Note alla correzione dell'esame del 7 luglio 2020
Introduzione agli Algoritmi M-Z
Prof. Saverio Caminiti

Es. 1
La funzione fa due chiamate ricorsive su n/2 e un while che viene eseguito t/2 volte (t decrementa di 2 in 2). Dato che $t = n^3$ alla fine si otteneva:
$T(n) = 2T(n/2) + \Theta(n^3)$ molti si sono persi n^3 e hanno ottenuto n, cn, c o altro.
$T(1) = \Theta(1)$ è sempre importante scrivere il costo del caso base

Risolvendo la versione semplificata (senza notazione asintotica) con il metodo iterativo (e non con albero della ricorsione o per sostituzione!) si arrivava ad avere al generico passo i.

$T(n) = 2^i\, T(n/2^i) + \sum_{j=0..i-1} [\, 2^j\, c(n/2^j)^3 \,]$

Al passo finale $k = \log_2 n$ si ottiene
$T(n) = 2^k\, T(1) + cn^3 \sum_{j=0..k-1} [\, 2^j / 8^j \,] = n\, c' + cn^3 * d = \Theta(n^3)$
Infatti la serie geometrica di ragione $¼^j$ converge ad una costante.

Es. 2
L'esercizio parlava di partizionare, quindi la scelta più naturale era usare/adattare la funzione partition del quicksort.

Si poteva ad esempio chiamarla due volte prima con pivot p1 e poi con pivot p2 (magari la seconda volta solo dall'indice m in poi per evitare di ripassare inutilmente sugli elementi <=p1 già posizionati all'inizio dell'array). Era necessario in questo caso dettagliare la funzione partition e darne una versione modificata che prende un pivot come parametro (e non come primo elemento dell'array) e magari che non da per scontato che il pivot appartenga all'array.
Andava bene anche dare una sola funzione con due while principali uno sotto l'altro corrispondenti alle due esecuzioni della partition.

In alternativa si poteva fare banalmente con un array d'appoggio in cui si scrivono prima i valori <=p1, poi quelli intermedi e poi quelli >p2 scorrendo 3 volte l'array A. Oppure usare 3 array d'appoggio e scorrere A una sola volta. Queste soluzioni usano un array aggiuntivo e sono state valutate fino a 90% del volto se scritte correttamente.

Ultima soluzione lecita era ordinare l'array (e quindi gli elementi erano tutti ordinati e di conseguenza lo erano anche rispetto a p1 e p2). Però questa soluzione richiede O(n log n) mentre le precedenti erano $\Theta(n)$.
Non era lecito usare integer sort: non ci sono informazioni sui valori contenuti in A.

Un errore molto frequente è stato confondere indici e valori.
Gli indici sono numeri da 0 a n-1 (o da 1 a n se volte), i valori nell'array invece possono rappresentare molte cose, pensate ad esempio che siano il saldo di conti bancari espressi in €.
Magari n = 200 e i valori invece possono raggiungere 1 milione di €.
Confondere indici e valori quando si manipola un array è un errore grave.

Es 3.
Qui si doveva scegliere un solo valore da inserire nell'albero. L'inserimento avviene secondo l'algoritmo degli ABR, quindi non si può dire aggiungo 100 come figlio sinistro di 20, non ha senso. Scelto il valore finisce dove deve finire in base all'algoritmo che parte dalla radice e scende a dx o sx rispettando la proprietà di ordinamento finchè non arriva a poter creare una foglia per il nuovo valore.

Per creare una doppia rotazione si doveva creare un caso SD aggiungendo un figlio dx a 20 (es inserisco il valore 25). Analogamente un figlio dx al 120 (inserendo qualcosa tipo 130), ma io qui scelgo il 25.
Altri valori creavano solo casi SS che si risolvono con una rotazione singola o semplicemente non sbilanciavano proprio niente (come i figli del 60).

Le rotazioni si fanno partendo dal basso e risalendo verso la radice.
Quindi il primo nodo sbilanciato era il 30, caso SD devo fare
rot_dx(30) preceduta però da una rotazione inversa sul figlio da cui viene lo sbilanciamento rot_sx(20).
Quindi prima 25 diventa figlio di 30 e 20 figlio di 25.
Poi 25 viene su e si ritrova con figli 25 e 30 e padre 50.
Fine.

La risposta alla E è che in generale negli AVL in inserimento basta una sola fix in quando le rotazioni fatte quando si trova un nodo v sbilanciato (qualsiasi casso sia SS, SD, DD, DS) fanno si che l'altezza di T_v dopo il fix sia identica a come era prima dell'inserimento. Quindi non possono esserci ulteriori sbilanciamenti.