1. Dalla letteratura sugli alberi binari di ricerca bilanciati sappiamo che la ricerca è più efficiente in alberi AVL rispetto ad alberi RedBlack. Inoltre, se i dati input sono ordinati o quasi ordinati, il costo complessivo di tutti gli inserimenti è inferiore in alberi AVL rispetto ad alberi RedBlack.
   Provare sperimentalmente le asserzioni precedenti.

2. Progettare la classe **MyTreeMap** che espone la stessa interfaccia di TreeMap ma implementa i metodi after() e before() con complessità di tempo O(1), senza aumentare la complessità di tempo di tutti gli altri metodi.
   Progettare la classe **MyAVLTreeMap**, derivata da MyTreeMap, che implementa tutti i metodi della classe padre garantendo almeno le stesse complessità di tempo di AVLTreeMap.

3. Implementare il metodo **LCA**(p, q) della classe MyTreeMap che, prese in input due position p e q, restituisce la position dell'antenato comune dei nodi identificati da p e q che si trova al livello più grande nell'albero. Il metodo LCA() deve aere complessità di tempo O(h), dove h è l'altezza dell'albero.

4. Progettare la classe **MyRBTreeMap** che estende RBTreeMap aggiungendo due metodi: **split**() e **fusion**().
   - Il metodo split(*k*), invocato sul MyRBTreeMap T, prende in input una chiave *k* e restituisce due MyRBTreeMap T1 e T2 contenenti, rispettivamente, tutte le chiavi di T minori di *k* e tutte le chiavi di T maggiori di *k* (la chiave *k* viene eliminata). Il metodo deve avere complessità O(log *n*), dove *n* è il numero delle chiavi di T.
     N.B. Il metodo split distrugge l'albero t originario.
   - Il metodo fusion(T1), invocato sul MyRBTreeMap T, prende in input un RBTreeMap T1, le cui chiavi sono tutte maggiori delle chiavi presenti in T e inserisce in T tutte le chiavi di T1. Nel caso in cui esista una chiave di T1 minore di una chiave in T, il metodo deve lanciare un'eccezione ValueError.
     (Domanda bonus 3 punti extra: fornire un'implementazione di fusion() con complessità di tempo di O(log n + log m), dove n sono le chiavi dell'albero T e m sono le chiavi dell'albero T1).

La soluzione deve essere caricata sul sito del corso **entro il 15 novembre**.
Il materiale consegnato deve essere costituito da un unico file compresso contenente:

- Un file pdf con la documentazione del progetto
- Un progetto Python chiamato #gruppo_2_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, 4);
  - script di testing per ciascun esercizio.