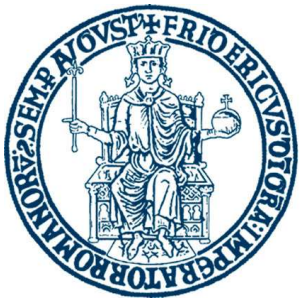


UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

LABORATORIO DI ALGORITMI E STRUTTURE DATI

3G TRAVEL

SVILUPPO DI UN SOFTWARE PER LA PRENOTAZIONE DI VIAGGI

IDENTIFICATIVO GRUPPO : 15

GIOVANNI ZAMPETTI
GIANLUCA PERNA
GIANMARCO LEMBO

INDICE :

1 DESCRIZIONE

Si svilupperà un software per la prenotazione di viaggi, dove un utente dopo essersi registrato può accedere al sistema e prenotare un viaggio specificando la partenza e la destinazione da una lista di mete. Il sistema offre diverse opzioni viaggio, si può scegliere se viaggiare in aereo o in treno o in entrambi i modi laddove fosse presente l’opzione e per ciascuna opzione viene proposta la scelta più economica o quella più veloce. Una volta selezionata l’opzione di viaggio il sistema propone una lista di hotel disponibili nella città selezionata. Quando l’utente ne sceglie uno, verrà proposto il modo per arrivare più velocemente dall’aeroporto o stazione all’hotel. Qualora la destinazione non fosse raggiungibile in alcun modo, il sistema notificherà al prossimo accesso lato admin, se si voglia rendere quella meta raggiungibile o eliminarla.

2 SVILUPPO DEL SOFTWARE

Per lo sviluppo si è scelto di implementare il software con strutture dati quali grafi, file, alberi, liste e code. Per ciò che riguarda gli utenti, si è deciso di implementare delle liste in quanto grafi e alberi già utilizzati. Al momento della registrazione dell’utente verrà scritto su un file, e al momento del login verranno effettuati dei controlli da file se i dati inseriti sono corretti.
Il grafo principale delle città è rappresentato con un albero binario di ricerca ordinato per il nome della città, ogni nodo dell'albero ha un puntatore alla città ed ogni città contiene delle liste di adiacenza (archi). In particolare ci sono due liste di adiacenza, una per le città raggiungibili tramite aereo e una per le città raggiungibili tramite

treno. Un elemento di una lista di adiacenza contiene un puntatore alla città raggiungibile, il prezzo e la durata in minuti del viaggio.
Per ogni città abbiamo un ulteriore grafo rappresentato in un vettore che indica tutti gli hotel di quella città, con i rispettivi prezzi. Inoltre gli archi del grafo degli hotel contengono le rispettive distanze.
Durante l'esecuzione del programma, sono stati implementati degli algoritmi per calcolare i percorsi minimi, in quanto viene richiesto di stampare il percorso più veloce per arrivare all'hotel desiderato. Per il grafo principale delle città si è scelto di implementare una struttura di supporto per l'implementazione del Dijkstra che in seguito tratteremo nel dettaglio.
Abbiamo poi una struct prenotazione, che ci mantiene tutti i dati relativi a quella medesima prenotazione, quali il prezzo totale di hotel e viaggio, la durata del viaggio ed una lista di nodi che rappresentano il percorso minimo calcolato.
Per quanto riguarda il funzionamento del software, all'inizio del programma, viene caricato tutto nelle varie strutture, e alla fine, dopo le eventuali modifiche si riscriverà sul file. Per quanto riguarda gli utenti, vengono caricati tutti i dati in una lista all'inizio dell'esecuzione, e se ci dovesse essere una modifica al saldo, viene modificata la lista e riscritta su file alla fine del programma.

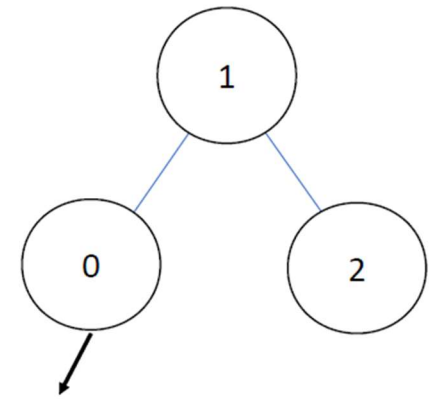
2.1 LATO ADMIN

Per quanto riguarda l'admin, può accedere al software senza registrarsi digitando una key segreta definita in seguito. È stata una ricerca lungo il grafo che crea una coda di notifiche che avvisano l'admin della presenza di nodi non raggiungibili all'interno del grafo delle città. A quel punto l'admin può decidere se rendere raggiungibile quella meta, attraverso dei collegamenti nel grafo, oppure eliminarla. Viene anche accodata una notifica anche nel momento in cui un utente sceglie di viaggiare verso una meta non raggiungibile.

2.2 LATO USER

Per quanto riguarda l'utente, si può decidere se effettuare un'operazione di registrazione o di accesso, e il programma provvederà a tutti i controlli del caso. Dopo la registrazione non c'è bisogno di effettuare di nuovo il login, ma si è già dentro al sistema, e si può scegliere di visionare le mete disponibili o di ricaricare il conto. Se si decide di visionare le mete se ne può scegliere una partenza e una di destinazione; verrà poi chiesto all'utente come desidera viaggiare, se in aereo o in treno o con entrambi laddove fosse disponibile. Dopo aver effettuato questa scelta, si può scegliere tra due opzioni, quella più economica e quella più veloce. Successivamente a quest'ultima, vengono visionati tutti gli hotel della meta di destinazione e una volta scelto l'hotel, viene proposto la soluzione più veloce per recarsi all'hotel dall'aeroporto o dalla stazione. Una volta scelto tutto, si può decidere se prenotare il viaggio oppure annullare. Verranno poi stampati i dettagli della prenotazione.

2.3 IMPLEMENTAZIONE DIJKSTRA SUL GRAFO DELLE CITTA'

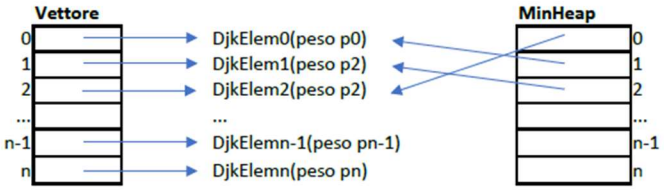



```
struct città {
   key
   nome
     ...
   ListaAdiacenzaTreni
   ListaAdiacenzaAerei
}
```

Per l'implementazione dell'algoritmo di Dijkstra sul grafo delle città è stata usata una struttura di supporto DjkElem contenente un puntatore alla città, un peso inizializzato a -1 (infinito), e un puntatore al precedente:

```
struct DjkElem {
   città*
   peso p
   DjkElem* precedente
}
```

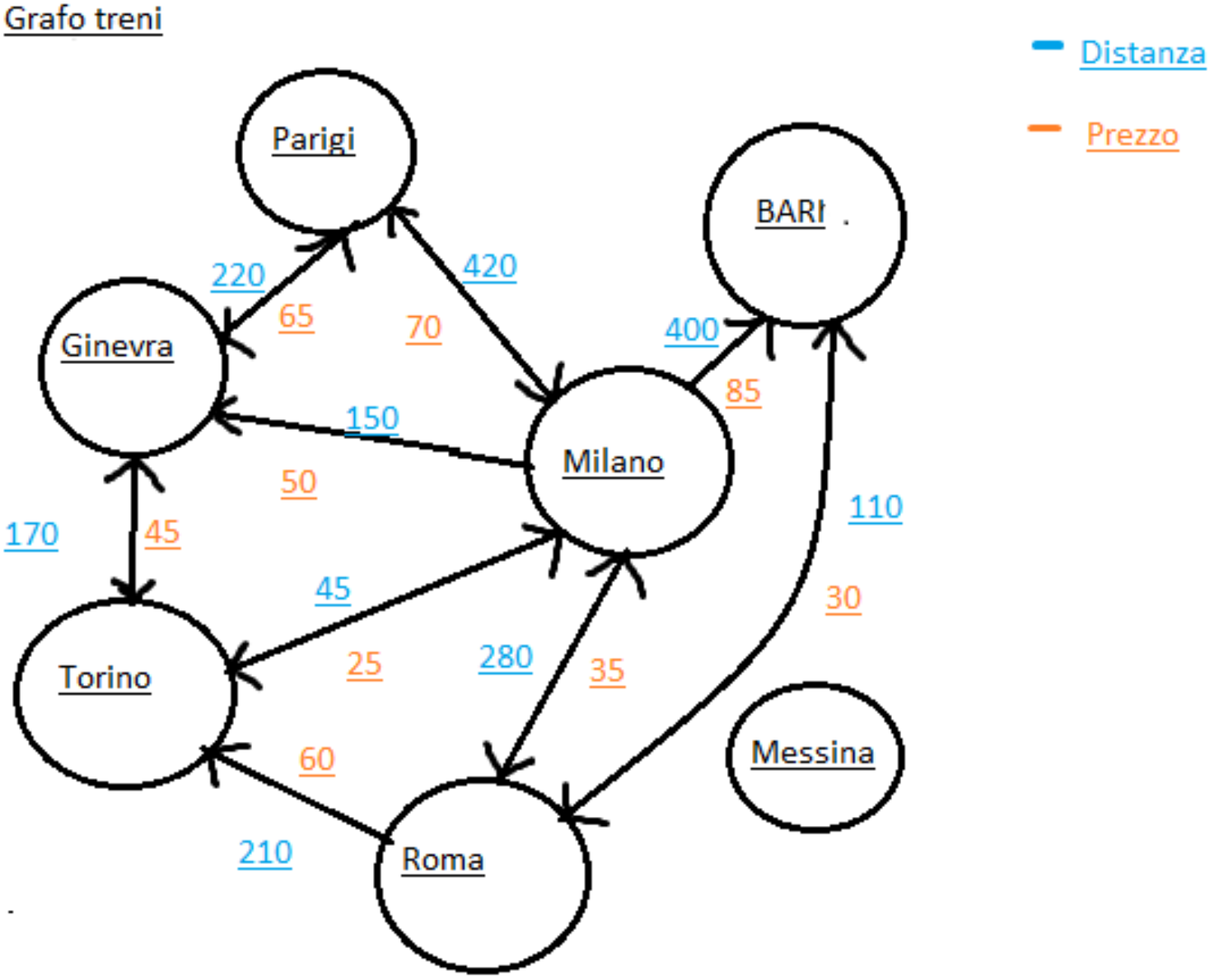
Di questa struttura sono stati allocati due vettori di questo tipo:



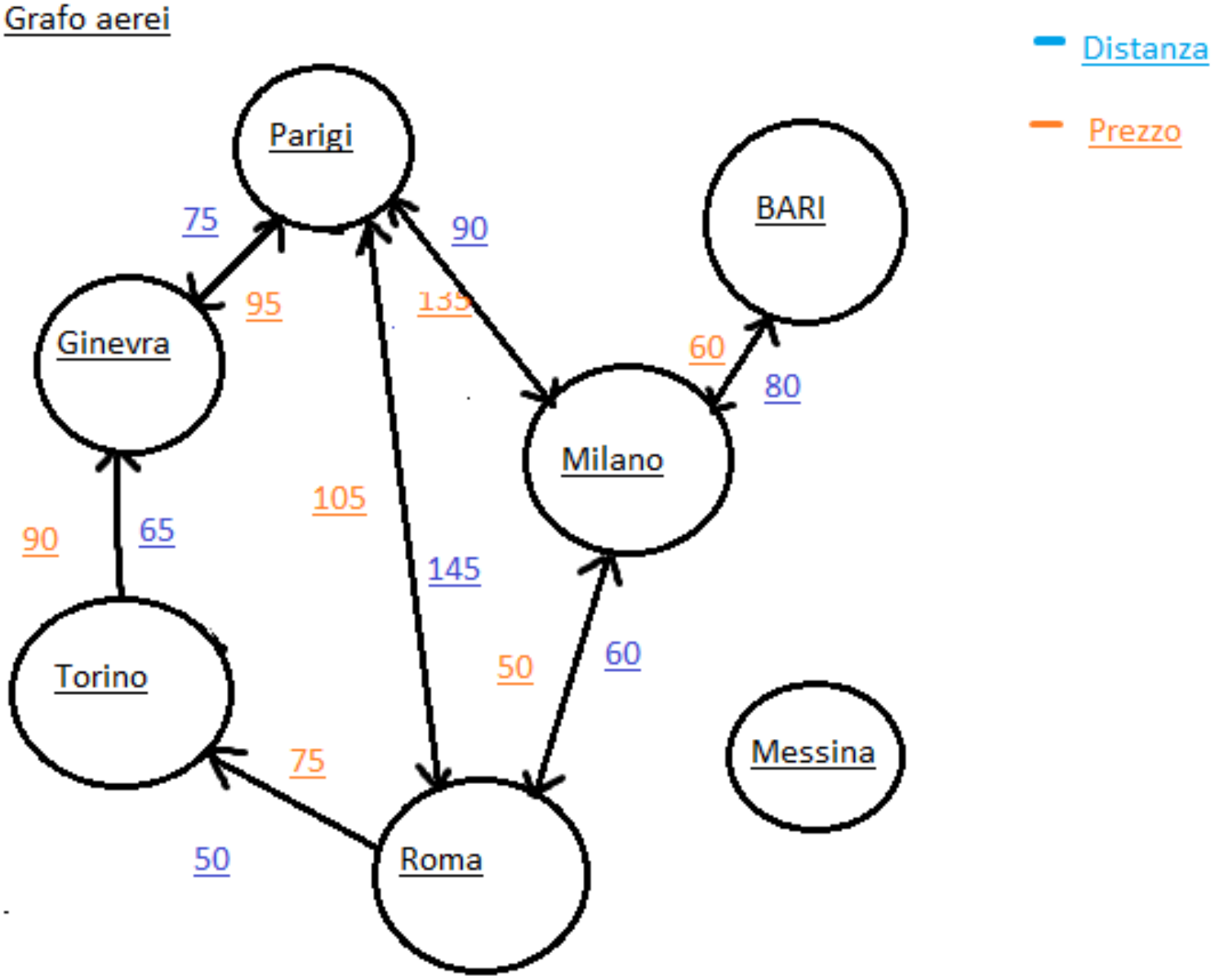
Per mantenere sia il vantaggio dell'accesso diretto sia la ricerca del minimo in log(n) si accede alle liste di adiacenza tramite il vettore (prendendo la key delle città adiacenti) per il rilassamento dei nodi, mentre l'estrazione del minimo viene effettuata sull'Heap.

Il peso da considerare varierà se si è scelto il percorso più veloce o quello più economico, così come le liste di adiacenza, da considerare a seconda di quale modalità di viaggio si è scelto (treno, aereo o mista).

Grafo treni


Distanza
Prezzo
Parigi
BARI
Ginevra
Milano
Torino
Messina
Roma
220
65
420
70
400
85
150
50
170
45
110
30
45
25
280
35
60
210

Grafo aerei


Distanza
Prezzo
Parigi
BARI
Ginevra
Milano
Torino
Messina
Roma
75
90
95
135
60
80
105
90
65
145
50
60
75
50