Esame di Algoritmi e Laboratorio - Modulo di Laboratorio

Appello del 9 aprile 2020

April 8, 2020

Note

Completata la prova, spedire il codice sorgente, costituito da un singolo file, all'indirizzo `santamaria@dmi.unict.it`. Prove inviate oltre l'orario concordato non saranno valutate.

Specifiche

Si implementi la classe Grafo supponendo che gli archi siano direzionati e che i nodi contengano chiavi di tipo generico. In seguito, si fornisca l'implementazione dell'algoritmo di Dijkstra con Heap in grado di calcolare il peso dei cammini minimi da una data sorgente verso tutti i nodi del grafo. Successivamente, si implementi una tabella Hash ad indirizzamento aperto con sondaggio lineare nella sua formulazione più semplice che supporti le operazioni di inserimento e ricerca. Quest'ultima deve fornire l'indice nella tabella del valore ricercato. Scrive una funzione che legga l'input e produca l'output come descritto sotto.

L'input è suddiviso in task, uno per ogni riga. Ogni riga del file di input è formata da N+M+5 elementi. I primi due elementi sono dei valori numerici interi, N ed M, i quali rappresentano il numero di nodi ed il numero di archi presenti nel grafo. Segue una stringa di testo che identifica il tipo di dato (int o double) che dovrà essere contenuto all'interno dei nodi del Grafo. Segue la sequenza delle chiavi relative agli N nodi, in ordine sparso, e la sequenza degli M archi del grafo (in ordine sparso). Ogni arco è rappresentato da tre elementi (le chiavi dei nodi sorgente e destinazione, nell'ordine, e il peso dell'arco) racchiusi tra parentesi. Gli ultimi due elementi sono la chiave del nodo sorgente e quella del nodo destinazione per i quali si vuole calcolare il cammino minimo.

L'output dovrà essere formato da una riga per ogni task. Ogni riga contiene due valori separati da uno spazio. Un valore intero che rappresenta la posizione nella tabella hash del valore della distanza di cammino minimo tra il nodo sorgente e in nodo destinazione del grafo considerato. La tabella hash deve avere dimensioni pari al numero di archi presente nel grafo. Il secondo valore rappresenta il numero di chiamate alla funzione *decreaseKey* della coda di priorità

impiegata nell'algoritmo di Dijkstra per il calcolo del cammino minimo. Nel caso in cui non esista alcun cammino tra i nodi indicati si dovrà considerare come valore di cammino minimo il valore "INT_MAX". La tabella hash deve avere dimensione pari al numero di archi presenti nel grafo considerato. Si noti che:

- N è sempre un valore inferiore a 100.
- Il peso degli archi è sempre minore o uguale a 100.
- Gli archi hanno tutti un peso non negativo.

Esempio

Il seguente esempio presenta un file di input, contenente 4 linee di input ed il corrispondente file di output (vedere file allegato).

5 11 int 33 48 50 72 34 (33 72 20) (33 34 36) (48 33 53) (48 50 56) (48 34 9) (50 48 93) (50 72 74) (50 34 98) (72 48 52) (72 34 28) (34 50 5) 48 72

5 11 int 54 91 88 58 63 (54 91 24) (54 88 43) (54 63 99) (91 88 26) (88 54 56) (88 91 69) (88 58 65) (88 63 13) (58 63 67) (63 91 42) (63 58 89) 58 54

5 9 int 72 19 57 55 56 (72 19 24) (72 55 80) (19 57 74) (57 72 18) (57 19 25) (57 56 89) (55 19 25) (56 57 49) (56 55 29) 56 19

6 20 double 2.4 7.7 4.7 1 6.7 6.8 (2.4 7.7 8) (2.4 1 20) (2.4 6.8 26) (7.7 4.7 31) (7.7 1 88) (7.7 6.7 21) (4.7 2.4 99) (4.7 7.7 30) (4.7 1 46) (4.7 6.7 60) (4.7 6.8 18) (1 7.7 65) (1 4.7 9) (1 6.7 87) (6.7 2.4 62) (6.7 7.7 49) (6.7 4.7 39) (6.7 1 93) (6.8 2.4 30) (6.8 4.7 15) 6.8 6.7

Il corrispondente file di output sarà:
7 6
4 4
0 4
19 8