

3 luglio 2018

<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>5</i>	<i>6</i>	<i>7</i>
5	5	5	6	5	5	2

### Esercizio 1

- Dare la definizione di albero binario bilanciato e quasi bilanciato.
- Scrivere una funzione in c++ con complessità  $O(n)$ , dove  $n$  è il numero di nodi, che, dato un albero binario e un intero  $x$ , conta i nodi che hanno esattamente  $x$  sottoalberi vuoti nel proprio sottoalbero.

```
int conta (Node* t, int x, int & vuoti) {
    if (!t) {vuoti=1; return 0;}
    int vuoti_l, vuoti_r, conta_l, conta_r;
    conta_l=conta(t->left, vuoti_l);
    conta_r=conta(t->right, vuoti_r);
    vuoti= vuoti_l + vuoti_r;
    return (vuoti==x) + conta_l + conta_r;
}
```

### Esercizio 2

- Descrivere la memorizzazione figlio-fratello dell'albero generico.
- Che relazioni ci sono fra le visite di un albero generico e la visite del suo "trasformato"?
- Scrivere una funzione in c++ che, dato un albero generico, scambia il primo con l'ultimo sottoalbero di ogni nodo.

```
void scambia-Figli (Node* & t) {
    if (!t) return;
    if (! t->right) return;
    for (Node * a=t; a->right->right; a=a->right);
    Node * b= t;
    t=a->right;
    t->right=b->right;
    a->right=b;
    b->right=0;
}
```

```
void scambia (Node* t) {
    if (!t) return;
    scambia_figli(t->left);
    scambia (t->left);
    scambia (t->right);
}
```

### Esercizio 3

- Dare la definizione del tipo di dato heap con le operazioni e relative complessità
- Dato lo heap [80, 75, 75, 60, 20, 10, 40] indicarne il contenuto dopo l'inserimento del valore 78 e una successiva estrazione.

heap	Chiamate a up e down	
80, 78, 75, 75, 20, 10, 40, 60	up(7), up(3), up(1)	Dopo l'inserimento di 78
78, 75, 75, 60, 20, 10, 40	down(0), down(1), down(3)	Dopo un'estrazione

- Le rappresentazioni tramite array di due heap che hanno gli stessi elementi sono uguali? Dimostrare la risposta. **NO (es [20 10 13] e [20 13 10])**
- Dove si trova l'elemento più piccolo di un heap ? **nelle foglie: a partire da  $n/2$**

### Esercizio 4

- Descrivere brevemente l'algoritmo di ordinamento quicksort e indicare la sua complessità.
- Applicare il quicksort all'array seguente indicando tutte le chiamate e i il valore del perno per ogni chiamata.
- La complessità del quicksort è sempre la stessa qualsiasi sia la configurazione dei dati nell'array? Spiegare.
- Cosa vuol dire che un algoritmo è ottimo? Quicksort è un algoritmo di ordinamento ottimo? Spiegare la risposta.

<i>Array</i>	<i>inf</i>	<i>sup</i>	<i>perno</i>
8, 3, 15, 4, 6	0	4	15
8, 3, 6, 4, 15	0	3	3
3, 8, 6, 4, 15	1	3	6
3, 4, 6, 8, 15			

### Esercizio 5

Sia  $l$  il puntatore ad una lista semplice di interi. Calcolare la complessità dell'istruzione

```
for(int i=0;i<g(f(l)); i++) cout << "!";
```

in funzione della lunghezza  $n$  di  $l$  (indicando le relazioni di ricorrenza di tempo e risultato per ogni funzione) con le funzioni  $f$  e  $g$  definite come segue:

```
int f(Elem* h) {
    if (!h) return 1;
    int a=0;
    for(Elem* e=h;e;e=e->next){
        e->inf++;
        a++;
    }
    int b=a*a;
    for(int i=1; i<=b;i++){
        a++;
    }
    return a*a+f(h->next);
}
```

```
int g(int x) {
    if (x<=1) return 1;
    int b = 1+2*g(x/2);
    return 2*b;
}
```

#### Tempo di $f$ (in funzione della lunghezza $n$ di $h$ )

1° for  $O(n)$   $n^\circ$  iterazioni:  $n$ ; complessità iterazione:  $O(1)$

2° for  $O(n^2)$   $n^\circ$  iterazioni:  $n^2$ ; complessità iterazione:  $O(1)$

$T_f(n) = a$  per  $n=0$

$T_f(n) = b \cdot n^2 + T_f(n-1)$  altrimenti

$T_f(n)$  è  $O(n^3)$

#### Risultato di $f$

$R_f(n) = a$  per  $n \leq 1$

$R_f(n) = b \cdot n^4 + R_f(n-1)$  altrimenti

$R_f(n)$  è  $O(n^5)$

#### Tempo di $g$

$T_g(x) = a$  per  $x \leq 1$

$T_g(x) = b + T_g(x/2)$  altrimenti

$T_g(x)$  è  $O(\log x)$

#### Risultato di $g$

$R_g(x) = a$  per  $x \leq 1$

$R_g(x) = b + 4 \cdot R_g(x/2)$  altrimenti

$R_g(x)$  è  $O(x^2)$

#### Calcolo del for

Numero iterazioni:  $R_g(R_f(n)) = R_g(n^5) = O(n^{10})$

Complessità singola iterazione =  $T_f(n) + T_g(n^5) = O(n^3) + O(\log n) = O(n^3)$

complessità del for:  $O(n^{10}) \cdot O(n^3) = O(n^{13})$

## Esercizio 6

- Indicare l'output del seguente programma c++;
- Le istanze delle classi e funzioni modello vengono create a tempo di compilazione o di esecuzione? **Di compilazione**
- Quali istanze delle classi e funzioni modello vengono create?  
**A<int>, A<double>, B<int>, B<double>, f<A<int>>, f<A<double>>**

```
template <class T>
class B {
T x;
public:
B() { x = 2.3;
      cout << "nuovo B" << endl;
    };
void print () { cout << x << endl; }
};

template <class T>
class A {
B<T>* a;
public:
A() { a = new B<T>;
      cout << "nuovo A" << endl;
    };
};

void print () { a->print(); }
};

template <class T>
void f (T o)
{ o.print ();
};

int main()
{ A<int> obj1;
  A<double>obj2;
  f(obj1);
  f(obj2);
}
```

a)  
nuovo B  
nuovo A  
nuovo B  
nuovo A  
2  
2.3

## Esercizio 7

Indicare per sommi capi un algoritmo che controlla se due sequenze sono l'uno l'anagramma dell'altra.

**Una possibile soluzione:**

**Step 1. Si ordinano le due sequenze separatamente  $O(n \log n)$**

**Step 2. Si controlla se sono uguali confrontando element per element  $O(n)$**

**Complessità:  $O(n \log n)$**