Corso di Algoritmi e Strutture Dati
Esercitazione 4: heap binari come code con priorità

Scopo di questa esercitazione è di scrivere un programma che simuli, in maniera semplificata, il movimento di N automezzi lungo la rete stradale segnalando gli istanti di tempo in cui ciascun automezzo transita attraverso le varie città.

Supponiamo di avere un programma che, data una mappa stradale e un insieme di veicoli identificati dalle rispettive targhe, calcola i tragitti che ciascun veicolo segue per recarsi dalla propria città di partenza alla propria città di destinazione. Un tragitto è una sequenza di tratte stradali, ciascuna con la propria lunghezza. Dunque, per ciascun veicolo il programma visualizza una successione di tratte stradali che attraversa varie città, e per ciascuna tratta visualizza la lunghezza in km.

Tale programma non è in realtà disponibile (sarà oggetto della successiva esercitazione), ma viene qui indicato il formato dell'output di quel programma, mediante un esempio. L'esempio si basa su una mappa stradale della Romania (fig. 1) e mostra i tragitti di due veicoli: il primo, targato `DK11702`, che segue il cammino Arad-Zerind-Oradea-Sibiu; e l'altro, targato `CT14700`, che segue il cammino Timisoara-Lugoj-Mehadia-Dobreta. Nell'ooutput risultano evidenti le targhe dei veicoli e i rispettivi tragitti, con indicazione delle città attraversate e dei chilometri tra una città e la successiva.

```
DK11702 Arad 75 Zerind 71 Oradea 151 Sibiu
CT14700 Timisoara 111 Lugoj 70 Mehadia 75 Dobreta
```

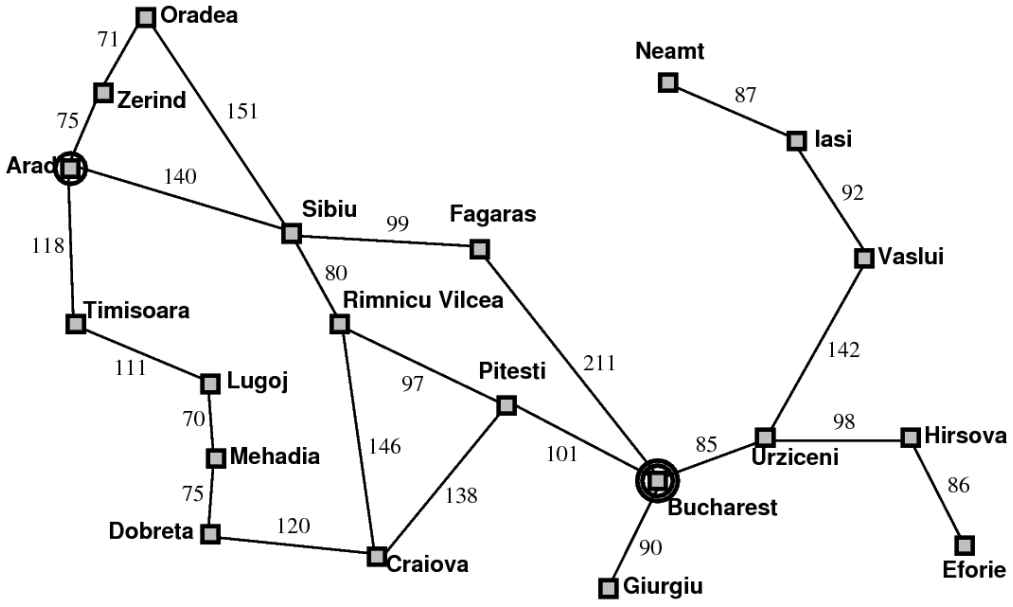

Figure 1: Principali città della Romania e collegamenti tra esse.

Ciascun tragitto, da solo, mostra le tappe di un veicolo ma non mostra lo svolgimento temporale dei suoi movimenti. Tuttavia, conoscendo le lunghezze delle tratte e impostando la velocità media del veicolo è possibile calcolare in quali istanti di tempo il veicolo raggiunge le varie città del cammino.

Per esempio, consideriamo il veicolo targato `DK11702` che segue il tragitto Arad-Zerind-Oradea-Sibiu. Se si muove alla velocità media di 60 km/h, tale veicolo si troverà ad Arad all'istante t=0, a Zerind all'istante t=4500, a Oradea a t=8760, e infine a Sibiu a t=17820 (tempi in secondi, arrotondati). Sempre nel nostro esempio, se il veicolo targato `CT14700` inizia il suo viaggio nello stesso istante del veicolo precedente (al tempo t=0) e si muove alla velocità media di 70 km/h lungo il tragitto Timisoara-Lugoj-Mehadia-Dobreta, sarà a Timisoara a t=0, a Lugoj a t=5709, a Mehadia a t=9309 e infine a Dobreta a t=13166.

Se si desidera seguire il movimento congiunto dei veicoli sulla rete stradale mostrando le varie tappe *nell'ordine temporale con cui esse vengono raggiunte*, occorre un programma che acquisisca in ingresso i tragitti e le velocità dei veicoli e mostri in uscita, nel nostro esempio, qualcosa del genere:

```
DK11702  0  Arad
CT14700  0  Timisoara
DK11702  4500  Zerind
CT14700  5709  Lugoj
DK11702  8760  Oradea
CT14700  9309  Mehadia
CT14700  13166  Dobreta
DK11702  17820  Sibiu
```

Naturalmente si può estendere al caso di N veicoli. Le informazioni sul movimento, qui rappresentate in modo testuale, possono essere inviate a una apposita interfaccia grafica per la visualizzazione effettiva sulla mappa stradale, ma ciò esula dagli scopi di questa esercitazione.

Ci sono vari modi per realizzare un tale programma di simulazione del movimento su strada, ma uno dei più semplici consiste nell'utilizzare una coda con priorità, realizzata come heap binario, e seguire l'approccio seguente:

- Si acquisisce da standard input il numero N di veicoli
- Per ciascun veicolo vengono acquisite targa, tragitto e velocità media in km/h
- Si estrae la prima tappa da ciascuno degli N tragitti, che deve coincidere con la città di partenza; la si marca con l'indicazione temporale (timestamp) `t=0`, e si inserisce nello heap; abbiamo così N elementi nello heap, tutti con timestamp `t=0`.
- Finché lo heap non risulta vuoto, si itera nel modo seguente: si estrae dallo heap l'elemento E con timestamp minimo e lo si elimina dallo heap, si deduce il veicolo V al quale tale elemento fa riferimento, si visualizza in output la targa del veicolo insieme al timestamp e alla tappa raggiunta (vedi esempio sopra), si calcola la successiva tappa di V (se tale tappa esiste) e il tempo T che occorre per raggiungerla, si somma T al timestamp di E creando così un nuovo elemento che va inserito nello heap.
- Quando lo heap è vuoto, tutti i veicoli hanno raggiunto le loro destinazioni e la simulazione è terminata. Si noti che, con questo approccio, la dimensione massima dello heap non è mai superiore a N.

Si realizzi un programma in C++ di simulazione di movimento su strada seguendo l'approccio suggerito sopra. Lo heap binario **deve essere realizzato mediante un array** e deve essere opportunamente incapsulato in una struct apposita che esporti solo i metodi strettamente necessari. Nei vari elementi dell'array possono essere inserite tutte le informazioni necessarie, tra cui ovviamente il timestamp che ha il ruolo di chiave, ma **non è consentito inserire o comunque fare**

uso di alcun puntatore o indice che faccia riferimento ad altri elementi dell'array per stabilire le relazioni tra padri e figli. Si ricorda che rappresentare un albero binario completo (che poi è lo "scheletro" dello heap) in un array è molto semplice: i figli dell'elemento di posto i si trovano infatti nelle posizioni $2i + 1$ e $2i + 2$, mentre il padre (se esiste) si trova in posizione $\lfloor (i-1)/2 \rfloor$. Non è dunque necessario usare puntatori o indici che riferiscano ai figli o al padre.

L'acquisizione dei dati di ciascun veicolo deve rispettare il formato
`targa tragitto velocità [invio]`; nel nostro esempio:

```
DK11702 Arad 75 Zerind 71 Oradea 151 Sibiu 60
CT14700 Timisoara 111 Lugoj 70 Mehadia 75 Dobreta 70
```

Dunque, per ogni riga di input, la stringa iniziale rappresenta la targa; l'ultimo numero in fondo rappresenta la velocità in km/h; e ciò che sta in mezzo rappresenta il tragitto, con le varie città e le lunghezze in km delle tratte stradali.