Progetti di Algoritmi e Strutture Dati—modulo 2

Anno Accademico 2016/2017

Gianluigi Zavattaro, Luca Bedogni

Esercizio 1 – Il Ladro Ingordo

Un ladro ingordo irrompe in un appartamento contenente oggetti di diverso valore, presenti nelle varie stanze di cui è composto. Il ladro attraversa tutte le stanze in una predefinita sequenza e può rubare tutti gli oggetti che vuole, a patto che l'oggetto da rubare sia di valore non inferiore all'ultimo che ha già scelto. Assumiamo che il valore sia indicato da un numero intero positivo. Scopo dell'esercizio è aiutare il ladro ingordo a trovare una sequenza di oggetti da rubare tale per cui il valore totale rubato sia massimo.

Il file di input contiene una prima riga con alcuni numeri interi: un primo numero n che indica il numero di stanze dell'appartamento seguito da n numeri N_1, N_2,…, N_n indicanti il numero di oggetti presenti nelle n stanze. Dopo la prima riga ci sono $N_1+N_2+\ldots+N_n$ righe ognuna delle quali contiene un unico numero intero che indica il valore di un oggetto. Gli oggetti sono elencati in base alla loro stanza di appartenenza, cioè prima gli N_1 oggetti della prima stanza, poi gli N_2 oggetti della seconda, e così via. Ecco un esempio di file di input:

```
3 4 6 3
12
6
1
3
14
45
12
32
1
12
4
45
13
```

Il file di output deve essere formattato in tante linee contenenti, una dopo l'altra, indicazione delle azioni che il ladro deve compiere: più precisamente, tali righe contengono o un numero intero indicante il valore dell'oggetto da rubare nella stanza in cui il ladro si trova, o il simbolo # indicante il fatto che il ladro deve passare alla stanza successiva. Assumiamo che all'inizio il ladro sia nella stanza *1*. Ecco un esempio di file di output relativo al precedente input:

```
1
3
6
12
#
12
12
14
32
45
#
45
```

Esercizio 2 – Il medico premuroso

In uno studio medico i pazienti arrivano tutti all'orario di apertura. Al medico premuroso basta uno sguardo per capire quanto tempo impiegherà a visitare ogni paziente e l'urgenza della visita. Scopo dell'esercizio è trovare un algoritmo efficiente che indichi al medico l'ordine in cui servire i pazienti in modo tale da minimizzare il tempo medio di attesa, servendo prima i pazienti più urgenti. Assumiamo che le priorità siano interi positivi (i pazienti con priorità maggiore devono essere serviti prima di pazienti con priorità inferiore), mentre i tempi di attesa sono numeri reali positivi.

Il file di input inizia con una prima riga che contiene un intero indicante il numero *n* di pazienti presenti. A seguire ci sono *n* righe, una per ogni paziente: ogni riga contiene un intero positivo (la priorità del paziente) ed un reale positivo (il tempo che verrà impiegato per la visita di tale paziente). Assumiamo di identificare i pazienti con *1*, *2*, …, *n* seguendo l'ordine in cui vengono descritti nel file di input. Ecco un esempio di file di input:

```
6
5 10.76
2 8.2
5 7.34
3 9.21
5 17.98
4 13.2
```

Il file di output dovrà contenere l'elenco dei pazienti nell'ordine in cui dovranno essere serviti. I pazienti vengono identificati con i valori *1*, *2*, …, *n* come descritto sopra. Ecco un esempio di file di output relativo al precedente input:

```
3
1
5
6
4
2
```

Esercizio 3 – La spending review

Una città al fine di ridurre i costi di manutenzione per le tubature della propria rete idrica, decide di chiuderne alcune. Le attuali tubature sono rappresentate tramite un grafo connesso non orientato pesato: il peso di un arco rappresenta il costo di manutenzione per una tubatura che collega i punti della città rappresentati dai due nodi su cui incide tale arco. Bisogna calcolare il massimo risparmio di costi di manutenzione che si può ottenere chiudendo tubature, garantendo comunque che la rete idrica continui a collegare tutti i punti della città.

Il file di input contiene una riga iniziale che contiene due interi *n* ed *m*, rispettivamente numero di nodi e numero di archi del grafo che rappresenta la rete idrica. Il file contiene successivamente *m* righe, una per ogni arco, ognuna della quali contiene due numeri interi ed un numero reale non negativo: i due numeri interi indicano l'indice dei nodi su cui l'arco incide, mentre il numero reale indica il peso dell'arco. Assumiamo che i nodi siano identificati tramite i numeri *0*, *1*,.., *n-1*. Ecco un esempio di file di input:

```
9 14
0 1 4.0
0 7 8.0
1 2 8.0
1 7 11.0
2 3 7.0
2 5 4.0
2 8 2.0
3 4 9.0
3 5 14.0
4 5 10.0
5 6 2.0
6 7 1.0
6 8 6.0
7 8 7.0
```

Il file di output semplicemente riporta il massimo risparmio ottenibile. Ecco un esempio di file di output relativo al precedente input:

```
56.0
```

Esercizio 4 – Il turista itinerante

Un turista ha a disposizione *N* giorni di vacanze che userà per visitare vari luoghi rappresentati tramite un grafo: i nodi indicano i possibili luoghi da visitare e gli archi le possibili connessioni per spostarsi da un luogo ad un altro. Il turista deve partire da un dato nodo *i* e alla fine della vacanza raggiungere un altro dato nodo *j*. In ogni nodo visitato deve rimanere almeno un giorno. Assumiamo che gli eventuali spostamenti vengano effettuati di notte (cioè, quando decide di spostarsi da un luogo ad uno connesso, il turista non perde giorni di vacanze). Ogni luogo ha associato un valore che rappresenta la soddisfazione del turista nell'essere in quel luogo per un giorno. Quando il turista decide di spostarsi da un luogo *s* ad un luogo connesso *t*, vista la fatica nell'affrontare il trasferimento, la sua soddisfazione complessiva diminuisce di un valore associato all'arco che collega *s* a *t*. Se invece resta nello stesso luogo, la sua soddisfazione non si riduce in quanto non effettua alcun spostamento, ma rimanendo nello stesso luogo un ulteriore giorno la sua soddisfazione complessiva incrementa del valore associato al luogo in cui si trova. Lo scopo dell'esercizio è aiutare il turista a capire come partire dal nodo *i* e arrivare al nodo j, in modo tale da massimizzare il suo grado di soddisfazione.

Il file di input inizia con una prima riga che contiene 5 numeri interi: il numero *n* di nodi del grafo, il numero *m* di archi, il numero *N* di giorni di vacanza, il nodo di partenza *i* ed il nodo di arrivo *j*. Assumiamo di identificare i nodi con gli indici *0*, *1*, *2*,…, *n-1*. A seguire ci sono *n*+*m* righe. Le prima *n* righe contengono un numero intero che indica il grado di soddisfazione del relativo nodo (i gradi di soddisfazione vengono riportati in ordine di indice di nodo: prima il nodo *0*, poi *1*, eccetera). Nelle successive *m* righe sono descritti gli archi: ogni riga contiene tre numeri interi, i primi due indicanti i due nodi su cui incide l'arco, mentre il terzo indica la quantità di soddisfazione che viene persa facendo lo spostamento rappresentato da tale arco. Ecco un esempio di file di input:

```
4 5 30 0 3
2
5
10
4
0 1 10
0 2 10
1 2 10
1 3 10
2 3 10
```

L'output atteso deve contenere la sequenza di luoghi visitati che massimizza il grado di soddisfazione del turista. I nodi visitati vengono riportati ognuno su una propria riga secondo l'ordine di visita. Ecco un esempio di file di output relativo al precedente input:

```
0
2
3
```

Poiché ovviamente il turista sceglierà di rimanere 30-2=28 giorni nel nodo 2, che dà una soddisfazione maggiore rispetto agli altri nodi, la soddisfazione finale di questo tipo di viaggio sarà come segue:
2 (nodo 0) + 4 (nodo 3) + (30-2)*10 (nodo 2) – 10 (costo da 0 a 2) – 10 (costo da 2 a 3) = 266