ESERCIZI (alberi binari, alberi binari di ricerca, dizionari)

1. Si consideri un array S di n chiavi intere.
   - Si dia il codice di un algoritmo che con un'unica scansione di S conti il numero r di chiavi distinte in S. Usare un dizionario D inizialmente vuoto (non interessa l'implementazione di D).
   - Facendo l'assunzione che D sia implementato come un array ordinato, si analizzi la complessità in funzione di n e del numero r di chiavi distinte.

2. Data una tabella hash T in cui le collisioni sono risolte con il concatenamento, definire una procedura di ricerca di una chiave che sposti la chiave cercata, se presente nella tabella, in testa alla propria lista di trabocco.

3. Si dimostri che in un albero binario il numero di puntatori *left* e *right* uguali a NIL è uguale a uno più il numero totale di nodi dell'albero.

4. Dato un albero binario di ricerca T, progettare e descrivere in pseudocodice un algoritmo efficiente che, per ogni nodo u dell'albero, stampi la chiave di u e la chiave di valore minimo del sottoalbero di cui u è radice. Analizzare la complessità dell'algoritmo proposto.

5. Progettare un algoritmo per trasformare un array ordinato di n interi distinti in un albero binario di ricerca che sia il più bilanciato possibile, in tempo $O(n)$. Valutare esplicitamente la complessità dell'algoritmo proposto.

6. Dato un intervallo $[a, b]$ e un albero binario di ricerca, progettare un algoritmo efficiente per stampare in ordine crescente le chiavi $k \in [a, b]$.

7. Dato un albero binario di ricerca T di n nodi, progettare un algoritmo che restituisca un array ordinato contenente le chiavi memorizzate nei nodi di T.

8. Un albero binario *completo* è un albero in cui ogni nodo interno ha esattamente due figli. Sia T un albero binario completo. Dato un nodo $v \in T$ si definisca `imbalance(`v`)` la differenza in valore assoluto tra il numero di foglie nei sottoalberi sinistro e destro di v (se v è una foglia `imbalance(`v`)=0`). Si definisca anche `imbalance(`T`)` $= max_{v \in T}$ `imbalance(`v`)`.
   - Dimostrare un limite superiore all'imbalance di un albero binario completo con n nodi, e descrivere un albero il cui imbalance raggiunge tale limite.
   - Disegnare un albero binario completo T in cui `imbalance(`T`)` = `imbalance(`v`)` e v non è la radice dellalbero.
   - Si progetti un algoritmo efficiente per determinare `imbalance(`T`)` e analizzarne la complessità in tempo.

9. Dato un albero binario T, i cui nodi sono colorati di *bianco, rosso* o *verde*, progettare un algoritmo efficiente che stabilisca se esiste un cammino di tre nodi in T i cui colori formano la bandiera italiana. (Il cammino può partire da un nodo qualsiasi, non necessariamente dalla radice.)

10. Dato un albero binario i cui nodi sono colorati di rosso o di nero, progettare un algoritmo efficiente che calcoli il numero di nodi aventi lo stesso numero di discendenti rossi e neri. (Un nodo è discendente di se stesso.)