FONDAMENTI DI INFORMATICA II –Algoritmi e Strutture dati a.a 2015/16

14 giugno 2016

1	2	3	4	5
7	6	7	7	6

Esercizio 1

Calcolare la complessità del blocco (indicando le relazioni di ricorrenza di tempo e risultato per ogni funzione) in funzione di n:

```
{
  int a = 0;
  for (int i=0; i <= g(n)/n; i++)
     a += f(n);
}
```

con le funzioni **f** e **g** definite come segue:

```
int f(int x) {
   if (x<=1)
     return 1;
   cout << f(x/3) + f(x/3);
   return f(x/3) + 1;
}
```

```
int g(int x) {
  int a=0;
  for (int i=0; i <= f(x); i++)
     a++;
  for (i=0; i <= 2*f(x); i++)
     a+=i;
  return a;
}
```

Indicare per esteso le relazioni di ricorrenza e, per ogni comando ripetitivo, il numero di iterazioni e la complessità dell'iterazione singola.

Tf(0,1)= k1
Tf(n)= k2 + 3 Tf(n/3) Tf(n) è O(n)
Rf(0,1)= 1
Rf(n)= 1 + Rf(n/3) Rf(n) è O(logn)

Calcolo Tg(n)
Entrambi i for hanno
numero iterazioni: O(logn) Complessità singola iterazione: O(n)
complessità dei for: O(nlogn)

il valore di alla fine e' O((logn)^2) perché a contiene logn + 2 volte la somma dei primi logn numeri
Tg(n) è O(nlogn)
Rg(n) è O((logn)^2)

Calcolo for del blocco:
numero iterazioni: Rg(n)/n = ((logn)^2/n)
Complessità singola iterazione: Tg(n)+ Tf(n)+= O(nlogn) + O(n)=O(nlogn)
Complessità totale blocco = O(((logn)^2)/n * nlogn) = O((logn)^3)

Esercizio 2

Indicare l'output del seguente programma e le istanze delle funzioni template generate dal compilatore.

```
class alpha {
protected:
      int a;
public:
      alpha(){a=8;
          cout << "nuovo alpha  " <<  endl;
        }
      void virtual f()=0;
      void g() {cout <<  a << endl; }
};

class beta: public alpha {
protected:
      int a;
public:
      beta() {a=5;
            cout << "nuovo beta  " << endl;
        }
      void virtual f() { }
      void virtual  g() {cout << a <<  endl;}
};

class delta: public  beta {
protected:
      beta ob;
public:
      delta() {
         cout << "nuovo delta  " << endl;
         }
      void f() { cout  << a << endl;}
      void g(){cout << a+1  << endl;}
};

template<class T>
void funzione( T *obj){
      static int a;
      obj->f();
      obj->g();
      a++;
      cout << a << endl;
}

template<class T1, class T2>
void funzione1( T1 *obj1, T2 *obj2){
      funzione(obj1);
      cout << endl;
      funzione(obj2);
      cout << endl;
```

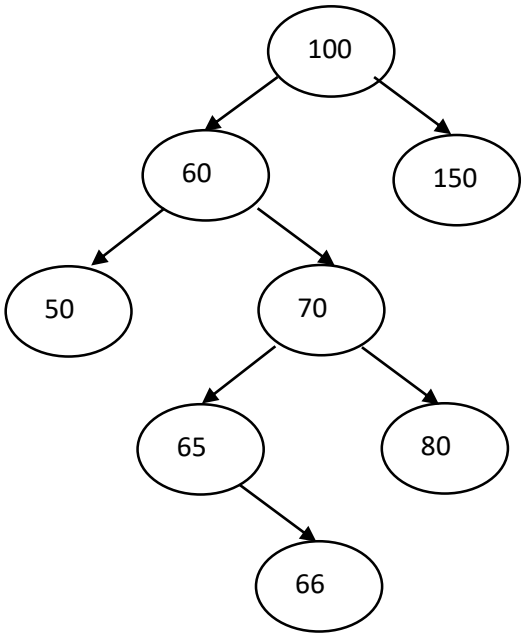
```
      funzione<alpha> (obj2);
      cout << endl;
}

void main(){
      delta *obj1= new delta;
      alpha *obj2=obj1;
      beta *obj3=obj1;
      funzione1(obj3, obj2);
}
```

nuovo alpha
nuovo beta
nuovo alpha
nuovo beta
nuovo delta
5
6
1
5
8
1
5
8
2

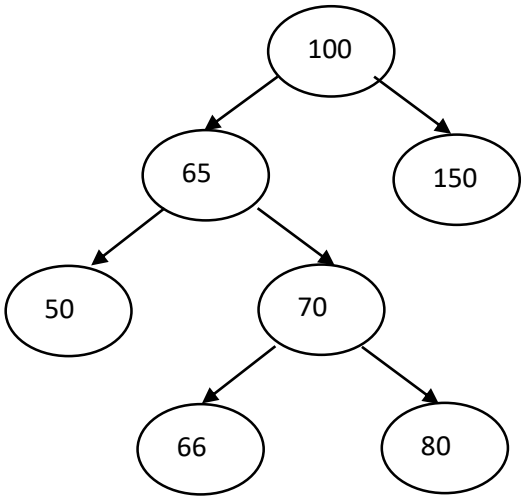
Istanze delle funzioni: `funzione1<beta, alpha>, funzione<beta>, funzione<alpha>`

Esercizio 3

Descrivere a parole il tipo di dato albero binario di ricerca: a) definizione e descrizione delle operazioni con relativa complessità. b) Dato l'albero in figura, indicare l'albero risultante dalla cancellazione del nodo 60. c) Indicare un esempio di albero binario di ricerca con 7 nodi per cui la complessità della ricerca è minima e uno per cui è massima.



b)



Esercizio 4

Scrivere una funzione **`void potatura(Node* t)`** che, dato un albero generico, cancella il secondo sottoalbero di ogni nodo, se esiste. Si supponga che l'albero generico sia non vuoto e memorizzato figlio-fratello. Indicare la complessità.

```
void potatura (Node* t) {
if (!t) return;
if (t->left) && (t->left->right) {
   Node* temp= t->left->right;
   t->left->right= t->left->right->right;
   temp->right=NULL;
   deltree(temp);
   }
potatura (t->left);
potatura (t->right);
}


void deltree (Node*  &t) {
if (!t) return;
deltree (t->left);
deltree (t->right);

delete (t);
t=NULL;
}
```

Esercizio 5

Descrivere l'algoritmo di compressione di Huffmann: a cosa serve, qual è la sua complessità e come viene calcolata.

Applicarlo al seguente alfabeto con le frequenze indicate, indicando l'albero risultante con la convenzione che per ogni nodo con figli il nodo minore sta a sinistra e corrisponde a 0.

carettere	frequenza
A	9
B	37
C	21
D	20
E	8
F	5

carettere	codifica
A	100
B	11
C	01
D	00
E	1011
F	1010