Algoritmi e Strutture dati - ANNO ACCADEMICO 2016/17

4 luglio 2017

1	2	3	4	5	6

Esercizio 1

a) Descrivere l'algoritmo PLSC: a cosa serve, su quale ragionamento è basato, come è implementato, qual è la sua complessità e come viene calcolata (scrivere sul retro del foglio).
b) Applicarlo per trovare la/le PLSC fra le due sequenze: xxyzxy e xyzzyx.

		x	y	z	z	y	x
	0	0	0	0	0	0	0
x	0	1	1	1	1	1	1
x	0	1	1	1	1	1	2
Y	0	1	2	2	2	2	2
z	0	1	2	3	3	3	3
x	0	1	2	3	3	3	4
y	0	1	2	3	3	4	4

PLSC: xyzx xyzy

Esercizio 2

Scrivere una funzione **`void sort (Node *tree1, Node *tree2, Elem * & list)`** che, dati due alberi binari di ricerca, costruisce una lista semplice ordinata in ordine crescente contenente tutti gli elementi dei due alberi.

```
void costruisci ( Node *tree, Elem * & list){
      if (tree) {
      costruisci(tree->right, list);
      Elem * list1= new Elem;
      list1->inf=tree->label;
      list1->next=list;
      list=list1;
      costruisci(tree->left, list);
      }
}
void sort (Node *tree1, Node *tree2, Elem * & list){
      list=NULL;
      Elem * list2=NULL;
      costruisci(tree1, list);     // O(n/2)
      costruisci(tree2, list2);    // O(n/2)
      merge(list, list2);     // O(n)
}
```

O(n)

Esercizio 3

Scrivere una funzione che, dato un albero generico con etichette intere memorizzato figlio-fratello, conta il numero delle foglie che sono secondi figli di un nodo.

```
int check(Node* tree) ){
      if (!tree || !tree->right) return 0;
      if (tree->right->left ==0 ) return 1;
                    else return 0;
      }


int conta ( Node *tree){
      if (!tree) return 0;
      return (check(tree->left) + conta(tree->left)+conta(tree->right);
      }
```

Esercizio 4

Calcolare la complessità in funzione di **n**>0 delle istruzioni `y=g(f(n));` **e** `y= f(g(n));`

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

```
int f(int x) {
   if (x<=1) return 1;
   int b=0, a=0, i;
   for (i=1; i<=3; i++) b+=i;
   for (i=1; i<=x; i++) a+=i;
   cout << b*a;
   return f(x-1) + 4 + b;
}
```

```
int g(int x) {
  if (x<=1) return 10;
  int a=0;
  for (int i=0; i<f(x)*f(x); i++)
      a++; cout << a;
  a=0;
  for (int i=0; i< x; i++)
      a++;
   return a+g(x/2)+g(x/2);
}
```

Indicare le eventuali relazioni di ricorrenza e spiegare brevemente il calcolo della complessità dei cicli.

Stima del tempo di f

numero iterazioni del 1° for = O(1)
complessità di un'iterazione = costante
tempo del 1° for = O(1)

numero iterazioni del 2° for = O(n)
complessità di un'iterazione = costante
tempo del for = O(n)

$T_f(1)= d$ **T_f è O(n^2)**
$T_f(n)= c\ n + T_f(n-1)$

$R_f(1) =1$ **R_f(n)è O(n)**
$R_f(n) = c + R_f(n-1)$

Stima del tempo di g:

numero iterazioni del 1° for = O(m^2)
complessità di un'iterazione = O(m^2)
tempo del for = O(n^4)

numero iterazioni del 2° for = O(m)
complessità di un'iterazione = O(1)
tempo del for = O(m)

tempo di g

$T_g(1)$= d **T_g è O(m^4)**
$T_g(m)$= c· m^4 + 2T_g(m/2)

stima del risultato di g

$R_g(1)$= cost **$R_g(m)$ = O(mlogm)**
$R_g(m)$= cm +2 R_g(m/2)

Complessità dell'istruzione `y=g(f(n))`:

C[f(n)]+ C[g(R_f(n))] = O(n^2) + C[g(n)] = O(n^2) + O(n^4) = O(n^4)

Complessità dell'istruzione `y=f(g(n))`:

C[g(n)]+ C[f(R_g(n))] = O(n^4) + C[f(nlogn)] = O(n^4) + O(n^2log^2n) = o(n^4)

Esercizio 5

a) Descrivere il tipo di dato heap con le sue operazioni e relative complessità (scrivere sul retro del foglio).

b) Dato lo heap

[88, 74, 57, 15, 30, 33, 55, 10]

Indicare
i. il contenuto dello heap dopo l'inserimento del valore 78
ii. il contenuto dello heap dopo una successiva estrazione

Indicare le chiamate a `up` e `down` che vengono eseguite in questi casi.

heap	chiamate
88, 74, 57, 15, 30, 33, 55, 10	
i. 88, 78, 57, 74, 30, 33, 55, 10, 15	up(8), up(3), up(1)
ii. 78, 74, 57, 15, 30, 33, 55, 10	down(0) , down(1), down (3)

Esercizio 6 Dato il programma seguente:

```
class A {
public:
int x=0;
A(){cout << "new A" << endl;};
};

class B: public A {
public:
int x=1;
B(){cout << "new B" << endl;};
};

class C: public A {
public:
int x=2;
C(){cout << "new C" << endl;};
};

template<class T1, class T2>
class D {
T1* oggetto1= new T1;
T2* oggetto2= new T2;
public:
D(){ cout << "new D" << endl;};
void f() {try
            {
*           T1* oggetto3= oggetto1;
             cout << "messaggio1" <<
endl;
             if (oggetto3->x!=0)
throw 0;
             T2* oggetto4= oggetto2;
             cout << "messaggio2" <<
endl;
             }
           catch(int){cout << "diverso
da 0" << endl;}
          cout << "fine f" << endl;
          };
};

    int main(){
    int i;
    D<B, C>* obj1 = new D<B, C>;
    obj1->f();
    D<B, B>* obj2 = new D<B, B>;
    obj2->f();
    cin >> i;}
```

a) indicare il suo output
b) indicare il suo output se si sostituisce l'istruzione asteriscata con :
   `A* oggetto3= oggetto1;`
c) dire cosa succede se si sostituisce l'istruzione asteriscata con :
   `T2* oggetto3= oggetto1;`

Spiegare brevemente le differenze.

a)

```
new B
new A
new C
new D
messaggio1
diverso da 0
fine f
new A
new B
new A
new B
new D
messaggio1
diverso da 0
fine f
```

```
new A
new B
new A
new C
new D
messaggio1
messaggio2
fine f
new A
new B
new A
new B
new D
messaggio1
messaggio2
fine f
```

b)

a) Il compilatore segnala un errore perchè per incompatibilità di tipo (B con C). nel caso b) invece c'è compatibilità perchè il tipo della variabile a cui si assegna il puntatore all'oggetto (A) è un supertipo del tipo dell'oggetto (B).