ANNO ACCADEMICO 2019/2020 - Algoritmi e Strutture Dati
22 settembre 2020
Gruppo B

n. quesito	1	2	3	4	voto
punti	8	8	8	9	33

quesito 1

Calcolare la complessità del seguente comando con le funzioni definite come segue. Indicare per esteso le relazioni di ricorrenza del tempo e del risultato di entrambe le funzioni e, per ogni comando ripetitivo, il numero di iterazioni e la complessità della singola iterazione.

```
for (int i=1; i <= f(f(n)); i++) cout << g(n);

int f(int x) {
     if (x==0) return 2;
     for (int i=1; i <=x; i++) cout << i;
     return 2*x*x+ f(x-1);
}
int g(int x) {
     if (x<=1) return 1;
     int a=f(x)+f(x); int b=0;
     for (int i=1; i <=x; i++) b++;
     return a+f(x)+4*b;
}
```

Funzione f

Calcolo del for
numero iterazioni: O(n)
complessità della singola iterazione O(1)
complessità del for: O(n)

$T_f(0)= d$
$T_f(n)= cn + T_f(n-1)$ $T_f(n)$ è $O(n^2)$

$R_f(0)= 2$
$R_f(n)= 2n^2+ R_f(n-1)$ $R_f(n)$ è $O(n^3)$

Funzione g

Calcolo del for
numero iterazioni: O(n)
complessità della singola iterazione O(1)
complessità del for: O(n)

$T_g(n) = O(n^2)+O(n) + O(n^2) = O(n^2)$

$R_g(n)$ è $O(n^3)$

Calcolo del comando:
numero iterazioni: $R_f(R_f\ f(n)) = R_f(n^3)= O(n^9)$
complessità della singola iterazione: $T_f(n) + T_g(n)+ T_f(R_f(n)) = O(n^2)+O(n^2)+ T_f(n^3) = O(n^2)+O(n^2)+ O(n^6) = O(n^6)$
complessità del for: $O(n^9) * O(n^6)= O(n^{15})$

quesito 2

Scrivere una funzione `int somma(Node* t)`, che, dato un albero generico `t` memorizzato figlio/fratello con etichette intere, fa la sommatoria delle etichette di tutti i nodi per i quali tutti i figli hanno la stessa etichetta (se un nodo non ha figli la condizione è vera).

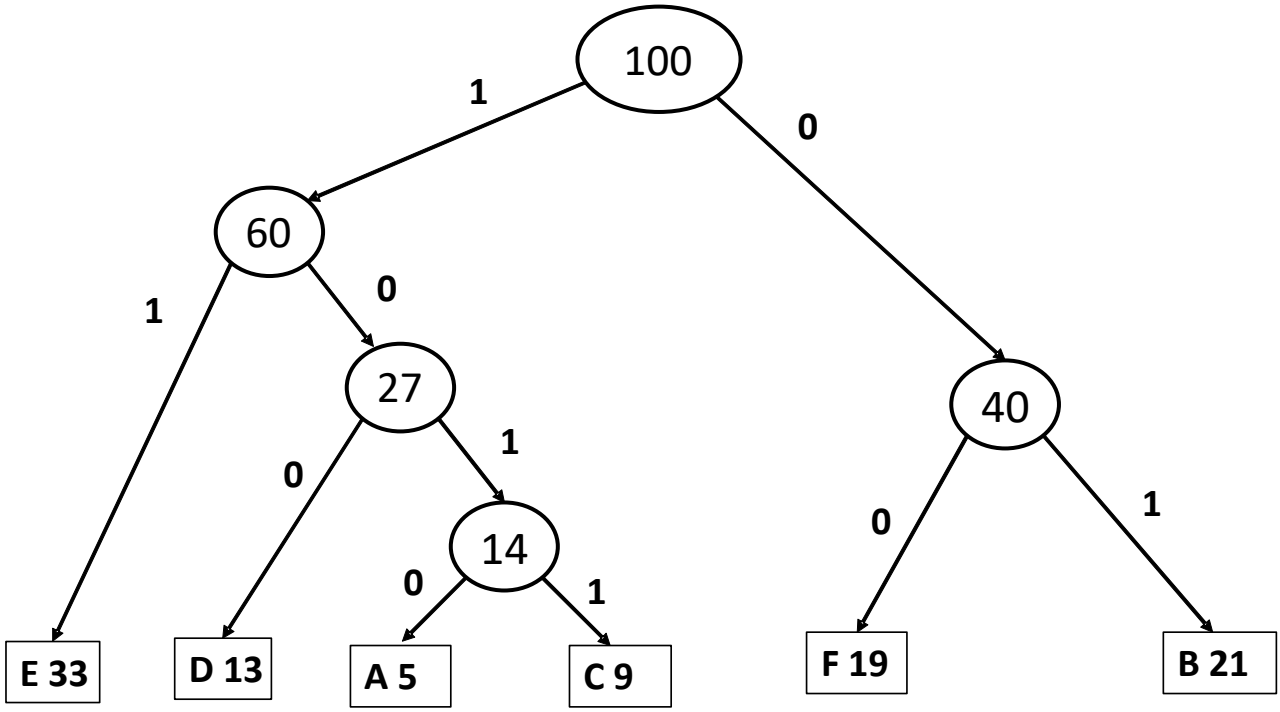
```
int somma (Node* t) {
   if (!t) return 0;
   if (!t->left) return t->label + somma(t->right);
   if (check(t->left->label, t->left->right) )
      return t->label + somma(t->left) + somma(t->right);
   return somma(t->left) + somma(t->right);
}


bool check (int a, Node* t) {
   if (!t) return true;
   if (t->label!=a) return false;
   return check_list (a,t->right);
}
```

quesito 3:

a) 1 A cosa serve l'algoritmo di Huffmann? Su quale metodologia è basato?
b) 2 Qual è la sua complessità e come si calcola?
c) 1 Cosa vuol dire che un codice ha la proprietà di prefisso?
d) 1 Che caratteristica ha l'albero che è l'output dell'algoritmo?
e) 3 Applicare l'algoritmo all'alfabeto seguente dove, per ogni carattere, è indicata la sua frequenza nel testo. Indicare l'albero risultante e le relative codifiche.

A	5
B	21
C	9
D	13
E	33
F	19



quesito 4

a) 3 Descrivere le classi P e NP, il teorema di Cook, i problemi NP-completi.
b) 3 Descrivere l'algoritmo di ordinamento heapsort e la sua complessità. Perchè si chiama così?
c) 3 Perchè non è possibile compilare separatamente funzioni e classi template in C++? In base a cosa viene scelto quali istanze compilare? Fare un esempio di istanziazione con parametri impliciti e un esempio di istanziazione con parametric espliciti.