ANNO ACCADEMICO 2019/2020 - Algoritmi e Strutture Dati - 7 luglio 2020

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

quesito 1

Calcolare la complessità del seguente comando in funzione del numero di nodi dell'albero binario quasi bilanciato t:

```
for (int i=0; i <= g(t)*f(t); i++) cout << g(t);
```

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

```
int g (Node* t ) {
 if (!t) return 1;
 t->label+= f(t);
 cout << g (t->left);
 return 1+3*g(t->left)+ g(t->right);
}
```

```
int f(Node* t) {
  if (!t) return 1;
  for (int i=0; i <= 2; i++)
     cout << f(t->left);
  return 1 + 2*f(t->right);
}
```

Funzione f

Numero di iterazioni del for: 3
Complessità della singola iterazione: T(n/2)
Complessità del for: 3 T(n/2)

$T_f(0)= d$
$T_f(n)= c + 4T_f(n/2)$ $T_f(n)$ è O(n^2)

$R_f(0)= 1$
$R_f(n)= 1+ 2R_f(n/2)$ $R_f(n)$ è O(n)

Funzione g

$T_g(0)=d$
$T_g(n)= cn^2+ 3\ T_g(n/2)$ $T_g(n)$ è O(n^2)

$R_g(0)=d$
$R_g(n)= c + 4\ R_g(n/2)$ $R_g(n)$ è O(n^2)

Calcolo del comando:
numero iterazioni: $R_f(n) * R_g(n)$ = O(n) * O(n^2) = O(n^3)

complessità della singola iterazione: $T_f(n) + T_g(n) = O(n^2) + O(n^2) = O(n^2)$
complessità del for: $O(n^5)$

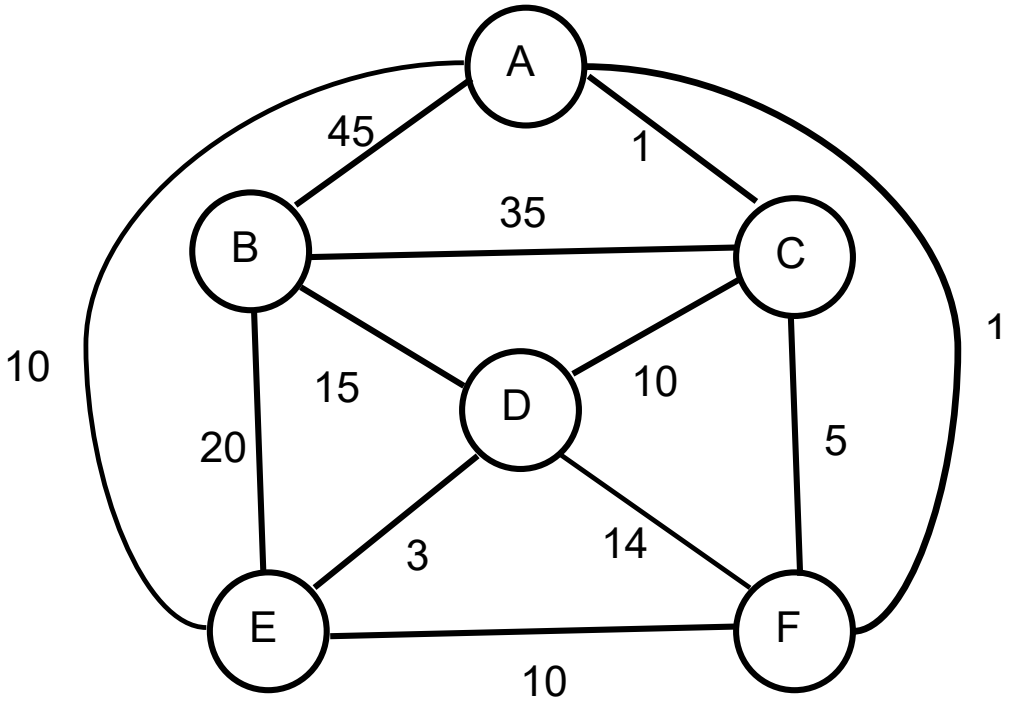
quesito 2
Scrivere una funzione `void somma(Node* t)`, che, dato un albero generico `t` con etichette intere memorizzato figlio/fratello, somma ad ogni nodo l'etichetta dell'ultimo figlio del suo ultimo figlio, se esiste.

```
Node* ultimo_fratello(Node*t){
   if (!t) return 0;
   if (!t->right) return t;
   return ultimo_fratello (t->right);
}

void somma (Node* t) {
   if (!t) return;
   if (t->left) {
   Node* t1= ultimo_fratello(t->left);
   Node* t2= ultimo_fratello(t1->left);
   if (t2) t->label += t2->label;
   }
   somma(t->left);
   somma(t->right);
}
```

quesito 3

a) 1 A cosa serve l'algoritmo di Dijkstra?
b) 3 Qual è la sua complessità e come si calcola?
c) 1 Si può applicare a tutti grafi? NO: soltanto a quelli con peso positivo sugli archi
d) 4 Applicarlo al grafo in figura a partire dal nodo B. Mostrare tutti i passaggi in una tabella fatta come nel disegno. Indicare i cammini minimi.



Insieme Q	Nodo scelto	A	B	C	D	E	F
A,B,C,D,E, F	-	inf | -	0 | -	inf | -	inf | -	inf | -	inf | -
A,C,D,E, F	B	45| B	0 | -	35| B	15| B	20| B	inf | -
A,C,E, F	D	45| B	0 | -	25| D	15| B	18| D	29| D
A,C,F	E	28| E	0 | -	25| D	15| B	18| D	28| E
A,F	C	26| C	0 | -	25| D	15| B	18| D	28| E
F	A	26| C	0 | -	25| D	15| B	18| D	27| A

Cammini minimi: BCDA, BDC, BD, BDE, BDCAF

quesito 4

a) 2 Descrivere brevemente tutti gli algoritmi di **ricerca** (esclusa la ricerca hash) visti a lezione indicando la struttura a cui sono applicati e la complessità nel caso medio e nel caso peggiore.
b) 1 Qual è il limite inferiore al problema della ricerca stabilito dagli alberi di decisione? La ricerca hash rispetta questo limite? Spiegare. O(logn); la ricerca hash non lo rispetta ma non è basata su confronti.
c) 2 Indicare per sommi capi un algoritmo con complessità minore possibile che, data una lista semplice, elimina dalla lista tutti i doppioni, in modo che ogni elemento compaia una sola volta. E' possibile modificare l'ordine degli elementi nella lista iniziale. Si ordina la lista con mergesort e poi la si scorre eliminando i doppioni (che sono adiacenti): O(nlogn) + O(n)= O(nlogn)
d) 3 Fare un esempio di programma dove una eccezione viene lanciata all'interno di un blocco try appartenente a una una funzione ma viene gestita (correttamente) dal main.