Esercitazione - Grafi

Fondamenti 2 - Corso di Laurea in Informatica

Facendo uso della classe Grafo e GrafoOrientato, scrivere un main implementando, e successivamente "testando", le seguenti funzioni:

- **void stampaGrafo(const Grafo& g)** che stampa tutte le informazioni relative al grafo g;
- **int getNodoConGradoMassimo(const GrafoNonOrientato& g)** che restituisce il nodo avente grado massimo all'interno del grafo non orientato g;
- **bool stessoNumeroNodiStessoGrado (const GrafoNonOrientato& g1, const GrafoNonOrientato& g2)** che restituisce *true* se e solo se g1 e g2 hanno lo stesso numero di nodi per ogni possibile grado.

  Esempio:
  g1: deg(0) = 0, deg(1) = 3, deg(2) = 3, deg(3) = 1;
  g2: deg(0) = 3, deg(1) = 3, deg(2) = 1, deg(3) = 0.

  La funzione dovrebbe restituire *true* in quanto il numero di nodi aventi grado 0 è pari a 1 per entrambi i grafi, il numero di nodo aventi grado 1 è pari a 1 per entrambi i grafi, il numero di nodi aventi grado 2 è pari a 0 per entrambi i grafi ed infine il numero di nodi aventi grado 3 è pari a 2 per entrambi i grafi.
- **bool almenoUnNodoAdiacenteATutti(const GrafoNonOrientato& g)** che restituisce *true* se e solo se esiste almeno un nodo in g che sia adiacente a tutti i rimanenti.
- **CoppiaNodi getCoppiaConPiuAdiacenti(const GrafoNonOrientato& g)** che restituisce la coppia di nodi che hanno il maggior numero di nodi adiacenti in comune.
  Si noti che tale coppia va restituita all'interno di un oggetto di tipo CoppiaNodi che deve rappresentare un pair<int,int>.
- **bool connesso(const GrafoNonOrientato& g)** che restituisce *true* se e solo se il grafo g è connesso.
  Si noti che, essendo g un grafo non orientato, non occorre controllare che per ogni coppia di nodi debba esistere un cammino che li unisce.

- **bool inUnCiclo(const Grafo& g, int nodo)** che restituisce *true* se e solo se il nodo *"nodo"* si trova in un ciclo all'interno di g.

- **bool proprieta_1(const Grafo& g, int k)** che restituisce *true* se e solo se vale la seguente proprietà per il grafo g: il numero di coppie di nodi che hanno almeno un adiacente in comune deve essere maggiore o uguale a k. La coppia generica (i, j) è considerata uguale alla coppia (j, i) e, dunque, va contata una sola volta.

- **bool proprieta_2(const Grafo& g, vector<int> pesi)**. Sapendo che pesi[i] rappresenta il peso del nodo i nel grafo g, la funzione deve restituire *true* se e solo se vale per g la seguente proprietà: per ogni nodo i, il peso di i moltiplicato per il numero di nodi adiacenti a i deve essere maggiore o uguale alla somma dei pesi dei nodi a lui adiacenti.

Esercizio: Facendo uso della classe GrafoPesato implementare l'algoritmo di Dijkstra per il calcolo del cammino minimo da un nodo x a tutti gli altri nodi del grafo. Stampare, inoltre, il cammino minimo dal nodo x ad un nodo qualsiasi a proprio piacimento.