DIPARTIMENTO D'INGEGNERIA ELETTRICA E DELLE TECNOLOGIE DELL'INFORMAZIONE

CORSO DI LAUREA IN INFORMATICA

Gruppo 25

Giuliano Galloppi N86001508

Piero Junior Gaetani N86002210

Luigi Audino N86001513

Corso di Laboratorio di Algoritmi e Strutture dati gr.2- Prof. A. Murano

Progetto II: Compagnia Aerea – A.A. 2019/2020

Progettazione di un programma per la gestione delle prenotazioni di voli.

1. Approccio alla soluzione.

Per svolgere le richieste della traccia si è ipotizzato uno scenario simile alla prenotazione di un volo fatta su un qualsiasi applicativo che oggigiorno lo permetta. Le credenziali di accesso come utente/admin sono:
silvia@mail.it password utente:123 , password admin: admin.
I possibili utenti che utilizzano il programma sono 'Admin' ed 'Utenti semplici', i quali potranno accedere al programma dalla prima schermata facendo il login se già presenti nel sistema oppure registrandosi ad esso. Accedendo al sistema, l'admin potrà aggiungere e/o rimuovere destinazioni e voli. L'utente, invece, potrà effettuare una nuova prenotazione di un viaggio, visualizzare le prenotazioni effettuate in precedenza o visualizzare i suoi 'punti sconto'. I punti vengono accumulati effettuando una prenotazione di un volo e possono essere utilizzati per risparmiare su una prenotazione. Il sistema è già popolato con 20 destinazioni e tratte come da traccia. Quando l'utente effettuerà una nuova prenotazione, potrà decidere, da una lista delle città che sono presenti nel sistema, partenza e destinazione o solo la partenza, in quel caso il sistema proporrà una destinazione secondo i seguenti criteri:
Meta più economica: è la città di destinazione tra quelle che hanno un volo diretto con la città di partenza, ed è scelta per il costo inferiore a qualunque altra.

Meta più gettonata: è scelta tra tutte le città che hanno una tratta con la città di partenza, scali inclusi, di conseguenza la città scelta ha il massimo 'punteggio di città gettonata', punteggio che ogni città possiede.
Una volta decisa la destinazione il sistema chiederà all'utente di scegliere la tratta più economica o più breve, le quali saranno calcolate mediante *l'algoritmo di Dijkstra*, ed infine potrà confermare la prenotazione dopo che gli sarà stato riepilogato il prezzo finale con annesso uso di sconti o meno.
All'ultima pagina è riportato un disegno del grafo già popolato nel sistema.

2. Scelte implementative delle strutture dati

Le strutture dati scelte sono le seguenti:

- **Graph**: per la gestione delle città e delle tratte è stata utilizzata la struttura di grafo pesato orientato con un'implementazione a liste d'adiacenza. Oltre a quest'ultima ed il numero di nodi che le appartengono, il grafo presenta anche un array dinamico della struttura per i vertici denominato *infoVertex* che contiene le informazioni relative ad ogni vertice del grafo. Siccome un vertice necessita di più informazioni oltre ad una semplice chiave intera, questo infatti è stato

definito come una struttura '*Vertex'* contenente chiave, nome, il punteggio di città più gettonata e, per l'algoritmo di Dijkstra, un campo distanza dalla sorgente e predecessore.
Gli archi invece presentano due pesi 'km' e 'price' che indicano rispettivamente la distanza dell'arco ed il prezzo dell'arco, l'arco in questo caso è inteso come la tratta della traccia.

- **User:** struttura dati che memorizza le informazioni di un singolo utente o admin, quali nome, cognome, email, password, ruolo e punti.

- **UserTree:** la struttura dati utilizzata per salvare gli utenti/admin è un Albero Binario di Ricerca, in cui gli admin sono memorizzati nel sottoalbero destro, mentre, gli utenti semplici, nel sottoalbero sinistro.
  Per tale ragione, il numero di passi massimi necessari per ricercare un determinato utente nell' albero è N/2 + 1

- **City:** è una LinkedList, struttura utilizzata per memorizzare tutte le città rappresentanti le possibili mete di un viaggio.

- **Booking:** è una LinkedList, in cui ogni nodo corrisponde ad una prenotazione di un viaggio. In particolare, ogni nodo conterrà il prezzo e la tratta rappresentata dalla struttura City.

- **UserBooking:** è la struttura che associa una lista di prenotazioni ad un singolo utente.

2.1 Descrizione delle funzioni principali.

2.1.1 ***– Login***

La funzione **login** permette di identificare l'utente consentendogli o meno di accedere alle funzionalità esposte dal programma.

L'utilizzatore può essere "User" o "Admin", qualora l'esito del login fosse positivo, in base al ruolo con cui accede, gli verrà mostrato il relativo menu con il quale può interagire con le relative procedure del programma.

Semmai le credenziali di accesso fossero errate, verrà visualizzato un messaggio di errore e potrà provare nuovamente ad accedere, registrarsi o uscire dal programma.

2.1.2 – ***Registrazione***

La funzione **signIn** permette di aggiungere un utente o un admin al sistema, effettuando i seguenti controlli:

- L’utente o l’admin non siano già presenti.
- L’email inserita rispetta la reg-express.
- La password e conferma password siano uguali.

2.1.3 – ***Prenotazione***

- Tratta più economica e Tratta più gettonata.

Qui viene eseguito il metodo **bookingCheaperOrShortestPath**, il quale chiede all’utente quale tratta preferisce per il volo che sta per prenotare, successivamente, in base alla scelta, viene chiamato il metodo **dijkstraCheaper** per la tratta più economica o **dijkstraShortestDistance** per la tratta più breve. Il metodo **getSP** poi provvederà a salvare la tratta scelta, sfruttando a ritroso i predecessori della città di destinazione in una lista di città e **getFlyCost** ugualmente salverà il costo dell’intera tratta in una variabile intera.

Subito dopo, il metodo **summaryPurchase** effettuerà i controlli ed i calcoli relativi ai costi della tratta, ovvero, l’acquisizione di nuovi punti a favore dell’utente in seguito alla conferma di voler prenotare in base alla somma totale spesa per il viaggio fratto 2. Permette, qualora l’utente ne avesse a disposizione, la spesa di quest’ultimi in cambio di uno sconto corrispondente (1punto = 1euro) da applicare alla cifra di acquisto di prenotazione. All’utente viene sempre proposta la possibilità di non utilizzare tali punti e conservarli per una occasione successiva, infatti il programma prima verifica se chi sta interagendo ha a disposizione almeno un punto, di seguito domanda se vi è la volontà di utilizzarli ed infine mostra a schermo un riepilogo comprensivo di prezzo finale con lo sconto applicato qualora precedentemente l’utente abbia risposto affermativamente all’utilizzo dei propri punti accumulati.

Infine se l’utente conferma l’acquisto della prenotazione, essa verrà aggiunta ad una lista globale **listUserBooking** che contiene gli utenti e tutte le relative prenotazioni di quest’ultimi.

- Meta più economica e meta più gettonata

Qui vengono eseguiti a seconda della scelta dell’utente il metodo **destinationCheaper** per la meta più economica o **destinationMostPopular** per la meta più gettonata.

Il metodo destinationCheaper controlla nella lista d’adiacenza della città di partenza se la città di destinazione ha il prezzo della tratta minore, ritornandola.

Il metodo destinationMostPopular invece utilizza il metodo **existLinkDfs** (che

consiste in una visita del grafo in DFS per controllare se esiste una tratta tra la città di partenza e quella di destinazione) per verificare se la città di partenza ha una tratta confrontandola con ogni altra città del grafo, in tal caso viene salvata la città più gettonata. Nel caso in cui si verificasse che più di una città abbia lo stesso punteggio sarà discrezione dell'utente la scelta.
Infine, verrà richiamato nuovamente il metodo **bookingCheaperOrShortestPath** che ultimerà la prenotazione.

2.1.4 – ***Visualizzazione prenotazioni***

Qui verrà richiamato il metodo **printBooking**, il cui compito è visualizzare tutte le prenotazioni dell'utente, qualora esse esistano.

2.1.5 – ***Visualizzazione punti utente***

Qui l'utente può controllare il suo saldo punti.

3. Suddivisione del lavoro:
   Cooperazione sulla totalità del progetto utilizzando la piattaforma Teams e Github.
   In particolare:

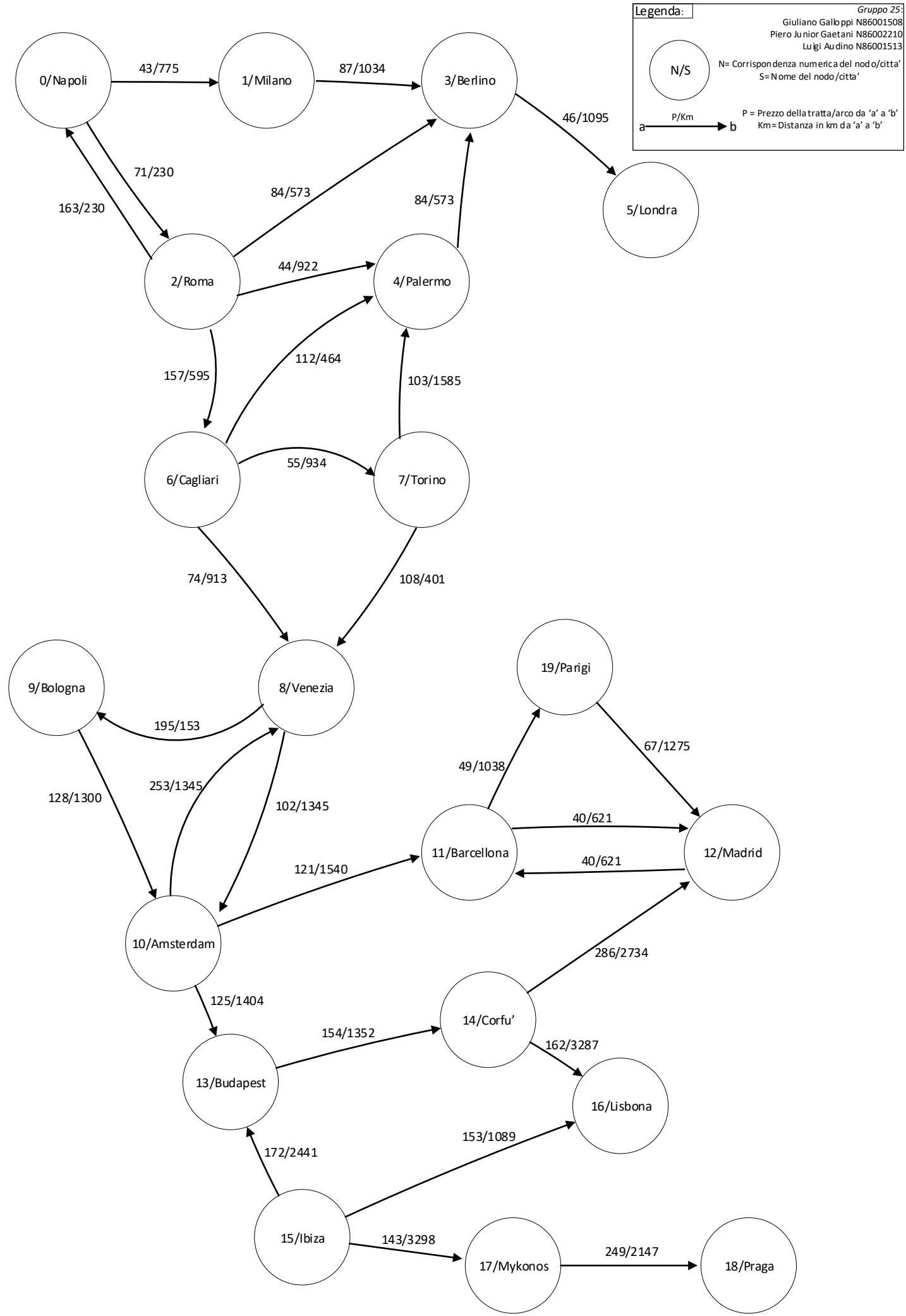
Luigi Audino si è occupato dell'autenticazione al sistema, strutturazione dell'utente, della memorizzazione ed aggiornamento delle prenotazioni effettuate dagli utenti, supporto al grafo.
Giuliano Galloppi si è occupato dell'implementazione della struttura dei vertici e della struttura base del grafo, del controllo tramite DFS dell'esistenza della tratta, del metodo per il criterio della meta più gettonata, e funzioni relax.

Piero Junior Gaetani si è occupato della navigazione del sistema, del criterio di aggiornamento punti utente, supporto al grafo ed all'utente.

Tutti i membri hanno lavorato sui due algoritmi di Dijkstra.

4. Esempio d'esecuzione: vedere cartella allegata.

Repository progetto: https://github.com/luigiAudino/Progetto2Gruppo25


Legenda:
Gruppo 25:
Giuliano Galloppi N86001508
Piero Junior Gaetani N86001210
Luigi Audino N86001513
N/S
N= Corrispondenza numerica del nodo/citta'
S= Nome del nodo/citta'
P/Km
a
b
P = Prezzo della tratta/arco da 'a' a 'b'
Km = Distanza in km da 'a' a 'b'
0/Napoli
1/Milano
3/Berlino
5/Londra
2/Roma
4/Palermo
6/Cagliari
7/Torino
9/Bologna
8/Venezia
19/Parigi
11/Barcellona
12/Madrid
10/Amsterdam
14/Corfu'
13/Budapest
16/Lisbona
15/Ibiza
17/Mykonos
18/Praga
43/775
87/1034
46/1095
71/230
163/230
84/573
84/573
44/922
157/595
112/464
103/1585
55/934
74/913
108/401
195/153
128/1300
253/1345
102/1345
49/1038
67/1275
40/621
40/621
121/1540
286/2734
125/1404
154/1352
162/3287
153/1089
172/2441
143/3298
249/2147