**ANNO ACCADEMICO 2019/2020 - Algoritmi e Strutture Dati - 16 giugno 2020**
**Gruppo A**

| n. quesito | 1 | 2 | 3 | 4 | tot |
|---|---|---|---|---|---|
| punti | 8 | 8 | 8 | 9 | 33 |

**quesito 1**
Calcolare la complessità del seguente comando in funzione di n:

```
for (int i=0; i <= g(n)/n; i++) cout << f(n);
```

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

```
int f(int x) {
   if (x<=1)
     return 1;
   cout << f(x/3) + 2* 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 <= x*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 della singola iterazione.

$T_f(0,1)= d$
$T_f(n)= c + 3\ T_f(n/3)$ $T_f(n)$ è $O(n)$

$R_f(0,1)= 1$
$R_f(n)= 1 + R_f(n/3)$ $R_f(n)$ è $O(\log n)$

Calcolo $T_g(n)$
1 for:
numero iterazioni: $O(\log n)$ Complessità singola iterazione: $O(n)$
complessità dei for: $O(n \log n)$

2for:
numero iterazioni: $O(n^2)$ Complessità singola iterazione: $O(1)$
complessità dei for: $O(n^2)$

il valore di a alla fine e' $O(n^4)$

$T_g(n)$ è $O(n \log n) + O(n^2)= O(n^2)$
$R_g(n)$ è $O(n^4)$

Calcolo for del blocco:
numero iterazioni: $R_g(n)/n= O(n^3)$
Complessità singola iterazione: $T_g(n)+ T_f(n)+= O(n^2) + O(n)=O(n^2)$
Complessità totale blocco = $O(n^5)$

## quesito 2

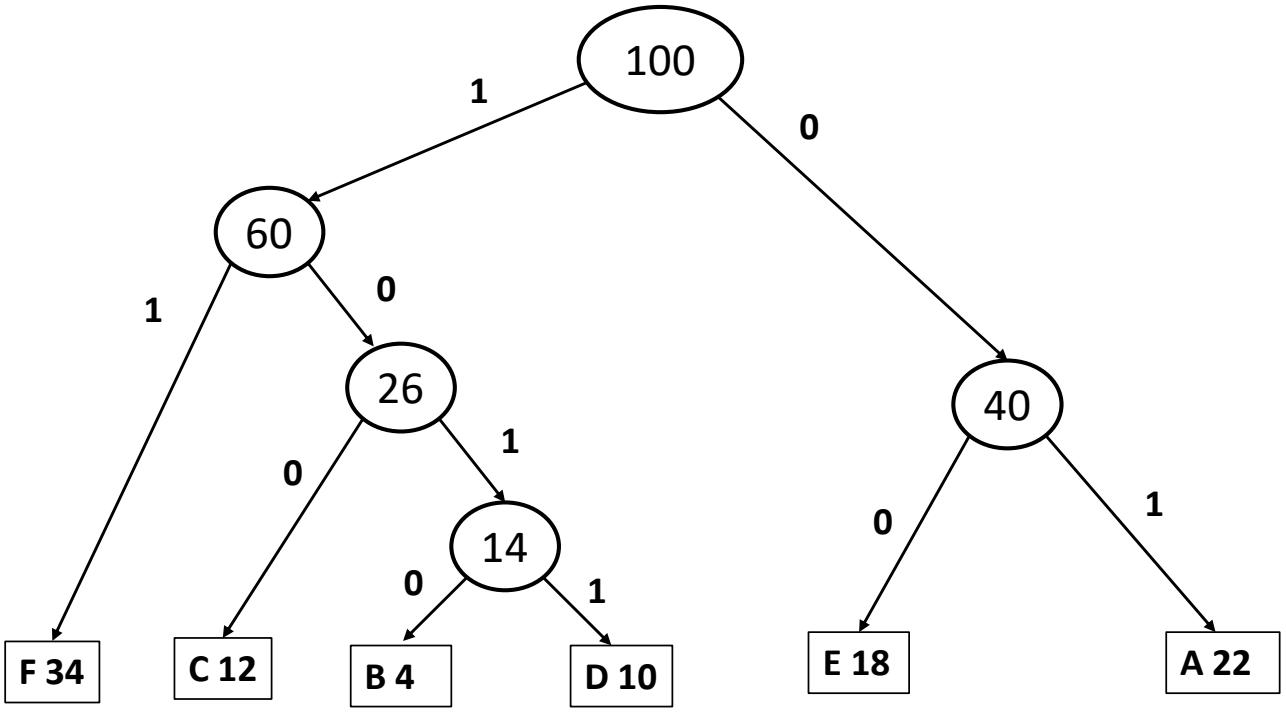
Scrivere una funzione **`void aggiungi(Node* t)`**, che, dato un albero generico `t` con etichette intere memorizzato figlio/fratello, aggiunge ad ogni nodo un primo figlio con etichetta uguale alla somma delle etichette dei figli (uguale a 0 se il nodo non ha figli).

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

void figlio (Node* &t) {
   Node* t1= new Node;
   t1->left=0;
   t1->label= somma (t);
   t1->right=t;
   t=t1;
};

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

## quesito 3

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

| | |
|---|---|
| **A** | **22** |
| **B** | **4** |
| **C** | **12** |
| **D** | **10** |
| **E** | **18** |
| **F** | **34** |




**quesito 4**

a) 3 Elencare tutti gli algoritmi di ordinamento visti a lezione con la loro complessità nel caso medio e nel caso peggiore.

b) 3 Qual è il limite inferiore alla complessità degli algoritmi di ordinamento in base al metodo degli alberi di decisione? Ci sono algoritmi di ordinamento che non rispettano questo limite? Come mai?

c) 3 Fare un esempio di gerarchia con due classi con una funzione **`fun`** nella classe base e un **`main`** che mostra un diverso comportamento se la funzione **`fun`** è **`virtual`** oppure no. Spiegare.