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

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

quesito 1

Calcolare la complessità del seguente comando in funzione di n>=0. 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(n); i++) cout << g(n)+f(f(n));
```

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.

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

Funzione f

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

$T_f(0)= d$
$T_f(n)= cn + T_f(n/2)$ $T_f(n)$ è O(n)

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

Funzione g

Calcolo del for

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

$T_g(0,1)=d$
$T_g(n)= cn^2+ T_g(n/2)$ $T_g(n)$ è $O(n^2)$

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

Calcolo del comando:
numero iterazioni: $R_f(f(n)) = R_f(n^2)$
complessità della singola iterazione: $T_f(n) + T_g(n)+ T_f(R_f(n)) = O(n)+O(n^2)+ T_f(n^2)= O(n)+O(n^2)+ O(n^2) = O(n^2)$
complessità del for: $O(n^4)$

quesito 2

Scrivere una funzione che, dato un albero generico `t` memorizzato figlio/fratello con etichette intere, restituisce la sommatoria delle etichette di tutti i nodi che hanno la stessa etichetta del padre.

```
int somma (Node* t, Node* p=NULL) {
   if (!t) return 0;
   if (p=NULL) return somma(t->left, t);
                    // t è il puntatore alla radice dell'albero
   if(t->label==p->label)
            return t->label + somma(t->left, t) + somma(t->right, p);
   return somma(t->left, t) + somma(t->right, p);
}
```

quesito 3

A. 3 Descrivere il metodo di ricerca hash: accesso diretto, collisioni, agglomerati, metodi di scansione, metodo di concatenazione. Da cosa dipende l'efficienza del metodo?

B. 3 Dato un insieme di al massimo 30 elementi inizialmente vuoto, considerare le seguenti operazioni

a) Inserimento dell'elemento 61
b) Inserimento dell'elemento 62
c) Inserimento dell'elemento 183
d) Cancellazione dell'elemento 61
e) Inserimento dell'elemento 122

e mostrare il loro effetto con la memorizzazione mediante un array di 61 posizioni (indici da 0 a 60) con indirizzamento con il metodo del resto e scansione lineare unitaria: indicare il contenuto delle prime 3 celle dell'array, inizialmente vuoto, dopo ognuna delle operazioni.

C. 2 Qual è il fattore di carico? **Circa ½ (30/61).** Si è formato un agglomerato? **SI**

indice	**inizio**	**Dopo a)**	**Dopo b)**	**Dopo c)**	**Dopo d)**	**Dopo e)**
0	**-1**	**61**	**61**	**61**	**-2**	**122**
1	**-1**	**-1**	**62**	**62**	**62**	**62**
2	**-1**	**-1**	**-1**	**183**	**183**	**183**

quesito 4

a) 3 Cosa sono gli alberi di decisione e a cosa servono? Considerare la loro applicazione agli algoritmi di ricerca e a quelli di ordinamento.
b) 3 Descrivere l'algoritmo quicksort e indicare la sua complessità nel caso medio e nel caso peggiore. Confrontarlo con l'algoritmo mergesort.
c) 3 Considerare la seguente gerarchia di classi in C++:

```
class alpha {
protected:
        int a;
public:
alpha(){a=10; }
void virtual f()=0;
};

class beta: public alpha {
protected:
        int a;
public:
beta() {a=20;  }
        void f() { cout << a;}
};
```

Con le classi definite come sopra, date le seguenti istruzioni:

```
beta *p1=new beta;
alpha *p2=new beta;
```

dire quali sono giuste e quali errate fra le seguenti istruzioni, dandone la motivazione:

1. **alpha *p3=p1; corretto (conversione da sottoclasse a superclasse)**
2. **beta *p3=p2; non corretto (conversione da superclasse a sottoclasse)**
3. **alpha obj3; non corretto (istanziazione classe astratta)**
4. **cout << p2->a; non corretto (a è protected)**