008AA – ALGORITMICA E LABORATORIO

Appello del 24 giugno 2010

SOLUZIONI

Esercizio 1.
Dato un array a di n numeri interi distinti, progettare un algoritmo per determinare se esiste un intero k tale che la somma degli elementi di a di valore minore o uguale a k sia uguale alla somma degli elementi di valore maggiore di k. Discutere la complessità dell'algoritmo proposto.

Soluzione

```
EsisteK(a)
  HeapSort(a, 0, n-1);
  somma = 0;
  for (int i = 0; i < n; i++)   somma = somma + a[i];
  sommaMinK = 0;
  for (int i = 0; i < n; i++) {
     sommaMinK = sommaMinK + a[i]; // k = a[i];
     sommaMagK = somma - sommaMinK;
     if (sommaMinK == sommaMagK) return true;
  }
  return false;
```

La complessità in tempo è $T(n) = O(n \log n + n + n) = O(n \log n)$, la complessità in spazio è $S(n) = O(1)$.

Esercizio 2.
Per un certo problema sono stati trovati due possibili algoritmi risolutivi. Il tempo di esecuzione del primo soddisfa alla relazione di ricorrenza:

$$T_1(n) = \begin{cases} 4T_1(n/2) + 2n^2 & n > 1 \\ 1 & n = 1 \end{cases}$$

mentre il tempo di esecuzione del secondo soddisfa alla relazione di ricorrenza:

$$T_2(n) = \begin{cases} 3T_2(n/2) + 3n^2 \log^2 n & n > 1 \\ 1 & n = 1 \end{cases}$$

Si dica, giustificando la risposta, quale dei due algoritmi è da preferire nel caso si debbano risolvere problemi di grandi dimensioni.

Soluzione
Le due relazioni di ricorrenza si possono risolvere applicando il teorema principale:

$$\begin{aligned} T_1(n) &= \Theta(n^2 \log n) \quad \text{(II caso)}, \\ T_2(n) &= \Theta(n^2 \log^2 n) \quad \text{(III caso)}. \end{aligned}$$

Per risolvere problemi di grandi dimensioni è dunque da preferire il primo algoritmo.

Esercizio 3.
Data la sequenza di chiavi:

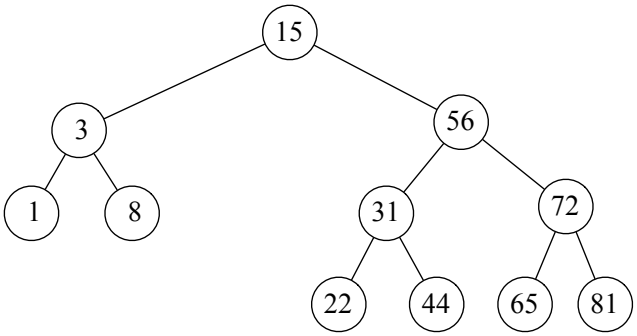
$$1, 3, 8, 15, 22, 31, 44, 56, 65, 72, 81$$

disegnare:

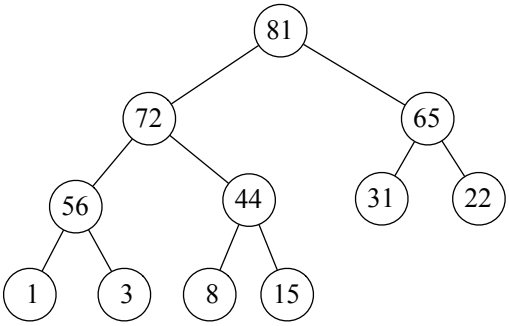
1. un albero binario di ricerca
2. un heap di massimo
3. un albero AVL

Soluzione

Albero binario di ricerca e albero AVL



Heap di massimo



Esercizio 4.
Scrivere una funzione C

```
int ConvertiBin (char *num);
```

che, data una stringa binaria positiva di al più 16 bit, restituisce il corrispondente valore intero. Scrivere anche il `main()` che invoca `ConvertiBin` su una stringa binaria inserita a tempo d'esecuzione.

Soluzione

```
#include <stdio.h>
#define DIM 16

int main () {
char num[DIM];

printf ("inserisci un numero intero positivo in forma binaria: ");
scanf ("%s",num);

printf ("il numero che hai inserito corrisponde,
                            in decimale, a: %d", ConvertiBin(num));

}//main

int ConvertiBin (char *num) {

int i=0;
int somma = 0;

while (num[i]!='\0')
    {
somma = somma * 2;  // moltiplico la somma parziale
if (num[i]=='1') somma++; // se e' il caso sommo 1
i++;
    }//while

return somma;

}
```