTS 12 //The max possible amount of lights in an intersection

#define CURRENTLIGHTS 4 //The amount of trafficlights used in the current simulation

#define REFRESHTIME 86400 //The time interval that the simulation updates the green light times (24hrs = 86400s)

using namespace std;

class Intersection {

private: //Private variable declaration

double Totalflow = 0; //The total flow rate of cars in the intersection

double Cycle\_length = 0; //The total amount of time a loop through the intersection takes

double Q[MAXLIGHTS]; //The flow rate of each traffic light

int NumOfTrafficlights; //Variable to keep track of the number of traffic lights

int currentlights= CURRENTLIGHTS +1;

Trafficlight lightarray[MAXLIGHTS]; //Array of Trafficlight objects

public:

Intersection() { //Default constuctor for the Intersection class

NumOfTrafficlights = 0;

Q[MAXLIGHTS - 1] = { 0 };

Cycle\_length = 0;

lightarray[MAXLIGHTS - 1] = { 0 };

}

void printIntersectionInfo() {

cout <<"Number of traffic lights: "<< NumOfTrafficlights << endl;

cout <<"Cycle length: "<<Cycle\_length << endl;

cout << "Total flow: " << Totalflow << endl;

}

void AddLight(int i) { //Used too add a Trafficlight object to the lightarray array

lightarray[i] = Trafficlight(i);

NumOfTrafficlights++;

}

void DropLight(int ID) { //Used too remove a Trafficlight object to the lightarray array

for (int j = ID; j < MAXLIGHTS;j++)

{

lightarray[j] = lightarray[j + 1];

}

NumOfTrafficlights--;

currentlights--;

}

int getcurrentlights() { //Used to access currentlights

return currentlights;

}

int getNumOfTrafficLights() { //Used to access NumOfTrafficLights

return NumOfTrafficlights;

}

void run() { //This function simulates one loop of the trafficlight intersection

cout << "Calculating green light timings..." << endl;

cout << endl;

updateTiming(); //Initializes the green time in each traffic light object

for (int l = 1;l < getcurrentlights();l++) { //This loop prints the information of all the trafficlights

getlightarray(l).printLightInfo();

}

time\_t start = time(NULL); //infinite loop to simulate a traffic intersection

for (;;) {

for (int i = 1;i < currentlights;i++) { //This for loop is for displaying the light state aas well as waiting the amount of time

if (getlightarray(i).getstate() > 0) {

lightarray[i].setgreen();

}

for (int j = 1;j < currentlights;j++) {

lightarray[j].DisplayLightState(j);

}

cout << endl;

cout << endl;

lightarray[i].wait();

if (getlightarray(i).getstate() > 0) {

lightarray[i].setyellow();

}

for (int k = 1;k < currentlights;k++) {

lightarray[k].DisplayLightState(k);

}

lightarray[i].wait();

if (getlightarray(i).getstate() > 0) {

lightarray[i].setred();

}

cout << endl;

cout << endl;

}

time\_t end = time(NULL); //Keep track of the time the simulation starts

double elapsed\_time = difftime(end, start);

if (elapsed\_time >= REFRESHTIME) { //if the time elapsed is more than 86400 which is 24 hrs in seconds then refresh the flow rates/green light times

cout << "Updating timings..." << endl;

cout << endl;

updateTiming(); //Refreshes the green times

for (int m = 1;m < getcurrentlights();m++) { //This loop prints the information of all the trafficlights

getlightarray(m).printLightInfo();

}

start = time(NULL); //resets the start time so the time interval resets

}

}

}

Trafficlight getlightarray(int i) { //Used to get access to the private lightarray object

return lightarray[i];

}

void setlightarraystateoff(int i) {

lightarray[i].setoff();

}

void readTrafficData() { //Used to get data from a file into the variables of the Intersection object

ifstream infile;

infile.open("in.txt", ios::in);

if (infile.fail()) //Error message if file fails to open

{

cerr << "File could not be opened" << endl;

exit(1);

}

infile >> Cycle\_length; //Inputting the Data into their respective variables

for (int i = 1;i < currentlights;i++) {

infile >> Q[i];

Totalflow += Q[i];

}

}

void updateTiming() { //Used to calculate the green light time from the provided data in the file

readTrafficData();

double time;

for (int i = 1;i < currentlights;i++) {

if (getlightarray(i).getstate() == 0) {

lightarray[i].setgreentime(0);

}

else {

time = (Q[i] / Totalflow) \* Cycle\_length;

lightarray[i].setgreentime(time);

}

}

Totalflow = 0;

}

};

Source.cpp

#include <iostream>

#include <Windows.h>

#include <time.h>

#include <fstream>

#include "Intersection.h"

using namespace std;

int main() {

Intersection I1;

for (int i = 1;i < MAXLIGHTS;i++) { //For loop to declare all the objects.

I1.AddLight(i);

}

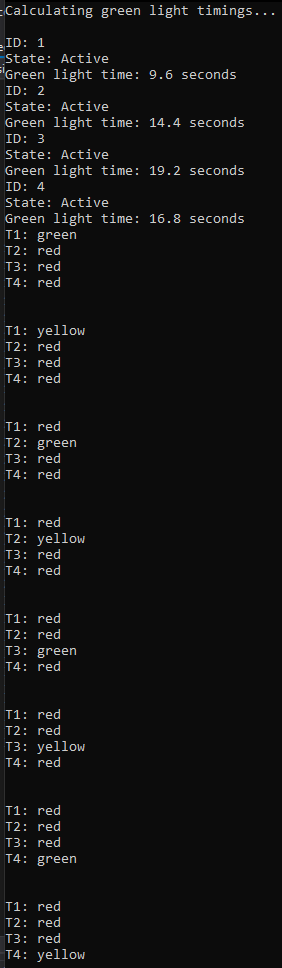
I1.run();

return 0;

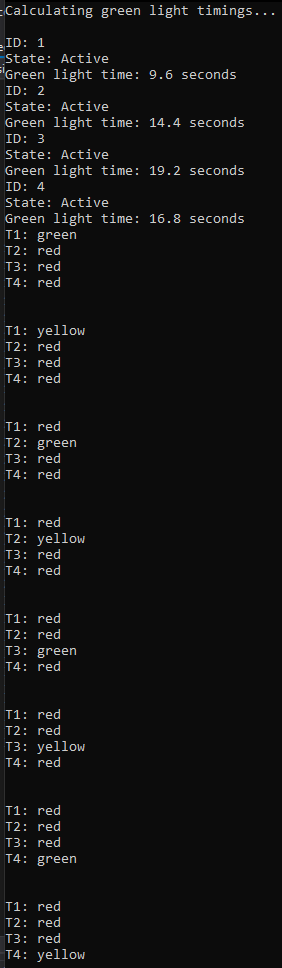
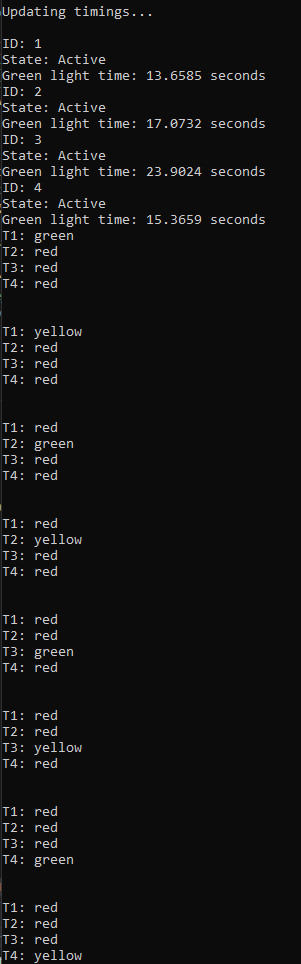
}

Testing and Verification:

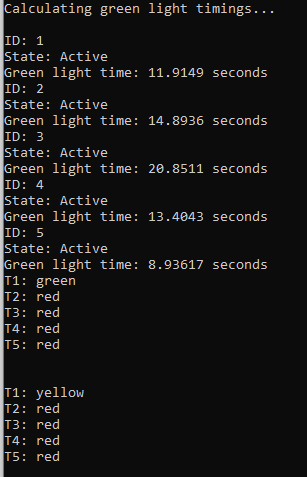
Case 1:



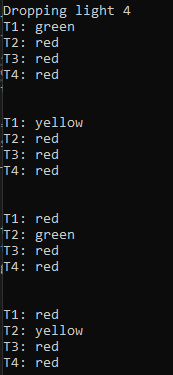
Case 2:

Case 3:



(After the end of the first loop)



Case 4:

