ANNO ACCADEMICO 2015/16

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

6 febbraio 2017

1	2	3	4	5
5	7	7	7	7

Esercizio 1

Sia dato lo heap:

[80 70 60 60 50 10 15]

mostrare lo stato dello stesso e le chiamate ad up e down
a) Dopo l'estrazione di un elemento
b) Dopo l'inserimento del valore 65(a partire dallo heap ottenuto al passo a)

a) [70 60 60 15 50 10] down(0), down(1), down(3)

b) [70 60 65 15 50 10 60] up(6), up(2)

Esercizio 5

a) Cosa sono gli alberi di decisione e a cosa servono.
b) Spiegare come con gli alberi di decisione si arriva ad affermare che mergesort è un algoritmo ottimo.
c) Disegnare l'albero di decisione per il problema di trovare il minimo in una sequenza di tre elementi. Qual è la complessità minima per questo problema che si ricava dall'albero di decisione?

Esercizio 2
Calcolare la complessità dell'espressione g(f(n)) + f(g(n)) in funzione di n (indicando le relazioni di ricorrenza di tempo e risultato per ogni funzione) con le funzioni f e g definite come segue:

```
int f(int x) {
  if (x<=1) return 2;
  int a = f(x/2)+f(x/2);
  cout << a;
  return 1+ x + 4*a;
}
```

```
int g(int x) {
int b=0;
  if (x<=1) return 5;
  for (int i=1, i<=2*f(x);i++)
     b++;
  return 10 + b +g(x-2);
}
```

$T_f(1)= k_1$
$T_f(n)= k_2 + 2T_f(n/2)$ $T_f(n)$ è $O(n)$

$R_f(1)= k$
$R_f(n)= n + 8T_f(n/2)$ $T_f(n)$ è $O(n^3)$

$R_f(1)$ è $O(n^3)$

Calcolo $T_g(n)$
for:
numero iterazioni: $R_f(x) = O(n^3)$
Complessità singola iterazione: $T_f(n) = O(n)$
complessità del for: $O(n^4)$

$T_g(1)= k_1$
$T_g(n)= k_2\ n^4 + T_g(n-2)$ $T_g(n)$ è $O(n^5)$

$R_g(1)= 5$
$R_g(n)= k\ n^3 + T_g(n-2)$ $T_g(n)$ è $O(n^4)$

Tempo di g(f(n))= Tempo per il calcolo di f(n) + tempo per il calcolo di g(n^3) =

O(n)+O(n^15)= O(n^15)

Tempo di f(g(n))= Tempo per il calcolo di g(n) + tempo per il calcolo di f(n^4) =

O(n^5)+O(n^4)= O(n^5)

Tempo per l'esecuzione di g(f(n)) * f(g(n)) = O(n^15) + O(n^5) = O(n^15)

Esercizio 3

Dato un albero generico memorizzato figlio-fratello con etichette intere, cancellare il primo figlio di ogni nodo se è una foglia e ha una etichetta maggiore o uguale della somma delle etichette degli altri figli del nodo (se è l'unico figlio si assume che tale somma sia 0).

```
int sommafratelli(Node*t) {
 if (! t) return 0;
 return t->label+sommafratelli(t->right);
}

void cancella (Node* t) {
   if (! t) return;
   if (! t->left)   { cancella(t->right); return; }
   if (! t->left->left &&
               t->left->label >=sommafratelli (t->left->right)) {
            Node* t1=t->left;
            t->left=t->left->right;
            t1->right=0;
            delete t1;
         }
   cancella(t->left);
   cancella(t->right);
}
```

Esercizio 4

Spiegare il meccanismo delle funzioni virtuali.
Indicare l'uscita del seguente programma con le funzioni chiamate per ogni linea di output.

```
#include <iostream>
using namespace std;

class A {
protected:
int x;
public:
A(){ x=10; cout << 15 << endl;};
void stampa(){ cout << x << endl;};
};


class B: public A {
public:
B(){ x=x+20; cout << 16+x << endl;};
virtual void stampa(){ cout << x << endl;};
};

class C: public B {
public:
C(){ cout << 17 << endl;};
```

```
void stampa(){ cout << 70 + x << endl;};
};

class D: public A {
public:
D(){ x+=9; cout << x << endl;};
void stampa(){ cout << x << endl;};
};

int main(){
A* objB=new B;
C* objC=new C;
A* objC1=objC;
A* objD=new D;
B* objB1= objC;
objB1->stampa();
objB->stampa();
objC1->stampa();
objD->stampa();
}
```

```
15      A()
46      B()
15      A()
46      B()
17      C()
15      A()
19      D()
100     C.stampa()
30      A.stampa()
30      A.stampa()
19      A.stampa()
```