Relazione 3/4 ASD

Mana Alessio, Perlo Giacomo, Sanino Fabrizio

Aprile 2021

1 Scelte implementative

Per la rappresentazione del grafo come descritto nel paragrafo 22.1[1] ci sono due modi per rappresentare i grafi: matrici di adiacenza o liste di adiacenza. Dovendo implementare una libreria per un grafo sparso abbiamo deciso di utilizzare le liste di adiacenza in modo tale da ottimizzare la memoria, cosa che non era possibile con la martice che ha una dimensione prefissata ($|Vertici|^2$). Le liste di adiacenza sono formate in questo modo: è presente un array nel quale gli indici sono i nodi, mentre l'elemento è una lista di archi rappresentati come coppia (nodo di destinazione, peso).

Nel caso teorico le complessità richieste dall'esercizio sono rispettate, in quanto i nodi sono rappresentati da un intero, utilizzabile quindi come indice nell'array. Nel caso pratico, invece, i nodi sono rappresentati con un tipo generico T non utilizzabile quindi come indice in un array. Ci giungono in soccorso le HashTable, array i cui indici sono rappresentabili come valori di tipi generici. Dunque la struttura attuale è una HashTable dove gli indici della quale sono i valori di tipo T dei nodi, mentre gli elementi sono liste di archi formati da coppie (nodo di destinazione, peso).

La struttura delle coppie (nodo, peso) è stata invece implementata usando la Pair, una struttura originaria di javafx che permette di creare coppie di tipi generici. In conclusione la struttura completa di un grafo è

```
grafo = HashTable<T, ArrayList<Pair<T, E>>
```

dove T è il tipo di un nodo, E il tipo di un peso.

Presentiamo l'analisi delle richieste di complessità più difficili da rispettare:

- Verifica se il grafo contiene un dato nodo – O(1): la struttura utilizzata permette di accedere in grafo[nodo] in una sola operazione, per verificarne l'esistenza.

- Verifica se il grafo contiene un dato arco – O(1): possiamo verificarne l'esistenza cercando all'interno della lista in grafo[nodo] il dato arco. Se il grafo è sparso la lista conterrà un solo elemento, dunque l'accesso richiederà una sola operazione.

- Recupero etichetta associata a una coppia di nodi – O(1): per il recupero andiamo ad accedere a grafo[nodo di partenza], cerchiamo nella lista il nodo di destinazione e ritorniamo il peso associato a quel determinato arco se lo troviamo. Anche in questo caso l'accesso richiederà una singola operazione se il grafo è realmente sparso.

1.1 Aggiornamento del 30/06/2021

Abbiamo deciso di apportare alcune piccole migliorie alla libreria *Graph* sugli archi di grafi non diretti:

- Metodo *GetArchsCount* ritornava il numero totale di archi, che in un grafo non diretto erano il doppio degli archi reali. Ora ritorna il numero corretto di archi dividendo il numero di archi per due.
- Metodo *GetArchs* ritornava tutti gli archi come coppie di nodi e, nel caso di grafi non diretti, ritornava archi doppi (esempio: ritornava $a \rightarrow b$ e $b \rightarrow a$). Ora gli archi vengono salvati in una matrice bidimensionale, temporanea per la ricerca, come coppie di nodi. Prima di aggiungere un arco alla lista da ritornare controlliamo se esiste già l'arco reciproco nella matrice ausiliaria e, se non esiste, lo aggiungiamo.

References

[1] Thomas H. Cormen. *Introduzione agli algoritmi e strutture dati*. McGraw-Hill, 2010.