ANNO ACCADEMICO 20016/2017- 16 gennaio 2018
Algoritmi e Strutture Dati

1	2	3	4	5	6
6	5	6	6	5	5

Esercizio 1

a) Descrivere l'algoritmo di Kruskal: a cosa serve, come funziona, qual è la sua complessità e come viene calcolata.

b) Si consideri la memorizzazione di un albero generico (senza etichette) mediante un array come nell'algoritmo di Kruskal, in cui cioè ogni elemento dell'array è un nodo e contiene l'indice del padre. I nodi sono numerati da 0 a n-1 e t[n]=-1 se n è la radice. Supponendo di avere un array `tree` con n elementi, scrivere dei frammenti di codice per le seguenti richieste.

1. Stampare il padre del nodo x
2. Stampare tutti figli del nodo x
3. Stampare tutti gli antenati del nodo x
4. Stampare tutti i fratelli del nodo x

SOLUZIONE

```
1. if (t[x]!=-1) cout << t[x];

2. for (int i=0; i<n; i++) if (t[i] ==x) cout << i;

3. for (int i=t[x]; i!=-1; i=t[i]) cout << i;

4. for (int i=0; i<n; i++) if (t[i] ==t[x] && i!=x) cout << i;
```

Esercizio 2
Calcolare la complessità del blocco in funzione del numero di nodi dell'albero binario t:

```
{
  int a = 0;
  for (int i=0; i <= Nodes(t); i++)
     {a += f(t);
      es(t);
     }
}
```

con le funzioni **f** e **es** definite come segue. Indicare per esteso le relazioni di ricorrenza e, per ogni comando ripetitivo, il numero di iterazioni e la complessità della singola iterazione.

```
void es (Node* t ) {
if (!t) return;
t->label*= f (t);
es (t->left);
es (t->left);
es (t->right);
es (t->right);
    }
```

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

Funzione f

Numero di iterazioni del for: n
Complessità della singola iterazione: O(n)
Complessità del for: =O(n^2)

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

Funzione es

$T_{es}(0)$=d
$T_{es}(n)$= cn^2+ 4 $T_{es}(n/2)$ $T_{es}(n)$ è O(n^2logn)

Calcolo for del blocco:
numero iterazioni: n
Complessità della singola iterazione: $T_{nodes}(n) + T_f(n) + T_{es}(n)$ = O(n) + O(n^2) + O(n^2logn) = O(n^2logn)
Complessità del for: O(n^3logn)

Esercizio 3
Scrivere una funzione che, dato un albero binario con etichette intere, restituisce il numero di nodi dell'albero che hanno almeno un nodo con etichetta 100 in entrambi i sottoalberi.

```
int conta (Node* t, bool & cento) {
if (!t) { cento=false; return 0; }
int conta_l, conta_r;
bool cento_l, cento_r;
conta_l=conta(t->left, cento_l);
conta_r=conta(t->right, cento_r);
cento = (cento_l || cento_R || t->label == 100);
return conta_l + conta_r + (cento_l && cento_r)
}
```

Esercizio 4

Scrivere una funzione **`void esau(Node* t)`** che, dato un albero generico memorizzato con la memorizzazione figlio-fratello, scambia il primo con il secondo sottoalbero di ogni nodo che ha almeno due sottoalberi.

```
void esau (Node* t) {
if (!t) return;
if (t->left && t->left->right) {
  Node* temp= t->left->right;
  t->left->right= temp->right;
  temp->right=t->left;
  t->left=temp;
  }
esau (t->left);
esau (t->right);
}
```

Esercizio 5

a) Eseguire il seguente programma e indicare le istanze della funzione f generate

```
#include<iostream>
using namespace std;
template<class tipo1, class tipo2>
class uno {
        tipo1 a;
        tipo2 b;
public:
        uno(tipo1 x, tipo2 y) { a=x; b=y; }
        tipo1 valore_a() { return a; }
        tipo2 valore_b() { return b; }
};
template<class T1, class T2>
void f(uno<T1,T2> obj){
         cout << obj.valore_a();
         cout << obj.valore_b();
}
int main() {
        uno<int, int> obj1(5,6);
        uno<char, char> obj2('m', 'n');
        uno<char, int> obj3('a', 7);
        f(obj1);            // L1
        f(obj2);            // L2
        f(obj3);            // L3
}
```

5 6 m n a 7

Istanze di f:
L1: void f<int,int>
L2: void f< char, char>
L3: void f< char, int>

b) Cosa cambia se la funzione f viene sostituita con la seguente funzione?

```
template<class T>
void f(uno<T,T> obj){
       cout << obj.valore_a();
       cout << obj.valore_b();
}
```

Errore a tempo di compilazione nell'istruzione L3 (chiamata f(obj3)); non si ricava il tipo T (non c'è conversion di tipo)

Esercizio 6

Descrivere la teoria della NP-completezza: cosa sono P e NP, il teorema di Cook, cosa vuol dire che un problema è NP-completo. Fare un esempio di problema NP-completo.