Compito del 2 aprile 2004

1. Completare la descrizione in pseudocodice dell'algoritmo HeapSort per l'ordinamento di un array sul posto

   Algorithm HeapSort (v, n):
   Input: un array v che contiene n *entry*
   Output: l'array v ordinato
   ...

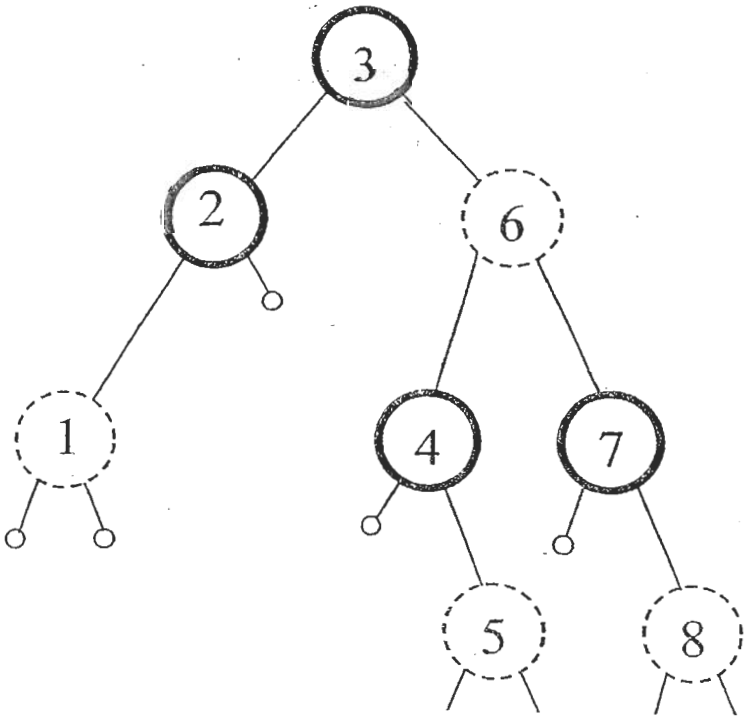
   Nella costruzione dell'algoritmo è vietato usare una coda prioritaria e/o altri algoritmi dei quali non venga data la descrizione in pseudocodice (10 punti).

2. Sappiamo che per fondere due array ordinati, ciascuno di lunghezza $n/2$, in un array ordinato di lunghezza n è necessario un tempo pari a $k_1 \cdot n$ e che per copiare un array di lunghezza n in un altro array è necessario un tempo pari a $k_2 \cdot n$. In tali ipotesi dimostrare che per ordinare un array di lunghezza $n=2^h$ l'algoritmo di ordinamento Mergesort richiede un tempo $T(n)=(k_1+k_2) \cdot n \cdot log_2(n)+k_3 \cdot n$ sapendo che per ordinare un array di lunghezza pari a 1 è necessario un tempo k_3 (10punti).

3. Proporre un algoritmo efficiente per visitare un albero ordinato "per livelli" (detto anche *level numbering*), visitando tutti noti alla profondità d prima di quelli a profondità $d+1$ (suggerimento, tenere traccia di ogni nodo visitato in una coda) (10 punti).

4. Dato l'albero rosso-nero di figura ← questo c'è SEMPRE
   a. Trasformare l'albero rosso-nero nel corrispondente albero (2, 4).
   b. Come si trasforma l'albero rosso-nero in conseguenza dell'inserzione delle chiavi 9 e 10?
   c. Come si trasforma l'albero rosso-nero di partenza in conseguenza della cancellazione delle chiavi 3 e 6?

   Nella figura i nodi rossi hanno bordo tratteggiato, quelli neri sono indicati in grassetto, nello svolgimento spiegare il perché dei vari passaggi (6 punti).



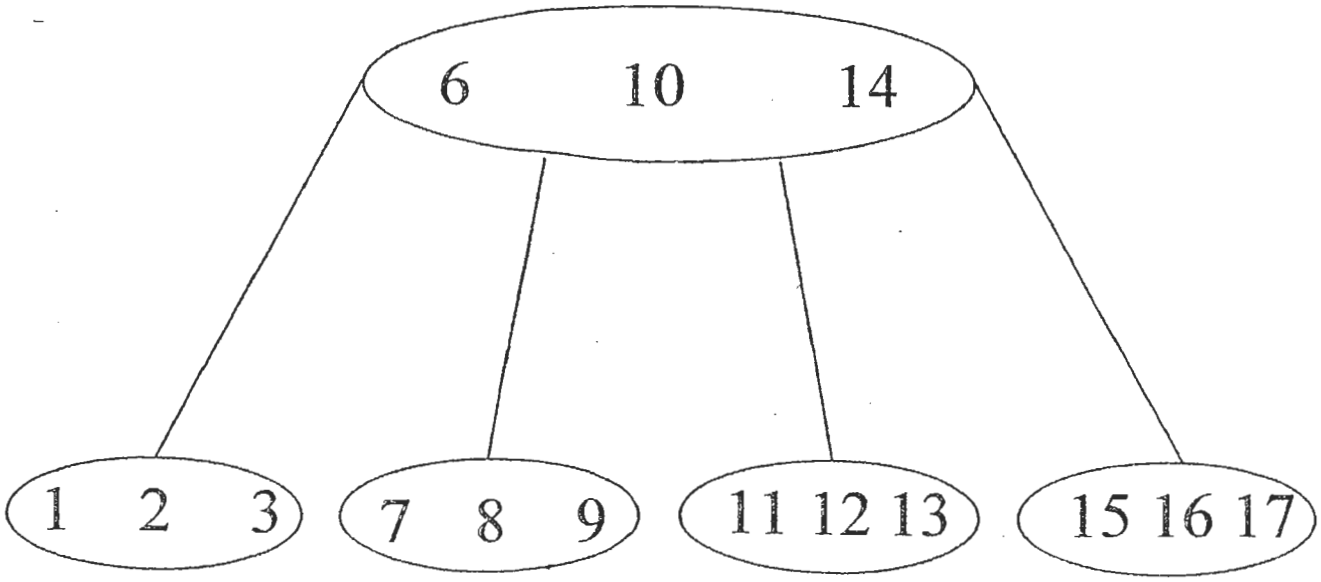
Dati e algoritmi 1 – matricole dispari

Compito del 22 marzo 2004

1. Completare la descrizione in pseudocodice dell'algoritmo di cancellazione di una chiave *k* da una "skip list"

   Algorithm SkipRemove (*k*):
   Input: la chiave *k* da eliminare
   Output: il valore *v* associato alla chiave *k*, se presente, *null* se la chiave *k* non è presente nella skip list

2. L'altezza di un albero T non vuoto è data dall'altezza della sua radice, dopo aver riportato le definizioni di altezza e profondità di un nodo di un albero dimostrare che l'altezza di un albero T è pari alla massima profondità dei nodi esterni di T.

3. Date n chiavi memorizzate in un vettore, proporre un algoritmo efficiente per inserire le chiavi in un albero di ricerca binario in modo che l'albero risulti di altezza minima.

4. Dato l'albero (2, 4) di figura
   a. Trasformare l'albero (2, 4) nel corrispondente albero rosso-nero.
   b. Come si trasforma l'albero (2, 4) in conseguenza dell'inserzione delle chiavi 4 e 5.
   c. Come si trasforma l'albero (2, 4) in conseguenza della successiva cancellazione delle chiavi 10 e 11.