008AA – ALGORITMICA E LABORATORIO
Appello del 31 maggio 2010

SOLUZIONI

Esercizio 1.
Sia a un array di n interi distinti, tale che esiste una posizione j, $0 \leq j < n$, per cui:

- gli elementi nel segmento $a[0, j]$ sono in ordine crescente;
- gli elementi in $a[j+1, n-1]$ sono in ordine decrescente;
- $a[j] > a[j+1]$, se $j < n-1$.

Descrivere un algoritmo che, ricevuto in ingresso a, trova la posizione j in tempo $O(\log n)$. Calcolare la complessità al caso pessimo dell'algoritmo indicando, e risolvendo, la corrispondente relazione di ricorrenza.

```
TrovaPosizione(a, sx, dx)
    if (sx > dx) return -1;
    if (sx == dx) return sx;
    centro = (sx + dx) / 2;
    if (a[centro] < a[centro+1]) return TrovaPos(a, centro+1, dx);
    else return TrovaPos(a, sx, centro);
```

Costo in tempo:
$T(n) = T(n/2) + O(1)$,
Teorema Principale, secondo caso: $T(n) = O(\log n)$

Esercizio 2.
Si vuole inserire la sequenza di chiavi intere

$$5, 15, 2, 8, 30, 4, 19, 11, 26$$

in una tabella hash di $m = 11$ posizioni, inizialmente vuota, gestendo le collisioni con indirizzamento aperto e hashing doppio, con $h_1(k) = k \bmod 11$ e $h_2(k) = k \bmod 10 + 1$.

Indicare come è riempita la tabella hash alla fine delle operazioni, specificando per ogni chiave la sequenza di posizioni esaminate.

chiave	sequenza
5	5
15	4
2	2
8	8
30	8, 9
4	4, 9, 3
19	8, 7
11	0
26	4, 0, 7, 3, 10

11		**2**	**4**	**15**	**5**		**19**	**8**	**30**	**26**

Esercizio 3.
Dato un grafo non orientato $G = (V, E)$, progettare un algoritmo che stabilisca se G è un albero in tempo $O(n)$, dove $n = |V|$.
Suggerimento: *un grafo G è un albero se e solo se G è connesso e $|E| = |V| - 1$.*

Schema di soluzione:

1. Si contano gli archi. Se si supera il valore $n = |V|$, si restituisce `false`. In questo conteggio si deve tener conto del fatto che ogni arco è contato due volte.
2. se il numero di archi è minore di $|V| - 1$, si restituisce `false`;
3. altrimenti, si controlla con una visita in ampiezza o in profondità se il grafo è connesso.

Costo in tempo:
Il conteggio degli archi si arresta non appena si raggiunge $|V|$, e la visita si esegue solo se $|E| = |V| - 1$. Dunque il costo è $O(|V| + |V| + |E|) = O(|V|)$.

Esercizio 4.
Scrivere una funzione C `int *Converti (int numero);` che, dato un intero maggiore o UGUALE a 0, restituisce la corrispondente stringa binaria. Scrivere anche il `main()` che invoca `Converti` su un intero inserito a tempo d'esecuzione.

```
# include <stdio.h>
# include <stdlib.h>
# include <math.h>

double my_log(double x, double base) {
  return log(x) / log(base);
}

int *Converti(int numero) {
  int *binario;
  int resto;
  if (numero == 0) {
    binario=malloc(2*sizeof(int));
    binario[0]=0;
    binario[1]=-1;
    return binario;
  }
  int cifre=my_log(numero,2);
  binario = malloc ((cifre+2)*sizeof(int));
  int i=0;
  do {
    resto=numero % 2;
    numero=numero/2;
    binario[cifre-i]=resto;
    i++;
  }
  while(numero!=0);
  // Fine stringa binaria
  binario[cifre+1]=-1;
  return binario;
}

int main () {
  int num;
  int resto;
  int* binario;
  do {
    printf( "inserisci un numero maggiore o uguale a 0\n");
    scanf("%d",&num);
    if (num<0) printf("sai leggere? deve essere maggiore o uguale di 0!!!");
   } while(num<0);
   binario=Converti(num);
   printf ("la conversione in binario e': ");
   int i=0;
   while (binario[i]!=-1) {
      printf("%d",binario[i]);
    i++;
   }
}
```