1η Εργασία

Δομές Δεδομένων

Σοφράς Αντώνης – Π10119

Δημακογιάννη Μαρία – Π17025

Υλοποίηση Απλά Συνδεδεμένης Λίστας:

Αρχείο ListNode.h:

Δήλωση templated κλάσης List (περισσότερα στο αρχείο List.h).

Δήλωση και υλοποίηση templated κλάσης ListNode.

Υλοποίηση κλάσης ListNode:

Δήλωση της κλάσης σαν friend της List για εύκολη πρόσβαση αυτής στα πεδία data και next.

Constructor της κλάσης θα δέχεται ένα όρισμα το οποίο θα είναι τα δεδομένα αυτού του node και θα κάνει τον δείκτη ο οποίος δείχνει στο επόμενο ListNode να δείχνει στο NULL.

Υλοποιούμε και μία μέθοδο getData() όπου μας επιστρέφει τα δεδομένα του node σε περίπτωση που χρειαστεί.

Αρχείο List.h:

Δήλωση και υλοποίηση templated κλάσης List.

Υλοποίηση κλάσης List:

Πεδία *head και *tail: Δύο δείκτες σε αντικείμενο ListNode (κόμβο της λίστας), αρχή και τέλος της λίστας αντίστοιχα.

List(): Ο default constructor της κλάσης αρχικοποιεί τα πεδία head και tail και βάζει τους δείκτες να δείχνουν στο NULL.

List(const List & src):head(NULL): Copy constructor για την αντιγραφή λιστών. Γεμίζει την νέα λίστα με τα περιεχόμενα της src αφου αρχικοποιήσει τον δείκτη head της να δείχνει στο NULL.

~List(): Ο destructor της κλάσης αφού ελέγξει ότι η λίστα δεν είναι άδεια, χρησιμοποιεί δύο δείκτες (current και temp), τον έναν (currrent) για να κάνει iterate τα nodes και τον temp για να αποδεσμεύσει την μνήμη στην εκάστοτε θέση.

operator=(List<T> src): Overloaded = operator για καταχώρηση λίστας σε μία άλλη χρησιμοποιώντας την έτοιμη συνάρτηση std::swap().

bool isEmpty(): Επιστρέφει true εφόσον οι δείκτες head και tail της λίστας δείχνουν στο NULL, συνεπώς η λίστα είναι κενή.

Int insertStart(const T& dataIn): Επιστρέφει 1 εφόσον κατάφερε να εισάγει το dataIn στην αρχή της λίστας, ειδάλλως επιστρέφει 0.

int insertEnd(const T& dataIn): Επιστρέφει 1 εφόσον κατάφερε να εισάγει το dataIn στο τέλος της λίστας, ειδάλλως επιστρέφει 0.

int insertPos(const T& dataIn, int pos): Επιστρέφει 1 εφόσον κατάφερε να εισάγει το dataIn στην θέση pos της λίστας (η αρίθμηση ξεκινάει από το 0), ειδάλλως επιστρέφει 0, όπως και αν ο χρήστης δώσει αρνητική θέση ή θέση που δεν υπάρχει στην λίστα.

Int search(T& key): Επιστρέφει την θέσει που βρέθηκε το στοιχείο key μέσα στην λίστα, ειδάλλως επιστρέφει -1 εφόσον δεν βρέθηκε.

Void print(): Εκτυπώνει τα δεδομένα της λίστας στην οθόνη.

Void print(ofstream &tempResultsWrite): Εκτυπώνει τα δεδομένα της λίστας σε κάποιο ανοιχτό αρχείο στο stream tempResultsWrite.

Int deleteStart(T& deleted): Επιστρέφει 1 εφόσον κατάφερε να διαγράψει το 1ο στοιχείο της λίστας το οποίο καταχωρεί στο όρισμα deleted. Ειδάλλως επιστρέφει 0 εφόσον η λίστα είναι άδεια.

Int deletePos(T& deleted, int pos): Επιστρέφει 1 εφόσον κατάφερε να διαγράψει το στοιχείο της λίστας στην θέση pos, ειδάλλως επιστρέφει 0.

int length(): Επιστρέφει το πλήθος των αρχείων της λίστας.

Bool findElem(int pos, T& elem): Επιστρέφει true αν κατάφερε να καταχωρήσει στο όρισμα elem το στοιχείο της λίστας στην θέση pos, ειδάλλως επιστρέφει false.

Αρχείο POI.h:

Δήλωση και υλοποίηση κλάσης POI (Point Of Interest).

Private πεδία:

int Id: Ο αύξων αριθμός του αξιοθέατου

double x,y: Οι συντεταγμένες χ, y, αντίστοιχα.

Int d: Ο χρόνος που απαιτείται για μία επίσκεψη.

Int s: Η βαθμολογία του αξιοθέατου.

Int open[7], close[7]: Πίνακες 7 θέσεων για τις τιμές των ωρών ανοίγματος και κλείσιμου κάθε αξιοθέατου αντίστοιχα.

Μέθοδοι:

POI() { }: default constructor.

POI(int vn, double xcoord, double ycoord, int opent, int closet): Overloaded constructor για το ξενοδοχείο που έχει ειδικές τιμές.

Operator==(const POI& rhs) const και != αντίστοιχα: Overload των operators == και != για να γίνεται εφικτά γρήγορα η σύγκριση 2 POI σύμφωνα με το id τους.

Οι υπόλοιπες μέθοδοι της κλάσης είναι **Getters** των private πεδίων.

Αρχείο ergasia.cpp:

Συναρτήσεις εκτός της main:

std::ostream& operator<<(std::ostream& os, POI& p): Overloaded << operator για την εύκολη εκτύπωση αντικειμένων POI, τυπώνουμε το ID τους και το Score τους (χρησιμοποιώντας τους getters θα μπορούσαμε να τυπώσουμε οποιοδήποτε άλλο πεδίο του αντικειμένου POI).

Double calculateDistance(POI start, POI end): Επιστρέφει την ευκλείδια απόσταση μεταξύ 2 αξιοθεάτων.

int calculateScore(int M ,List<POI> d[]): Επιστρέφει το συνολικό Score όλων των δρομολογίων μέσα σε έναν πίνακα μεγέθους M από δρομολόγια.

int calculateScore(List<POI> trip): Επιστρέφει το συνολικό Score των αξιοθεάτων ενός δρομολογίου.

double calculateOf(POI start, POI end, POI between): Επιστρέφει το όφελος σύμφωνα με τον τύπο που μας δίνεται.

double timeAddition(POI start, POI end, POI between): Επιστρέφει την αύξηση του χρόνου στο δρομολόγιο με την πρόσθηκη του αξιοθέατου between ανάμεσα στα start και end.

Main():

Στο πρώτο κομμάτι της main δηλώνουμε τις περισσότερες μεταβλητές που θα μας χρειαστούν:

Οι 3 πρώτες δίνονται ως ορίσματα κατά την εκτέλεση και είναι οι εξής:

int epan: αριθμός επαναλήψεων που θα τρέξει ο αλγόριθμος

double pie: συντελεστής π

string filename: όνομα αρχείου test instance.

Έπειτα δηλώνουμε τις μεταβλητές που θα διαβάσουμε από το αρχείο. Τα ονόματα είναι τα ίδια.

Διαβάζοντας το entry του ξενοδοχείου χρησιμοποιούμε τον overloaded constructor και δημιουργούμε τον pointer που δείχνει σε αυτό. Έπειτα αρχικοποιούμε έναν πίνακα μεγέθους N (αριθμός αξιοθεάτων) **POI pois[N]** όπου θα τοποθετήσουμε τα αξιοθέατα και έναν μεγέθους M (αριθμός δρομολογίων) **List<POI> dromologia[M]** όπου θα τοποθετήσουμε τα δρομολόγια (λίστες). Αρχικοποιούμε άλλον έναν πίνακα μεγέθους M **currentTime[M]** με την ώρα που αρχίζει το εκάστοτε δρομολόγιο (το οποίο είναι το opening time του ξενοδοχείου).

Τέλος αρχικοποιούμε την μεταβλητή int maxScore = 0 και έναν πίνακα μεγέθους M List<POI> finalDromologia[M] όπου θα μας χρειαστούν στην εύρεση του βέλτιστου δρομολογίου.

Υλοποίηση προτεινόμενου αλγορίθμου:

Εξωτερικός Βρόγχος επανάληψης **epan** φορές, πρόκειται για τον αριθμό φορών που θα τρέξει ο αλγόριθμος. Όλα τα άλλα βρίσκονται μέσα σε αυτόν εκτός από τα τελικά αποτελέσματα.

Επόμενος βρόγχος **N (αριθμός αξιοθεάτων)** φορές. Θα ξεκινήσουμε παίρνοντας αξιοθέατο-αξιοθέατο. Σε αυτό το σημείο αρχικοποιώ την μεταβλητή **bool found = false** την οποία θα χρησιμοποιήσω σαν συνθήκη με την βοήθεια της μεθόδου search() της λίστας μου για να εξετάσω αν το αξιοθέατο i βρίσκεται σε κάποιο από τα δρομολόγια. Αν ναι, θα προχωρήσω στο επόμενο αξιοθέατο. Αυτό θα συμβεί επίσης στην περίπτωση που το αξιοθέατο που εξετάζω έχει χρόνο ανοιγματος = χρονο κλεισίματος = 0 το οποίο σημαίνει ότι ειναι κλειστό την συγκεκριμένη μέρα.

Εαν όχι αρχικοποιώ τις μεταβλητές **double maxOf = 0**, **int savedM**, **savedJ**, **double savedTime** και επιστρέφω την **SD** στην αρχική της κατασταση.

Μπαίνω στον επόμενο βρόγχο **M (αριθμος δρομολογίων)** φορές. Στην μεταβλητη **list_sz** καταχωρο το μέγεθος του εκαστοτε δρομολογίου με την βοήθεια της μεθόδου length() της λίστας.

Έπειτα μπαίνω και στον τελευταίο βρόγχο όπου ξεκινάω από την 1η θέση (0 θέση ξενοδοχείο) της λίστας και κουνάω τον δείκτη **j** για να προσπελάσω όλες τις θέσεις της λίστας, αφού κάνω έναν έλεγχο για να αποφύγω να βγω out of bounds.

Σε αυτό το κομμάτι του αλγορίθμου έπειτα από διάφορους ελέγχους (βλ. Σχόλια στον κώδικα) βρίσκω την βέλτιστη θέση στο βέλτιστο δρομολόγιο γι’αυτό το κάθε αξιοθέατο και κάνω την εισαγωγή. Όταν αυτά γεμίσουν συγκρίνω το συνολικό **Score** τους (άθροισμα βαθμολογιών αξιοθεάτων δρομολογίου) και αν αυτό είναι μεγαλύτερο από το **maxScore** κρατάω τον συγκεκριμένο συνδυασμό δρομολογίων.

Στο επόμενο κομμάτι εκτελώ τις διαγραφές σύμφωνα με τα μεγέθη των δρομολογίων και τον συντελεστή **π** που δίνει ο χρήστης σαν όρισμα (εκτός απο την τελευταία φορά που θα τρέξει ο αλγόριθμος) και προχωράει στην επόμενη επανάληψη.

Προσοχή. Στον τύπο που μας δίνεται για αυτόν τον αλγόριθμο δεν λαμβάνεται υπ’όψιν η ώρα ανοίγματος του αξιοθέατου και τι συμβαίνει αν ο χρήστης πάει σε αυτό πριν αυτό ανοίξει. Χρησιμοποιήσαμε τον τύπο όπως μας δώθηκε.

Ο κώδικας έγινε compile από τον g++ 5.4.0 και έτρεξε σε περιβάλλον kde

Παράδειγμα εκτέλεσης:

ergasia.exe 1000 0.5 t101.txt

Κώδικας αρχείου ListNode.h

```
#ifndef LIST_NODE_H
#define LIST_NODE_H

template <typename T>
class List;

template <typename T>
class ListNode
{
	friend class List<T>;
	public:
		ListNode(T);
		T getData();
	private:
		T data;
		ListNode* next;
};

template <typename T>
ListNode<T>::ListNode(T dataIn)
{
	data = dataIn;
	next = 0;
}

template <typename T>
T ListNode<T>::getData(){
	return data;
}

#endif
```

Κώδικας αρχείου POI.h

```
#include <iostream>
using namespace std;
#pragma once

class POI {
private:
	int id;
	double x, y;
	int d;
	int s;
	int open[7];
	int close[7];

public:

	 void POIinitilize(int vn,double xcoord,double ycoord,int dur,int score, int opent[], int closet[]){
		this->id = vn;
		this->x = xcoord;
		this->y = ycoord;
		this->d = dur;
		this->s = score;
		for (int i = 0; i < 7; i++){
			this->open[i] = opent[i];
			this->close[i] = closet[i];
		}
	}
	double getCoordinates(char d){
		if(d == 'x'){
			return this->x;
		}
		else{
			return this->y;
		}
	}
	int getDuration() {
		return this->d;
	}
	int getScore() {
		return this->s;
	}
	int getOpenTime(int day){
		for(int j=0;j<7;j++){
			if(day==j){
				return this->open[j];
			}
		}
	}
```

```
	int getCloseTime(int day){
		for(int j=0;j<7;j++){
			if(day==j){
				return this->close[j];
			}
		}
	}
	int getId(){
		return this->id;
	}

	POI() { }
	//hotel constructor
	 POI(int vn,double xcoord,double ycoord, int opent, int closet){
		id = vn;
		x = xcoord;
		y = ycoord;
		for (int i = 0; i < 7; i++){
			open[i] = opent;
			close[i] = closet;
		}
	}
	bool operator==(const POI& rhs) const
{
	return id == rhs.id;
}
	bool operator!=(const POI& rhs) const
{
	return id != rhs.id;
}
};
```

Κώδικας αρχείου ergasia.cpp

```
#include <fstream>
#include <string>
#include <math.h>
#include <cstdlib>
#include "List.h"
#include "POI.h"

//printing POIs
ostream& operator<<(ostream& os, POI& p)
{
	os<<"Id: "<<p.getId()<<" Score: "<<p.getScore()<<endl;
	return os;
}

//returns the distance between starting point and ending point
double calculateDistance(POI start, POI end){
	double distance;
	double endx = end.getCoordinates('x');
	double endy = end.getCoordinates('y');
	double startx = start.getCoordinates('x');
	double starty = start.getCoordinates('y');
	distance = pow((endx-startx),2)+pow((endy - starty),2);
	return sqrt(distance);
}

//returns the total score of all trips in array of trips
int calculateScore(int M ,List<POI> d[])
{
	POI temp;
	int score = 0;
	for(int i=0; i<M; i++)
	{
		for(int j=0; j<d[i].length(); j++)
		{
			d[i].findElem(j,temp);
			score += temp.getScore();
		}
	}
	return score;
}
//returns the total score of a trip
int calculateScore(List<POI> trip)
{
	POI temp;
	int score = 0;
	for(int i=0; i<trip.length(); i++)
	{
		trip.findElem(i,temp);
		score += temp.getScore();
	}
```

```
	return score;
}

//returns Oφ ratio according to type
double calculateOf(POI start, POI end, POI between){
	double of = pow(between.getScore(),2)/(calculateDistance(start,between)
+between.getDuration()+calculateDistance(between,end)-calculateDistance(start,end));
	return of;
}

//returns time added to the trip if POI between was inserted in between of Start and End
double timeAddition(POI start, POI end, POI between)
{
	return calculateDistance(start,between)+between.getDuration()
+calculateDistance(between,end)-calculateDistance(start,end);

}



//---------MAIN----------
int main(int argc, char* argv[]) {

//-----METAVLITES KAI DIAVASMA APO TEST INSTANCE-----
	int epan = atoi(argv[1]); //FIX ME
	double pie = atof(argv[2]);
	string filename = argv[3];
	int K, M, SD, N, id;
	double x, y;
	int d, s, open, close;
	int opena[7];
	int closea[7];
	string garbage;

	//Diavasma arxeiou, arxikopoihsh antikeimenwn POI kai eisagogi tous se pinaka
megethous N
	ifstream inFile(argv[3]);

	inFile >> K >> M >> SD >> N;
	inFile.ignore(256,'\n');
	inFile >> id >> x >> y >> d >> s >> open >> close;
	POI * hotel = new POI(id, x, y, open, close);
	POI pois[N];
	List<POI> dromologia[M];
	double currentTime[M];
	//topothetisi hotel stin arxi kai telos ton dromologion
	for(int i=0; i<M; i++){
		dromologia[i].insertStart(*hotel);
		dromologia[i].insertEnd(*hotel);
		currentTime[i] = hotel->getOpenTime(SD);
	}
```

```
	int counter = 0;
	while(!inFile.eof())
	{
		inFile >> id >> x >> y >> d >> s;
		inFile >> garbagecan;
		for(int i = 0; i<7; i++){
			inFile >> open;
			opena[i] = open;
			inFile >> close;
			closea[i] = close;
		}
		pois[id].POIinitilize(id,x,y,d,s,opena,closea);
		inFile.ignore(256,'\n');
		counter++;
	}
	inFile.close();


	int SD_ = SD;
	int maxScore = 0;
	List<POI> finalDromologia[M];

	//----ALGORITHMOS----
cout<<"Start of tour: "<<currentTime[0]<<endl;

for(int r = 0; r < epan; r++)
{
	for (int i=1; i < N; i++)
	{
		bool found = false;			//yparxei to pois[i] se ena apo ta tria
dromologia?
		for(int m=0; m < M; m++)
		{
			if(dromologia[m].search(pois[i])!=-1)
			{
				found = true;

				break;
			}
		}
		if(found or pois[i].getOpenTime(SD) == pois[i].getCloseTime(SD)) //ean nai h
an einai kleisto sinexise sto epomeno
		{
			continue;
		}
		double maxOf = 0;
		int savedM, savedJ;
		double savedTime;
		SD = SD_;
		for(int m=0; m < M; m++)
		{
			int list_sz = dromologia[m].length();
```

```
                        for (int j=1; j < list_sz; j++)
                        {
                                if(j+1 >= list_sz and j != 1)
                                {
                                        break;
                                }
                                POI start, end;
                                dromologia[m].findElem(j-1, start); //pairno to stoixio stin 8esi j-1
tis listas
                                dromologia[m].findElem(j, end); //to stoixio j
                                double currentTimeTemp = currentTime[m] + timeAddition(start,
end, pois[i]); //ypologizo ton xrono pou 8a mou pros8esei
                                //elegxo arxika ean o xronos pou exw stin tin apostasi aytou
pou
                                //tha prostheso einai mikroteros apo to kleisimo tou
ksenodoxiou (end of tour)
                                if(currentTimeTemp + calculateDistance(end, *hotel) <= hotel-
>getCloseTime(SD))
                                {
                                        bool outOfTime = false;
                                        //elegxos ean gia ta epomena aksio8eata tou
dromologiou
                                        //o xronos pou proste8ike den ftanei na ta episkeftoume
                                        for(int y=j+1; y < list_sz; y++)
                                        {
                                                POI next;
                                                dromologia[m].findElem(y, next);
                                                if(currentTimeTemp > next.getCloseTime(SD))
                                                {
                                                        outOfTime = true;
                                                        break;
                                                }
                                        }
                                        if (!outOfTime)
                                        {
                                                //an den exei ginei anefikti h episkepsi twn
epomenwn
                                                //vrisko to ofelos se ayti tin 8esi
                                                //kai elegxo an einai megalitero apo to maxOf
                                                double of = calculateOf(start, end, pois[i]);
                                                if (maxOf <= of)
                                                {
                                                        //an einai tote kratao tin thesi, to dromologio
                                                        //kai ton xrono pou tha exw meta tin
prosthiki tou
                                                        maxOf = of;
                                                        savedJ = j;
                                                        savedM = m;
                                                        savedTime = currentTimeTemp;
                                                }
                                        }
                                }
```

```
                        }
                        //ayksano tin mera
                        SD++;
                        if (SD > 6)
                        {
                                //an einai meta apo 6 (kiriaki), ginetai 0 (deytera)
                                SD = 0;
                        }
                }
                //exontas elegksei tis 8eseis olwn twn dromologiwn giayto to aksiotheato
                //elegxo oti sigoura exei allaksei to maxOf apo tin arxiki tou timi
                if (maxOf > 0)
                {
                        //kano enan teleytaio elegxo ean o xronos sto sigkekrimeno
dromologio
                        //stin sigkekrimeni thesi meta tin prosthiki tou aksiotheatou pou
eksetazo
                        //einai megaliteros h isws apo to kleisimo tou epomenou
                        POI temp;
                        dromologia[savedM].findElem(savedJ, temp);
                        if(savedTime >= temp.getCloseTime(SD))
                        {
                                //ean nai den tha ginei prosthiki
                                continue;
                        }
                        //alliws prostheto to aksiotheato sto dromologio
                        //kai rithmizo analogws ton xrono tou
                        dromologia[savedM].insertPos(pois[i], savedJ);
                        currentTime[savedM] = savedTime;
                        // cout<<"Inserted: "<<pois[i]<<" in: "<<savedM<<" at pos:
"<<savedJ<<endl;
                        // cout << "Last Closing time: " << pois[i].getCloseTime(SD)<<" of pos
" << savedJ << " with ID: " << pois[i].getId() << endl;
                        // cout<<currentTime[savedM]<<endl;
                        //cout << currentTime[savedM] << endl;
                }
        }
        //exontas eite gemisei ta dromologia, eite eksantlisei ta aksiotheata
        //proxoraw sto na ypologiso to score twn dromologiwn
        int totalScore = calculateScore(M,dromologia);
        if (maxScore<totalScore)
        {
                //ean einai megalitero tou maxScore to kratao
                //kai kratao ta dromologia sta finalDromologia
                maxScore = totalScore;
                for (int i=0; i<M; i++)
                {
                        finalDromologia[i] = dromologia[i];
                }

        }
```

```
	//stin teleytaia epanalipsi den thelo tis diagrafes pou akolou8oun
	if(r == epan-1)
		break;
	//diagrafes
	//gia kathe dromologio
	for(int i=0; i<M; i++)
	{
		//list_sz megethos dromologiou
		int list_sz = dromologia[i].length();
		//tixaio aksiotheato tou dromologiou
		int randomIndex = rand() % (list_sz -2) + 1;
		//posa tha diagrapso
		int deletionCount = (list_sz-2) * pie;
		int j = randomIndex;

			for(int counter = 0; counter < deletionCount; counter++)
			{
				POI temp,start,end;
				dromologia[i].findElem(randomIndex, temp);
				//ean ftasei sto ksenodoxio
				//sinexise na diagrafeis pera apo ayto
				//diladi thesi 1 tis listas
				if(temp == *hotel);
				{
					randomIndex = 1;
					dromologia[i].findElem(randomIndex, temp);
				}
				dromologia[i].findElem(randomIndex-1,start);
				dromologia[i].findElem(randomIndex+1,end);

				dromologia[i].deletePos(temp, randomIndex);
				//cout<<"Deleted: "<<temp<<"From drom: "<<i<<endl;
				//afairesi kai tis antistoixis prosthikis xronou
				//pou tha edine to diegrameno dromologio
				currentTime[i] -= timeAddition(start, end, temp);
			}

					//cout<<currentTime[i]<<" remaining now"<<endl;
	}
}

//-------FINAL RESULTS------
	cout<<"----------FINAL-----------"<<endl;
	for (int i=0; i<M; i++){
		cout<<"\nDromologio No: "<<i+1<<endl;
		finalDromologia[i].print();
		cout<<"End of tour: "<<currentTime[i]<<endl;
		cout<<"Score: "<<calculateScore(finalDromologia[i])<<endl;
		cout<<"Hotel closes at: "<<hotel->getCloseTime(SD)<<endl;
		SD++;
			if (SD > 6){
				SD = 0;
```

```
                    }
            }
            cout<<"\nScore: "<<maxScore<<endl;
            cout<<"Score: "<<calculateScore(M,finalDromologia)<<endl;


            delete hotel;
            return 0;


}
```

```
anthony@anthonysof-KuBeast: ~/data_structures/ttdp 84x63
anthony@anthonysof-KuBeast:~/data_structures/ttdp$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 16.04.4 LTS
Release:        16.04
Codename:       xenial
anthony@anthonysof-KuBeast:~/data_structures/ttdp$ ./ergasia.exe 5000 0.5 t103.txt
Start of tour: 510
----------FINAL-----------

Dromologio No: 1
The contents are:
Id: 0 Score: 0
->Id: 79 Score: 18
->Id: 61 Score: 6
->Id: 14 Score: 37
->Id: 6 Score: 21
->Id: 17 Score: 8
->Id: 5 Score: 3
->Id: 38 Score: 48
->Id: 33 Score: 8
->Id: 49 Score: 46
->Id: 125 Score: 8
->Id: 132 Score: 2
->Id: 77 Score: 10
->Id: 70 Score: 3
->Id: 9 Score: 9
->Id: 0 Score: 0
->
End of tour: 1175.25
Score: 227
Hotel closes at: 1210

Dromologio No: 2
The contents are:
Id: 0 Score: 0
->Id: 3 Score: 37
->Id: 10 Score: 9
->Id: 1 Score: 43
->Id: 11 Score: 38
->Id: 15 Score: 44
->Id: 12 Score: 8
->Id: 82 Score: 8
->Id: 21 Score: 3
->Id: 43 Score: 8
->Id: 47 Score: 19
->Id: 45 Score: 2
->Id: 46 Score: 8
->Id: 2 Score: 39
->Id: 0 Score: 0
->
End of tour: 1169.18
Score: 266
Hotel closes at: 1210

maxScore: 493
Final Dromologia Score: 493
anthony@anthonysof-KuBeast:~/data_structures/ttdp$
```