

Esercizi

2. Alberi

Esercizio 2.1

Scrivete una funzione ricorsiva che, ricevendo il riferimento a un albero contenente numeri interi, restituisca la somma di tutti i valori pari contenuti nei nodi dell'albero.

Esercizio 2.2

Disegnate l'albero di ricerca ottenuto inserendo uno dopo l'altro in un albero inizialmente vuoto i numeri 15 7 2 16 13 1 4 20 17.

Indicate l'elenco dei valori contenuti nei nodi effettuando le seguenti visite:

- in ordine anticipato
- in ordine simmetrico
- in ordine posticipato

Indicate l'elenco dei nodi incontrati effettuando la visita in ampiezza.

Esercizio 2.3

Ripetete l'esercizio 2.2 utilizzando i numeri 20 11 18 16 17 3 30 4 19 25. Disegnate poi l'albero che si ottiene rimuovendo, nell'ordine, i valori 20 18 dall'albero ottenuto.

Esercizio 2.4

Ripetete gli esercizi 2.2 e 2.3 effettuando gli inserimenti in modo da ottenere, a ciascun passo, un albero AVL.

Esercizio 2.5

Disegnate un albero di ricerca *perfettamente bilanciato* contenente tutti i numeri da 1 a 9 (estremi compresi).

Esercizio 2.6

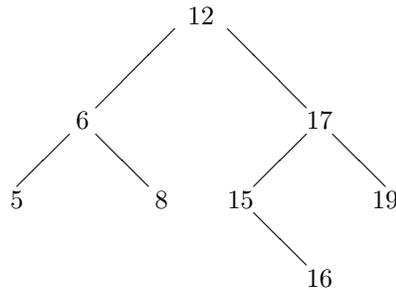
Disegnate un albero di ricerca *perfettamente bilanciato* contenente i numeri 1 4 5 9 13 17 23 29 30.

Esercizio 2.7

Disegnate un albero di ricerca *perfettamente bilanciato* contenente i numeri 1 3 5 7 9 11 13 15 17 19 21 23 25.

Esercizio 2.8

Considerate il seguente albero di ricerca:



Indicate l'elenco dei valori contenuti nei nodi effettuando le seguenti visite:

- in ordine anticipato
- in ordine simmetrico
- in ordine posticipato

Indicate l'elenco dei nodi incontrati effettuando la visita in ampiezza.

Disegnate l'albero che si ottiene rimuovendo, nell'ordine, i valori 12 6 dall'albero ottenuto.

Esercizio 2.9

Disegnate un qualsiasi albero 2-3 contenente i valori 1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31. Disegnate un albero 2-3 con gli stessi valori e il minimo numero possibile di nodi. Disegnate un albero 2-3 con gli stessi valori e il massimo numero di nodi.

Esercizio 2.10

Quant'è l'altezza minima di un albero 2-3 in cui sono memorizzate 20 chiavi? E l'altezza massima? Rispondete alle stesse domande nel caso di 200 e 2000 chiavi.

Esercizio 2.11

Disegnate il B-albero di ordine (o grado minimo) $t = 2$ che si ottiene a partire dall'albero vuoto inserendo, nell'ordine, i valori 12 14 20 25 8 13 10 11 26 22 19 24 27 15 16 21.

Esercizio 2.12

Ripetete l'esercizio 2.11 con $t = 3$.

Esercizio 2.13

Calcolate il numero massimo di chiavi che possono essere collocate in un B-albero di altezza h e ordine t (potete ispirarvi al calcolo del numero minimo, presentato a lezione e nel Lemma 6.6 sul libro).

Esercizio 2.14

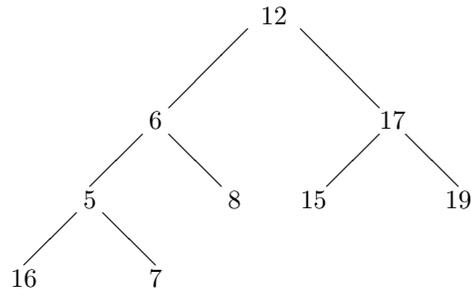
Dimostrate che se in un albero binario ogni nodo interno possiede entrambi i figli, allora il numero delle foglie supera di 1 il numero di nodi interni. Che relazioni ci sono tra numero di nodi interni e numero di foglie nel caso ci possano essere nodi interni con un unico figlio?

Esercizio 2.15

Considerate un albero binario contenente n nodi, in cui i nodi contengono puntatori ai figli sinistro e destro. Esprimete il numero di puntatori null presenti nei nodi dell'albero in funzione di n .

Esercizio 2.16

Considerate il seguente albero binario quasi completo:



Applicate l'algoritmo heapsort all'albero: prima trasformate l'albero in uno heap, partendo dalle foglie e disegnate la struttura ottenuta; successivamente, per ogni iterazione dell'algoritmo, disegnate lo heap che si ottiene dopo avere rimosso la foglia più a destra nell'ultimo livello, collocato il suo valore nella radice, e risistemato lo heap.

Esercizio 2.17

Considerate il vettore posizionale contenente i seguenti elementi 12 14 20 25 13 10 22 19 24 27 15 16 21. Disegnatelo come albero binario e applicate l'algoritmo heapsort, disegnando l'evoluzione dello heap, come per l'esercizio 2.16.

Esercizio 2.18

Considerate il vettore posizionale contenente i seguenti elementi 20 11 18 16 17 3 30 4 19 25. Applicate l'algoritmo heapsort *in loco*, scrivendo il contenuto dell'array dopo averlo trasformato in uno heap, e dopo ogni iterazione dell'algoritmo.

Esercizio 2.19

Considerate un albero binario quasi completo contenente n nodi in cui, nel livello massimo, tutte le foglie del livello massimo si trovano il più a sinistra possibile. Calcolate il valore esatto dell'altezza dell'albero in funzione di n e, per ciascun livello, il numero di nodi.

Supponete di rappresentare l'albero con un vettore posizionale, come quello utilizzato per heapsort, memorizzando la radice nella posizione 0. Per ogni livello ℓ dell'albero esprimete in funzione di ℓ l'intervallo di posizioni dell'array che corrispondono ai nodi di livello ℓ dell'albero (si può osservare che il livello 0 corrisponde alla posizione 0, il livello 1 alle posizioni 1, 2, il livello 2 alle posizioni 3, ..., 6, cercate di trovare una formula generale e verificatela poi per induzione). Dimostrate che i figli del nodo interno memorizzato in posizione i dell'array, sono memorizzati nelle posizioni $2i + 1$ e $2i + 2$. Ricavate infine una formula per ottenere, dalla posizione di un nodo diverso dalla radice, la posizione del nodo padre.

Esercizio 2.20

Ripetete la seconda parte dell'esercizio 2.19, supponendo di utilizzare indici di array a partire da 1 e, dunque, di memorizzare la radice in posizione 1.