ESERCIZI SUI GRAFI

1. Progettare un algoritmo di visita in ampiezza di un grafo il cui insieme di vertici non sia preventivamente noto e analizzarne la complessità.
   [Suggerimento: utilizzare un dizionario.]

2. Dato un grafo $G = (V, E)$ non orientato, progettare un algoritmo efficiente per stabilire se G è un albero.

3. Sia $G = (V, E)$ un grafo orientato, e siano x, y, z tre vertici di G. Stabilire se y si trova su un cammino da x verso z.

4. Sia $G = (V, E)$ un grafo memorizzato con liste di adiacenza. Progettare un algoritmo che trasformi G nel grafo G' che contiene la stessa informazione, ma è memorizzato con una matrice di adiacenza.

5. Sia $G = (V, E)$ un grafo connesso e non orientato. Progettare un algortimo che ricevuto in ingresso G e un suo vertice r, restituisca il numero di vertici che si trovano a distanza massima da r.

6. Un pozzo in un grafo orientato G è un vertice di grado uscente 0 e di grado entrante uguale a n-1, dove n è il numero di vertici del grafo. Si osservi che se esiste, il pozzo è unico. Scrivere una procedura in pseudocodice per trovare il pozzo in G, se esiste.

7. Dato un grafo orientato $G = (V, E)$ con n vertici e m archi, un vertice bersaglio $b \in V$ un sottoinsieme $W \subset V$ contenente k vertici, progettare un algoritmo per individuare il vertice di W con la minima distanza dal bersaglio b. L'algoritmo deve restituire uno qualunque dei vertici a distanza minima, oppure NIL se il bersaglio non è raggiungibile da nessun vertice di W. Si supponga di avere in input, oltre al grafo G rappresentato con liste di adiacenza, il bersaglio b e un array binario S tale che S[i] = 1 se il vertice i appartiene a W, 0 altrimenti.
   - Progettare un algoritmo di costo in tempo $T(n, m, k) = O(k(n + m))$.
   - Progettare un algoritmo di costo in tempo $T(n, m, k) = O(n + m)$ che risolve il problema con una sola visita.
     [Suggerimento: è ammessa la modifica del grafo G in input all'algoritmo]

8. Sia $G = (V, E)$ un grafo non orientato con n vertici e m archi che rappresenta una rete di comunicazione. Ciascun vertice può essere di uno tra tre tipi: *transmitter, receiver,* o *switch*. Il tipo del vertice v è indicato nell'attributo *v.type*. La rete rappresentata da G è considerata **valida** se ogni receiver è raggiungibile da almeno un transmitter.
   - Descrivere (in pseudocodice) un algoritmo che determini se G rappresenta una rete valida
   - Analizzarne la complessità dell'algoritmo proposto. Eventuali modifiche agli algoritmi di visita possono essere descritte a parole, senza darne lo pseudocodice.