Fondamenti di informatica II – modulo Algoritmi e Strutture dati – a.a. 2011/2012

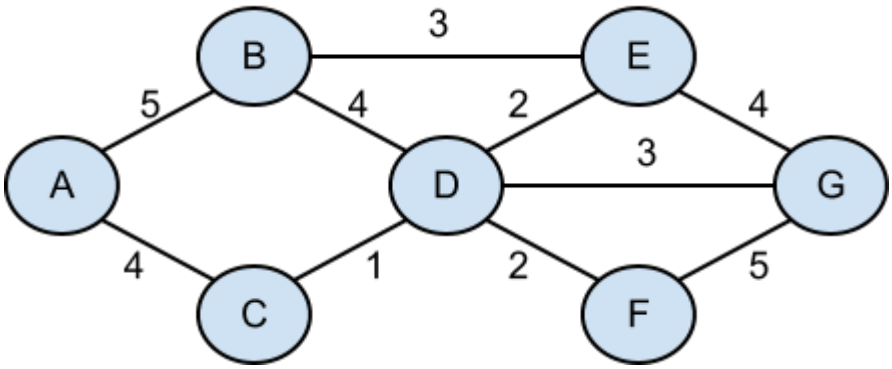
17 settembre 2012

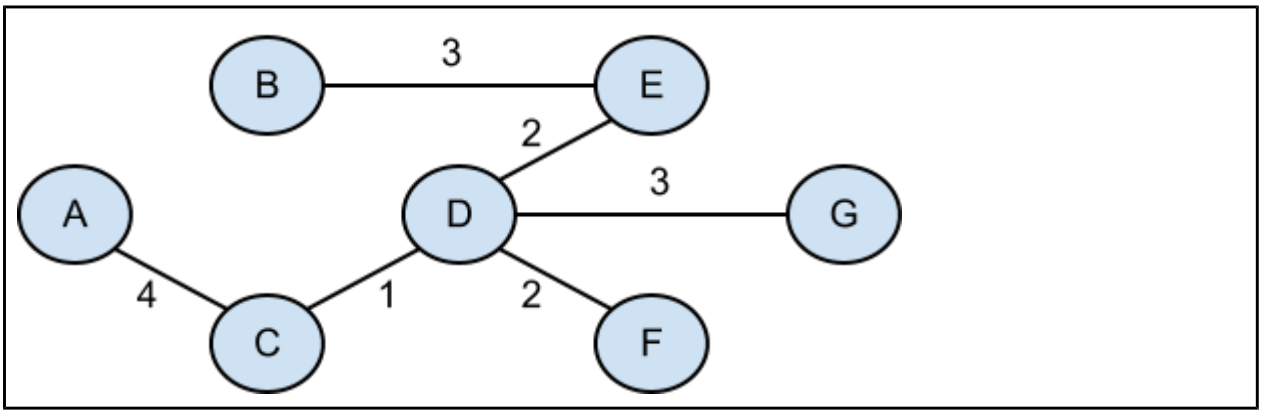
Cognome	Nome	Matricola

1	2	3	4	5

Esercizio 1

Dato il grafo in figura, trovare il minimo albero di copertura.





Esercizio 2

Dimostrare che, se f(n) è O(g(n)) e g(n) è O(h(n)), allora f(n)g(n) è O(k(n)) con k(n)=h(n)h(n).

Per ipotesi
Esistono n0, c0 tali che per ogni n>=n0 f(n)<=c0g(n)
Esistono n1, c1 tali che per ogni n>=n1 g(n)<=c1h(n)

Per n2=max(n0, n1) vale che f(n)*g(n) <= c0g(n) * c1h(n) <= c0c1h(n)*c1h(n)

Quindi

Per n2=max(n0, n1) e c= c0c1c1 vale che : per ogni n>=n2 f(n)g(n)<=k(n)

Esercizio 3

Calcolare la complessità del **for** in funzione di **n**>0.
for (int i=0; i <= f(n); i++) cout << f(g(n*n));
con le funzioni **f** e **g** 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.

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

```
int f(int x) {
  if (x<=1) return 1;
  int a=g(x);
  cout << a;
  return x + f(x-1);
}
```

Funzione g

$T_g(0)$= d
$T_g(n)$= cn + 4$T_g(n/2)$ $T_g(n)$ è O(n^2)

$R_g(n)$= 1 $R_g(n)$ è O(logn)
$R_g(n)$= 1 + $R_g(n/2)$

Funzione f

$T_f(0)$= d
$T_f(n)$= cn^2 + $T_f(n-1)$ $T_f(n)$ è O(n^3)

$R_f(0)$= d
$R_f(n)$= cn + $R_f(n-1)$ $R_f(n)$ è O(n^2)

Calcolo for del blocco:

numero iterazioni: O(n^2)
Complessità della singola iterazione: $T_f(n)$ + $T_g(n^2)$ + $T_f(logn)$ = O(n3) + O(n^4) + O((logn)^3) = O(n^4)
Complessità del for: =O(n^6)

Esercizio 4

Scrivere una funzione che, dato un albero binario **t** ad etichette intere, sommi 1 all'etichetta di ogni nodo che ha fra i suoi discendenti almeno una etichetta uguale a 10 e un'etichetta uguale a 20.

```
void somma (Node* t, int& dieci, int& venti) {
  if(!t) { dieci=venti=0; return; }
  int dieci_l,dieci_r,venti_r,venti_r;
  somma(t->left, dieci_l,venti_l);
  somma(t->right, dieci_r,venti_r);
  dieci=dieci_l+dieci_r;
  venti=venti_l+venti_r;
  int l = t->label;
  t->label+=(dieci>0 && venti>0);
  dieci=dieci + (l==10);
  venti=venti + (l==20);
}
```

Esercizio 5

Scrivere una funzione che, dato un albero generico **t** ad etichette intere e un'etichetta **x**, inserisca l'ultimo figlio di **x** con tutto il suo sottoalbero come fratello immediatamente successivo ad **x**.

```
void es5(Node* t, int x){
  Node* n = find_node(t,x);
  if(n == NULL || n->left == NULL) return;
  Node c = n->left;
  if(c->right == NULL){
    // un figlio solo:
    n->left=NULL;
    c->right=n->right;
    n->right=c;
  }
  else{
    // più figli:
    for(;c->right->right != NULL; c=c->right);
    // c -> penultimo figlio
    c->right->right=n->right;
    n->right=c->right;
    c->right=NULL;
  }
}
```