計算機程式與應用hw7

/\* H349961598 侯瑋明 g6

This code can be compiled and run ok.

Simulate the system of bicycle's pickup and return process.

usage (how to run):

./hw7 (then input filename)

input file:

HW7\_input.txt

output files:

none

compile (how to compile):

g++ -o hw7 hw7.cpp

pseudocode:

自行車租借模擬：

List \*min;

while( demList.next != NULL ){

min = demList.next;

if(D[min->index].status==0){//抵達租借站點要借車

if(租借站點的自行車數>0) {

借車(bike-- & spaces++);

更新min->time (即再加上租還車站點間騎乘自行車時間);

視min->time大小插入demList中適當的位置;

D[min->index].status改為1;

if(該站點有人等待還車){

讓最早等待還車的人還車;

}

} else {

將租借者放到等待借車的List中;

}

} else if(dmand[min->index].status==1){ //抵達租借站點要還車

if(租借站點的空停車柱數>0){

還車(bike++ & spaces--);

D[min->index].status改為2;

if(該站點有人等待借車){

讓最早等待借車的人借車;

}

} else {

將租借者放到等待還車的List中;

}

}

}

coded by Wei-Ming Hou, group: 6, ID: h34996158, email: jacob0701@hotmail.com.tw

date: 2011.06.12

\*/

#include <iostream>

#include <fstream>

using namespace std;

struct Station{

int bikes;//記錄目前租借站擁有的自行車數

int spaces;//記錄目前租借站擁有的空位數

};

struct Demand{

int time;//使用者需求發生時間

int pickupSta;//使用者租車站編號

int returnSta;//使用者還車站編號

int status;//使用者租還車狀態

int pickupWait;//使用者租車等待時間

int returnWait;//使用者還車等待時間

};

struct List{

int time;//事件發生時間

int index;//使用者編號

List \*next;//記錄下一節點位址

};

void readFile(char \*filename, Station \*&, Demand \*&, int \*\*&, int &, int &, List \*&, List \*&);

void timeList(List &, Demand \*&, int );

void insertTime(List &, Demand \*& , int );

void deleteTime(List &, int );

int main()

{

Station \*S; //租借站陣列資訊

Demand \*D; //租借者陣列資訊

int \*\*rideTime; //站點間騎乘時間矩陣

int staNum; //租借站點個數，即為N

int demNum; //租借者個數，即為m

List demList; //模擬的時間序列

List \*pickupQ; //記錄各站點等待借車的租借者

List \*returnQ; //記錄各站點等待還車的租借者

int i=0, tmpTime, a;

char filename[50];

cout << "Please intput filename: ";

cin >> filename;

readFile(filename, S, D, rideTime, staNum, demNum, pickupQ, returnQ);

int &N=staNum, &m=demNum, \*\* &T=rideTime;

timeList(demList, D, demNum);

//--------------------------------------------------------------------------

//---PART 2.2 開始進行模擬--------------------------------------------------

List \*min;

while( demList.next != NULL ){

min = demList.next;

a=min->time; //紀錄最後時間

if(D[min->index].status==0){ //抵達租借站點要借車

if(S[D[min->index].pickupSta].bikes>0) {

S[D[min->index].pickupSta].bikes--; //借車

S[D[min->index].pickupSta].spaces++;

D[min->index].time+=T[D[min->index].pickupSta][D[min->index].returnSta]; //更新min->time (即再加上租還車站點間騎乘自行車時間)

insertTime(demList, D, min->index); //視min->time大小插入demList中適當的位置

D[min->index].status=1;

if(returnQ[D[min->index].pickupSta].next!=NULL){

//等待還車的人還車

D[returnQ[D[min->index].pickupSta].next->index].status=2;

S[D[returnQ[D[min->index].pickupSta].next->index].returnSta].bikes++;

S[D[returnQ[D[min->index].pickupSta].next->index].returnSta].spaces--;

D[returnQ[D[min->index].pickupSta].next->index].returnWait=(min->time)-D[returnQ[D[min->index].pickupSta].next->index].time; //歸還等待時間

deleteTime(returnQ[D[min->index].pickupSta], D[returnQ[D[min->index].pickupSta].next->index].time); //更新returnQ

}

deleteTime(demList, min->time); //更新demList

}

else{

D[min->index].time=min->time;

insertTime(pickupQ[D[min->index].pickupSta], D, min->index); //將租借者放到等待借車的List中

deleteTime(demList, min->time); //更新demList

}

}

else if(D[min->index].status==1){ //抵達租借站點要還車

if(S[D[min->index].returnSta].spaces>0){

S[D[min->index].returnSta].bikes++; //還車

S[D[min->index].returnSta].spaces--;

D[min->index].status=2;

if(pickupQ[D[min->index].returnSta].next!=NULL){

//等待借車的人借車

D[pickupQ[D[min->index].returnSta].next->index].status=1;

S[D[pickupQ[D[min->index].returnSta].next->index].pickupSta].bikes--;

S[D[pickupQ[D[min->index].returnSta].next->index].pickupSta].spaces++;

D[pickupQ[D[min->index].returnSta].next->index].pickupWait=(min->time)-(D[pickupQ[D[min->index].returnSta].next->index].time); //借車等待時間

tmpTime=D[pickupQ[D[min->index].returnSta].next->index].time; //暫存要刪的時間

(D[pickupQ[D[min->index].returnSta].next->index].time)+=((D[pickupQ[D[min->index].returnSta].next->index].pickupWait)+(rideTime[D[pickupQ[D[min->index].returnSta].next->index].pickupSta][D[pickupQ[D[min->index].returnSta].next->index].returnSta])); //增加站跟站之間的時間

insertTime(demList, D, pickupQ[D[min->index].returnSta].next->index);

deleteTime(pickupQ[D[min->index].returnSta], tmpTime); //更新pickupQ

}

deleteTime(demList, min->time); //更新demList

}

else{

D[min->index].time=min->time;

insertTime(returnQ[D[min->index].returnSta], D, min->index);//將租借者放到等待還車的List中

deleteTime(demList, min->time); //更新demList

}

}

}

//---計算租還車的使用時間

for(i=0; i<m; i++){

if(D[i].status==0)

D[i].pickupWait=a-D[i].time;

if(D[i].status==1)

D[i].returnWait=a-D[i].time;

}

//---計算各站點在結束時仍有等待租車或還車之需求其等待數目

int \*waitPick, \*waitReturn;

waitPick = new int[N];

waitReturn = new int[N];

for(i=0; i<N; i++){

waitPick[i]=0;

waitReturn[i]=0;

}

for(i=0; i<m; i++){

if(D[i].status==0)

waitPick[D[i].pickupSta]++;

else if(D[i].status==1)

waitReturn[D[i].returnSta]++;

}

//---計算 無車可租的需求總數及其總等待時間 與 無位可還的需求總數及其總等待時間

int count1=0, count2=0, pickupWaitSum=0, returnWaitSum=0;

for(i=0; i<m ; i++){

if(D[i].pickupWait!=0){

count1++;

pickupWaitSum+=D[i].pickupWait;

}

if(D[i].returnWait!=0){

count2++;

returnWaitSum+=D[i].returnWait;

}

}

//--------------------------------------------------------------------------

//---PART3 程式結果輸出-----------------------------------------------------

for(i=0; i<N; i++){ //各站點的自行車數與空停車數

cout << "station " << i << " : " << S[i].bikes << " bikes & " << S[i].spaces << " empty racks" << endl;

}

cout <<endl;

for(i=0; i<N; i++){ //若站點在結束時尚有等待租車或還車之需求，印出其等待者數目

if(waitPick[i]!=0){

cout << "station " << i << " : " << waitPick[i] << " customers wait for picking up bikes " << endl;

}

if(waitReturn[i]!=0){

cout << "station " << i << " : " << waitReturn[i] << " customers wait for returning bikes " << endl;

}

}

cout << endl;

cout << count1 << " customers has experienced \"no bike to pickup\", and wait for " << pickupWaitSum << " units." << endl //曾經歷無車可租之需求總數及其總等待時間

<< count2 << " customers has experienced \"no bike to pickup\", and wait for " << returnWaitSum << " units." << endl; //曾經歷無位可還之需求總數及其總等待時間

//--------------------------------------------------------------------------

system("pause");

return 0;

}

void readFile(char \*filename, Station \*&S, Demand \*&D, int \*\*&rideTime, int &staNum, int &demNum, List \*&pickupQ, List \*&returnQ)

{

char z;

int i=0, j=0, k=0, l=0;

int &N=staNum, &m=demNum, \*\* &T=rideTime;

ifstream inFile(filename);

while(!inFile.eof()) //讀檔

{

inFile >> z;

switch(z)

{

case 'N':

inFile >> N;

S = new Station[N]; //配置記憶體空間S

T = new int\*[N]; //配置記憶體空間T

for(i=0; i<N; i++){

T[i]=new int[N];

}

pickupQ = new List[N]; //配置記憶體空間pickupQ

returnQ = new List[N]; //配置記憶體空間returnQ

for(i=0; i<N; i++){

pickupQ[N].next=NULL;

pickupQ[N].next=NULL;

}

break;

case 'm':

inFile >> m;

D = new Demand[m]; //配置記憶體空間D與部份值歸0

for(i=0; i<m; i++){

D[i].status=0;

D[i].pickupWait=0;

D[i].returnWait=0;

}

break;

case 'S':

inFile >> S[j].bikes >> S[j].spaces;

j++;

break;

case 'T':

for(i=0; i<N; i++)

inFile >> T[k][i];

k++;

break;

case 'D':

inFile >> D[l].time >> D[l].pickupSta >> D[l].returnSta;

l++;

break;

}

}

inFile.close();

}

void timeList(List &demList, Demand \*&D, int demNum)

{

int i;

int insertVal; //紀錄插入時間大小

List \*tmp, \*t;

tmp=&demList;

tmp->time=0; //從時間0開始跑

tmp->index=-1;

tmp->next=NULL;

for(i=0; i<demNum; i++){

insertVal=D[i].time;

while(tmp->next != NULL){ //步驟一:找出插入位置

if(tmp->next->time > insertVal){

break;

}

tmp=tmp->next;

}

t=new List(); //步驟二:產生一新節點

t->time=insertVal;

t->index=i;

t->next=tmp->next; //步驟三

tmp->next=t; //步驟四

tmp=&demList; //回到串列最初

}

}

void insertTime(List &demList, Demand \*& D, int insertIndex)

{

List \*tmp, \*t;

tmp=&demList;

while(tmp->next != NULL){ //步驟一:找出插入位置

if(tmp->next->time > D[insertIndex].time){

break;

}

tmp=tmp->next;

}

t=new List(); //步驟二:產生一新節點

t->time=D[insertIndex].time; //插入節點的時間

t->index=insertIndex; ////插入節點的編號

t->next=tmp->next; //步驟三

tmp->next=t; //步驟四

}

void deleteTime(List &demList, int deleteVal)

{

List \*tmp, \*t;

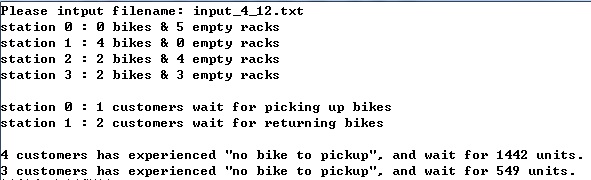
tmp=&demList;

while(tmp->next != NULL){ //步驟一:找出要刪除結點之位置

if(tmp->next->time == deleteVal){

break;

}

 tmp=tmp->next;

}

t=tmp->next;

tmp->next=t->next; //步驟二

t->next=NULL; //步驟三

delete t;

}