**1.** Un ricoprimento dei vertici di un grafo (**vertex cover**) è un sottinsieme di vertici C tale che per ogni arco del grafo almeno uno dei vertici adiacenti all'arco sia compreso in C. Il problema del minimo vertex cover è un tipico esempio di problema NP-hard.
Progettare la classe **MyGraph**, derivata da Graph, che estende la classe padre implementando i metodi **min_vertex_cover** e **greedy_vertex_cover**.
min_vertex_cover non prende input e restituisce un vertex cover di dimensione minima; greedy_vertex_cover non prende input e restituisce un vertex cover di dimensione al più doppia di quello minimo.
Valutare sperimentalmente le prestazioni dei due metodi su un campione di almeno ***k*** grafi (con ***k*** > 50) con ***n*** vertici (***n*** > 50) con scelti a caso. Per ciascun grafo fornire le dimensioni dei vertex cover restituiti dai due metodi ed i relativi tempi di esecuzione. Calcolare di quanto in media l'algoritmo greedy è più veloce rispetto all'algoritmo ottimo e di quanto il vertex cover restituito è più grande.

**2.** In un grafo non diretto e connesso si definisce bridge del grafo un arco la cui rimozione rende il grafo non connesso. Un grafo che non ha bridge è detto biconnesso. Implementare la funzione **bridge(G)** che, preso in input un oggetto MyGraph G, restituisce un bridge di G (None se il grafo è biconnesso).
Analizzare la complessità di tempo della soluzione proposta.

**3.** Il servizio di volanti della polizia prevede che una serie di pattuglie siano continuamente in giro sul territorio per poter intervenire in casi di emergenza. Quando giunge una richiesta di intervento, la centrale operativa individua le volanti che in quel momento possono intervenire nel minor tempo possibile e le allerta via radio.
Il ministro ha deciso di dotare le centrali operative di tutte le città di un sistema automatico, in funzione 24 ore al giorno, che registra le richieste di intervento, valuta il tipo di emergenza, decide quante volanti far intervenire ed individua le volanti che si trovano nella migliore posizione.
Implementare la funzione **emergency_call(G, pos, v, k)** che, preso in input un grafo diretto e connesso G che rappresenta la rete stradale cittadina (i vertici sono gli incroci e gli archi sono le strade), un dizionario pos che contiene la posizione delle volanti, il luogo v dove intervenire ed il numero k di volanti richieste per l'intervento, individua le volanti che devono essere allertate. Per semplicità si assuma che le volanti siano numerate da 1 a N, che stazionino sempre negli incroci e che il luogo dell'intervento sia un incrocio.
Analizzare la complessità di tempo della soluzione proposta.

La soluzione deve essere caricata sul sito del corso **entro il 7 gennaio**.
Il materiale consegnato deve essere costituito da un unico file compresso contenente:

- Un file pdf con la documentazione del progetto che deve contenere le analisi di tutte le soluzioni proposte.
- Un progetto Python chiamato #gruppo_4_TdP contenente
  - il package Python TdP_collections (con tutte le implementazioni delle strutture dati di supporto, eventualmente modificate);
  - il package Python pkg_i contenente le classi per la soluzione dell'esercizio i (i = 1, 2, 3);
  - script di testing per ciascun esercizio.

Nei prossimi giorni verranno caricati sul sito del corso dei data-set per testare la correttezza del progetto.