Corso di Algoritmi e Strutture Dati
Anno Accademico 2017–2018
Prof. Giovanni Pighizzini

Esercizi

4. Alberi 2-3, B-alberi, heapsort

Esercizio 4.1
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.

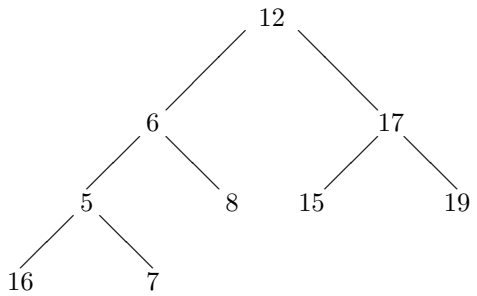
Esercizio 4.2
Quant'è l'altezza minima di un albero 2-3 in cui sono memorizzate 20 chiavi? E l'altezza massima? Rispondente alle stesse domande nel caso di 200 e 2000 chiavi.

Esercizio 4.3
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 4.4
Ripetete l'esercizio 4.3 con $t = 3$.

Esercizio 4.5
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 4.6
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 4.7
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 4.6.

Esercizio 4.8
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 4.9
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 profondità p dei nodi presenti nell'albero esprimente in funzione di p l'intervallo di posizioni dell'array che corrispondono ai nodi di profondità p (si può osservare che profondità 0 corrisponde alla posizione 0, profondità 1 alle posizioni $1, 2$, profondità 2 alle posizioni $3, \ldots, 6$, 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 4.10
Ripetete la seconda parte dell'esercizio 4.9, supponendo di utilizzare indici di array a partire da 1 e, dunque, di memorizzare la radice in posizione 1.

Esercizio 4.11
Fornite un esempio che mostri che l'algoritmo heapsort non fornisce un ordinamento stabile.