Algoritmi e Basi di dati – Modulo di Algoritmi e Strutture dati

ANNO ACCADEMICO 2011/2012 - gennaio 2012

Esercizio 1

Sia dato il seguente vettore **A** di interi:

```
A = [16  24  2  1  8  34 78]
```

Scrivere tutte le chiamate a **quicksort** generate dalla chiamata **quicksort(A, 0, 6)**. Per ogni chiamata indicare lo stato dell'array, il valore dei parametri attuali e del perno.

A	*Inf*	*Sup*	*Perno*
[16 24 2 1 8 34 78]	0	6	1
[1 24 2 16 8 34 78]	1	6	16
[1 8 2 16 24 34 78]	1	2	8
[1 2 8 16 24 34 78]	4	6	34
[1 2 8 16 24 34 78]			

Esercizio 2

Scrivere la funzione **buildHeap** e, tramite questa, trasformare in uno heap il seguente array:

A = [1 8 8 5 2 1 10 6 2 7]

Per ogni chiamata a **down**, indicare il valore del parametri **i** e lo stato dell'array dopo la chiamata.

Funzione **buildHeap**:

```
void buildHeap(int* A, int n) {
  for (int i = n/2; i >= 0; i--) down(A, i, n);
}
```

Esecuzione **buildHeap**:

A	**i**
[1 8 8 5 2 1 10 6 2 7]	**4**
[1 8 8 5 7 1 10 6 2 2]	**3**
[1 8 8 6 7 1 10 5 2 2]	**2**
[1 8 10 6 7 1 8 5 2 2]	**1**
[1 8 10 6 7 1 8 5 2 2]	**0**
[10 8 8 6 7 1 1 5 2 2]	

Esercizio 3

Calcolare la complessità del blocco in funzione del numero di nodi dell'albero binario **t**:

```
for (int i = 0; i <= es3(t)*f(t); i++) es3(t);
```

con le funzioni **f** e **es3** definite come segue. Indicare per esteso le relazioni di ricorrenza e, per ogni comando ripetitivo, il numero di iterazioni e la complessità della singola iterazione.

```
int es3(Node* t) {
  if (!t) return 1;
  t->label += f(t);
  cout << es3(t->left);
  return 1 + 2*es3(t->left) +
    es3(t->right);
}
```

```
int f(Node* t) {
  if (!t) return 1;
  for (int i = 0; i <= 2; i++)
    cout << f(t->left);
  return 1 + f(t->right);
}
```

Funzione f:

Numero di iterazioni del for: 3
Complessità della singola iterazione: $T_f(n/2)$
Complessità del for: $3T_f(n/2)$

$T_f(0)= d$
$T_f(n)= c + 4T_f(n/2)$ $T_f(n)$ è $O(n^2)$

$R_f(0)= 1$
$R_f(n)= 1+ R_f(n/2)$ $R_f(n)$ è $O(\log n)$

Funzione es3:
$T(0) = d$
$T(n) = cn^2+ 3T(n/2)$ $T(n)$ è $O(n^2)$

$R(0) = d$
$R(n) = c + 3R(n/2)$ $T(n)$ è $O(n^{\log 3})$

Calcolo for del blocco:
numero iterazioni: $n^{\log 3}\log n$
Complessità della singola iterazione: $T_f(n) + 2T(n) = O(n^2) + O(n^2) = O(n^2)$
Complessità del for: $O(n^{2+\log 3}\log n)$

Esercizio 4

Scrivere una funzione **`bool controllo(const Node* t1, const Node* t2)`** che, dati due alberi binari, restituisce **`true`** se **`t2`** coincide con l'albero ottenuto scambiando in **`t1`** il sottoalbero sinistro con il sottoalbero destro di ogni nodo.

```
bool controllo(const Node* t1, const Node* t2) {
   if (!t1 && !t2) return true;
   if ((!t1 && t2) || (t1 && !t2)) return false;
    if (t1->label != t2->label) return false;
    return controllo(t1->left, t2->right) && controllo(t1->right, t2-
    >left);
}
```

Esercizio 5

Scrivere una funzione **`void scambio(Node* t)`** che, dato un albero generico memorizzato con la rappresentazione figlio-fratello, scambia l'etichetta del primo figlio con quella dell'ultimo di ogni nodo che ha almeno due figli.

```
void scambio(Node* t) {
  if (!t) return;
  if (t->left && t->left->right) {
    for(temp = t->left; temp->right; temp = temp->right);
    int l = temp->label;
    temp->label = t->left->label;
    t->left->label = l;
  }
  scambio(t->left);
  scambio(t->right);
}
```