Laboratorio di Algoritmi e Strutture dati*

8 Maggio 2015

⋆Esercizio Graphs – *(6pt)*

Creare una interfaccia Graph che definisce il tipo di dato astratto grafo. Il grafo deve essere definito su nodi di un tipo generico V. Gli archi del grafo hanno una informazione associata di tipo E. L'interfaccia Graph deve essere definita come segue:

```
public interface Graph<V, E> {
  /*
   * Aggiunge il vertice vertex al grafo se il vertice non
   * vi appartiene già. Restituisce true sse vertex è stato
   * aggiunto.
   */
  boolean addVertex(V vertex);

  /*
   * Aggiunge l'arco (vertex1,vertex2) al grafo se l'arco non
   * vi appartiene già. Restituisce true sse l'arco è stato aggiunto.
   */
  boolean addEdge(V vertex1, V vertex2, E data);
  /*
   * Restituisce true sse vertex fa parte del grafo.
   */
  boolean hasVertex(V vertex);
  /*
   * Restituisce true sse l'arco (vertex1,vertex2) fa
   * parte del grafo.
   */
  boolean hasEdge(V vertex1, V vertex2);


  /*
   * Restituisce il dato associato all'arco vertex1 -> vertex2
   */
  E getData(V vertex1, V vertex2);

  /*
   * Restituisce la lista dei vertici del grafo.
```

*Revisione n. 1.0 del 13 maggio 2015

```
    */
    Collection<V> getVertices();

    /*
     * Restituisce la lista di adiacenza del vertice vertex
     * (o null se il vertice non appartiene al grafo).
     */
    Collection<V> getNeighbors(V vertex);

}
```

Implementare quindi una a scelta tra le seguenti classi:

SparseGraph : implementa l'interfaccia Graph usando strutture dati appropriate per un grafo orientato sparso;

DenseGraph : implementa l'interaccia Graph usando strutture dati appropriate per un grafo orientato denso.

Si crei quindi un'interfaccia GraphSearch che dichiara il seguente metodo:

```
void search(Graph<V, E> graph, V source, SearchCallback<V,E> callback);
```

dove SearchCallback è un'interfaccia che dichiara i seguenti metodi:

```
void onVisitingVertex(V vertex);
void onTraversingEdge(V source, V dest, E info);
```

L'idea è che gli algoritmi che visitano il grafo implementino Search richiamando gli opportuni metodi dell'oggetto "callback" ogni volta che considerano un vertice o attraversano un arco. Gli "utenti" dell'algoritmo dovranno fornire una implementazione di SearchCallback utile ai loro scopi.

Realizzare le classi BreadthFirstSearch e DepthFirstSearch che implementano gli algoritmi di visita in ampiezza e in profondità e testarli sulla struttura dati che si è scelto di implementare. Entrambe le classi devono implementare l'interfaccia GraphSearch.