Algoritmi e Strutture dati - ANNO ACCADEMICO 2017/18

24 luglio 2018

1	***2***	***3***	***4***	***5***	***6***	***7***
5	5	5	6	5	3	4

Esercizio 1. Scrivere una funzione in c++ , che, dato un albero binario, conta i nodi tali per cui il sottoalbero sinistro è di livello maggiore del sottoalbero destro.

```
int conta (Node* t, int $ level) {
     if (!t) {level=-1; return 0;}
     int l, r;
     int conta_l =conta(t->left, l);
     int conta_r =conta(t->right,r);
     level=max(l,r)+1;
     return conta_l+conta_r + (l>r);
     }
```

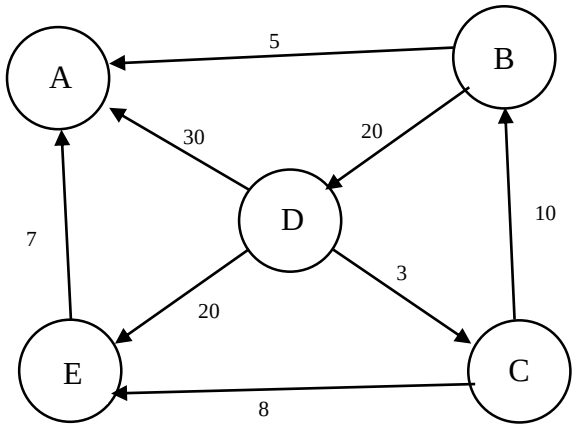
Esercizio 2

a) Descrivere la differenza fra albero binario e albero generico. Con due nodi A e B quanti alberi binari si possono fare? **4** E quanti generici? **2** Disegnarli.
b) Scrivere una funzione in c++ che, dato un albero generico memorizzato figlio-fratello, restituisce il numero dei nodi che hanno etichetta uguale a quella del padre.

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

Esercizio 3

a) Dare la definizione di grafo orientato.
b) Descrivere i metodi di memorizzazione di un grafo orientato visti a lezione.
c) Descrivere l'algoritmo di Dijkstra: 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).
d) Applicarlo al grafo seguente a partire dal nodo D:



	A	B	C	D	E
A, B, C, D, E	inf | -	inf | -	inf | -	0 | -	inf | -
A, B, C, E	30 | D	inf | -	3 | D	0 | -	20 | D
A, B, E	30 | D	13 | C	3 | D	0 | -	11 | C
A, B	18 | E	13 | C	3 | D	0 | -	11 | C
A	18 | E	13 | C	3 | D	0 | -	11 | C

Cammini minimi**:** DCEA (o DCBA), DCB, DC, DCE

Esercizio 4

a) Dimostrare, utilizzando le regole di semplificazione delle espressioni O-grande, che la funzione f(n)= 7n^2 +5n +4 è O(n^2)

b) Calcolare la complessità in funzione di n dell'istruzione

```
y=g(f(n));
```

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

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

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

Funzione g

numero iterazioni del for: O(n^2)
complessità di una iterazione: O(1)
tempo del for: O(n^2)

tempo di g
T(1)= d
T(m)= c n^2 + 2T(n/2) O(n^2)

Risultato di g
R(1)= d
R(m)= c n^4 + 2T(n/2) O(n^4)

Funzione f

numero iterazioni del for: O(n^4)
complessità di una iterazione: O(n^2)
tempo del for: O(n^6)

tempo di f
T(1)= d
T(n)= cn^6 + T(n-1) O(n^7)

Risultato di f O(n^4)

Complessità dell'istruzione:
C[f(n)]+ C[g(n^4)] = O(n^7) + O(n^8) = O(n^8)

Esercizio 5

a) Indicare l'output del seguente programma c++;

```
class R {
protected:
int z; int y;
public:
R() { z = 5; y=2;
     cout << "nuovo R" << endl;
     };
void print(){
     cout << z << endl;
     cout << y << endl;
     z++;
     y++;
    }
};
class S : public R {
protected:
int y;
public:
S() { z++; y=3;
     cout << "nuovo S" << endl;
    };
void print (){
       cout << y << endl;
       y++;
       z++;
     };
void virtual print1()=0;
};
class U: public S {
public:
U() { z=11; y=7;
     cout << "nuovo U" << endl;
    };
void print (){
       cout << z << endl;
       cout << y << endl;
       y++;
       z++;
       };
void print1(){
       cout << z << endl;
       cout << y << endl;
}
};
int main()
{ U* obj =new U;
  S* obj1= obj;
  R* obj2= obj1;
  obj->print();
  obj1->print();
  obj2->print();
  obj1->print1();
}
```

nuovo R
nuovo S
nuovo U
11
7
8
13
2
14
9

b) Quale di queste istruzioni può essere aggiunta in coda al main? Spiegare.

`S* obj3 = new S;`	NO: non si può istanziare una classe astratta
`R* obj4= new U;`	OK: conversion da sottoclasse a superclasse
`cout << obj2->z;`	NO: z è protetta
`obj1=obj2;`	NO: conversione non permessa da superclasse a sottoclasse
`obj2=obj1;`	OK: conversione da sottoclasse a superclasse

Esercizio 6
Indicare per sommi capi un algoritmo con complessità minore di O(n^2) che cancella da una lista semplice i doppioni.

Una possibile soluzione:
Step 1. Si ordina la lista O(nlogn)
Step 2. Si scorre la lista cancellando ogni elemento uguale al precedente O(n)
Complessità: O(nlogn) + O(n) = O(nlogn)

Esercizio 7
Spiegare brevemente la teoria della NP-completezza: gli insiemi P e NP, il teorema di Cook, i problem NP-completi.