

Progettazione di software sicuro online

Esame del 5 luglio 2013 – (Prova di laboratorio)

1 Undirected graph [pt. 2]

Scrivere una classe *UndirectedGraph* che rappresenti un grafo non orientato. La classe deve memorizzare gli archi usando una matrice booleana di incidenza di dimensione $n \times n$ (dove n è il numero di nodi): la cella (i, j) della matrice vale true se c'è un arco dal nodo i al nodo j (poiché l'arco non è orientato, anche (j, i) deve valere true).

Il costruttore della classe deve inizializzare la matrice di incidenza facendo in modo che questa rappresenti un grafo di n nodi senza archi: il valore n deve essere passato come parametro al costruttore.

Metodo addArc La classe deve disporre di un metodo *addArc(int nodeA, int nodeB)* che permetta di aggiungere archi al grafo; il metodo è mostrato nel Codice 1.

```
public boolean addArc(int nodeA, int nodeB) {
    if (nodeA >= 0 && nodeA < incidenceMatrix.length &&
        nodeB >= 0 && nodeB < incidenceMatrix.length) {
        incidenceMatrix[nodeA][nodeB] = true;
        incidenceMatrix[nodeB][nodeA] = true;
        return true;
    }
    else {
        return false;
    }
}
```

Codice 1: Metodo *addArc(int nodeA, int nodeB)*

✓ **Metodo cycle3** La classe deve anche disporre di un metodo booleano *cycle3* che dica se la il grafo contiene almeno un ciclo di lunghezza 3. Figura 1 contiene un esempio di ciclo di lunghezza 3.

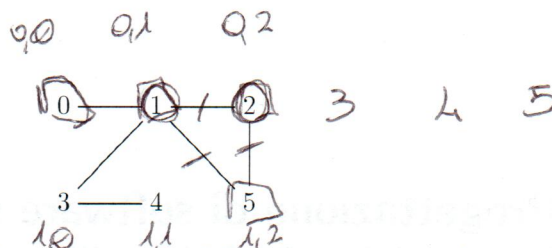


Figure 1: Grafo non orientato con ciclo di lunghezza 3 tra i nodi 1, 2 e 5

2 Junit

In JUnit, scrivere i seguenti casi di test nella classe *DirectedGraphTest*.

- Per il metodo *addArc*, scrivere:
 - ✓ – un caso di test che mostri che, se gli indici sono corretti, l'arco viene aggiunto (basandosi sul valore ritornato dal metodo); [pt. 1]
 - ✓ – un caso di test che mostri che, se gli indici non sono corretti, l'arco non viene aggiunto (basandosi sul valore ritornato dal metodo); [pt. 1]
- ✓ • scrivere un caso di test che mostri che un determinato grafo da voi costruito contiene un ciclo di lunghezza 3. [pt. 1]

✓ 3 Copertura

Scrivere in JUnit una test suite che soddisfi la copertura delle condizioni per il metodo *addArc*; mettere i test case nella classe *AddArcCoperturaCondizioni* e commentare in modo opportuno i singoli test case. [pt. 1]

4 JML

Scrivere in JML la seguente preconditione al costruttore:

- ✓ • il numero di nodi del grafo deve essere compreso tra 1 e 100. [pt. 0.25]

Scrivere in JML la seguente postcondizione al costruttore:

- ✓ • il grafo non contiene archi. [pt. 1]

Scrivere in JML la seguente preconditione al metodo *addArc(int nodeA, int nodeB)*:

- ✓ • non esiste un arco tra *nodeA* e *nodeB*. [pt. 1]

Scrivere in JML la seguente postcondizione al metodo *addArc(int nodeA, int nodeB)*:

- il numero di archi del grafo dopo l'esecuzione del metodo è uguale o al massimo maggiore di un'unità al numero di archi del grafo prima dell'esecuzione del metodo (attenzione che ogni arco viene rappresentato due volte). [pt. 1.75]

Totale punti = 2 + 3 + 1 + 4 = 10