

Esame scritto ASD settembre 2023

Tempo totale 50 minuti; punteggio massimo 6.9; sufficienza 4 punti

Risposte prive di motivazione chiara e convincente comportano l'assegnazione di 0 punti

Le risposte sulla complessità degli algoritmi saranno valutate solo se la descrizione del relativo algoritmo è corretta

Nome _____

Cognome _____

Matricola _____

Hai fatto richiesta ai docenti di agevolazioni per DSA? Si No

Se hai fatto correttamente richiesta, rispondi solo alle domande 1 e 2

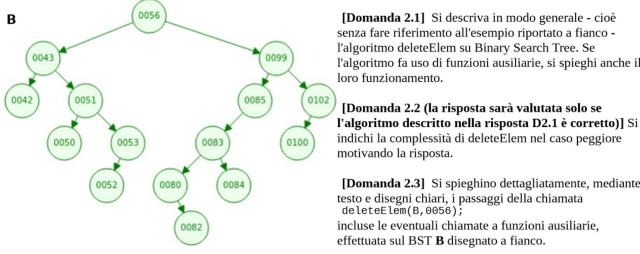
Domanda #1, quicksort

[1/2 punteggio, quicksort caso migliore] Si scriva quanto vale la complessità di quicksort nel caso **migliore** e si illustri dettagliatamente **quando** si ricade nel caso **migliore** e **come** si calcola la complessità di quicksort nel caso **migliore**.

[1/2 punteggio, quicksort caso peggiore] Si simuli l'esecuzione di quicksort sulla sequenza riportata sotto scegliendo il pivot in modo che si ricada nel caso **peggiore**.

82 17 33 29 23 41 59 16

Domanda #2, binary search tree



Domanda #3, tabelle di hash

Si consideri una tabella di hash con liste di collisione che implementa un dizionario **D** in cui le chiavi sono sequenze di 4 cifre decimali che indichiamo con **a**, **b**, **c**, **d** e i valori sono stringhe.

La tabella di hash ha 7 bucket indicizzati da 0 a 6 e la funzione di hash è **h**: $(a^2 + b + c + d) \bmod 7$.

[Preparazione dei dati, senza voto MA LA PRESENZA DI DUE O PIU' ERRORI NEL CALCOLO DELLA FUNZIONE DI HASH COMPORTA L'ASSEGNAZIONE DI ZERO PUNTI ALL'INTERO ESERCIZIO]

Completate su questo foglio lo schema mostrato sotto calcolando la funzione di hash delle chiavi scritte nella prima riga e riportandola nella corrispondente cella nella terza riga:

chiave = 3 7 4 1	chiave = 1 1 4 7	chiave = 2 2 7 1	chiave = 7 2 5 2	chiave = 2 1 6 3	chiave = 7 2 0 8	chiave = 1 3 1 1	chiave = 2 7 8 9
valore = "ma"	valore = "co"	valore = "gra"	valore = "pe"	valore = "le"	valore = "re"	valore = "ca"	valore = "giu"
h(chiave) =							

[Domanda 3.1, 1/3 del punteggio] Il dizionario D implementato dalla tabella di hash è inizialmente vuoto.

D3.1: Disegnate la tabella di hash che si ottiene dagli inserimenti delle stringhe "ma", "co", "gra", "pe", "le", "re", "ca", "giu" associate alle chiavi su quattro cifre riportate nello schema sopra, nell'ordine in cui compaiono nello schema da sinistra verso destra. Nel caso ci fossero delle chiavi duplicate, ignorate l'inserimento e scrivete esplicitamente sul foglio "questa coppia chiave-valore è stata ignorata perché la chiave è duplicata".

[Domanda 3.2, 1/3 del punteggio (la risposta sarà valutata solo se la risposta alla domanda 3.1 è corretta)]

D3.2.1: Quali sono le due principali "buone proprietà" che una funzione di hash deve avere?

D3.2.2: La funzione di hash **h** applicata alle chiavi nello schema disegnato sopra gode della **prima** di queste due "buone proprietà"? Motivate esaurientemente la risposta.

D3.2.3: La funzione di hash **h** applicata alle chiavi nello schema disegnato sopra gode della **seconda** di queste due "buone proprietà"? Motivate esaurientemente la risposta.

[Domanda 3.3, 1/3 del punteggio] Sia **m** la dimensione della tabella e sia **n** il numero di elementi nella tabella. Si assuma che **m** ed **n** siano entrambi diversi da 0.

D3.3.1: Si descriva l'algoritmo per l'inserimento di un elemento data la sua chiave; l'algoritmo deve esplicitamente e correttamente gestire il caso di chiavi duplicate.

D3.3.2 (la risposta sarà valutata solo se l'algoritmo descritto nella risposta D3.3.1 è corretto): Sotto quali condizioni l'inserimento ha complessità Theta(n)?

D3.3.3 (la risposta sarà valutata solo se l'algoritmo descritto nella risposta D3.3.1 è corretto): Sotto quali condizioni l'inserimento ha complessità Theta(n/m)?

D3.3.4 (la risposta sarà valutata solo se l'algoritmo descritto nella risposta D3.3.1 è corretto): Sotto quali condizioni l'inserimento ha complessità Theta(1)?

Domanda 1.1

La complessità di quicksort nel caso migliore è $O(n \log n)$. Si ricade nel caso migliore quando si sceglie l'elemento mediano come pivot; è però solo un caso ipotetico, in quanto non è possibile sapere qual'è il mediano senza riordinare la sequenza. Essendo che il pivot è sempre l'elemento mediano, l'array verrà diviso dalla partition in due sottosequenze di lunghezza circa $\frac{n}{2}$, ottenendo quindi un albero binario che ha altezza $\log_2(n)$, ossia il numero di iterazioni della funzione ricorsiva. Ad ogni livello J dell'albero la partition viene effettuata su 2^J sottoproblemi, ognuno lungo $\frac{n}{2^J}$: la complessità di ogni livello è quindi $(2^J) \cdot (\frac{n}{2^J}) = O(n)$. La complessità finale sarà quindi il prodotto tra la complessità di ogni livello e l'altezza dell'albero, quindi $O(n) \cdot \log_2(n) = O(n \log n)$

Domanda 1.2

82	17	33	29	23	41	59	16
17	33	29	23	41	59	16	82
17	33	29	23	41	16	59	82
17	29	23	16	33	41	59	82
17	23	16	29	33	41	59	82
17	16	23	29	33	41	59	82
16	17	23	29	33	41	59	82

Domanda 2.1

L'algoritmo `deleteElem` ha come obiettivo la rimozione di un nodo mantenendo le proprietà dell'albero. L'algoritmo inizia a cercare il nodo da eliminare, confrontando il valore del nodo da eliminare con quello corrente, partendo dalla radice. Se il valore è minore del nodo corrente, si procede cercando a sinistra, se è maggiore si cerca a destra. Trovato il nodo, si procede a eliminarlo. Se il nodo è una foglia, si rimuove semplicemente dall'albero; se il nodo ha un solo figlio, il nodo viene rimosso e il figlio viene collegato direttamente al padre del nodo eliminato; se il nodo ha due figli, l'algoritmo trova il successore in ordine (il nodo con il valore più piccolo nel sottoalbero destro) o il predecessore in ordine (il nodo con il valore più grande nel sottoalbero sinistro). Il valore del nodo predecessore o successore sostituisce il valore del nodo da eliminare e successivamente il nodo predecessore o successore viene eliminato ricorsivamente.

Domanda 2.2

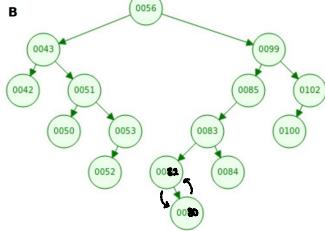
Nel caso peggiore la complessità è $O(n)$, con n =numero nodi. Questo avviene quando l'albero è completamente sbilanciato, somigliando ad una lista collegata, e il nodo da eliminare si trova all'estremità dell'albero. In questo scenario, l'algoritmo deve quindi attraversare tutti i nodi fino a trovare e eliminare il nodo desiderato.



completamente sbilanciato, somigliando ad una lista collegata, e il nodo da eliminare si trova all'estremità dell'albero. In questo scenario, l'algoritmo deve quindi attraversare tutti i nodi fino a trovare e eliminare il nodo desiderato.

Domanda 2.3

1) cerchiamo il successore in ordine, ossia il nodo più piccolo nel sottoalbero destro. Quest'ultimo è 0080, che però ha un figlio, quindi prima lo swap.



2) Elimino 0080 (funzione ausiliare deleteMin)

temp=0080



3) deleteMin passa la chiave valore a deleteElem, che sostituisce 0056 con 0080.

