ANNO ACCADEMICO 2015/16

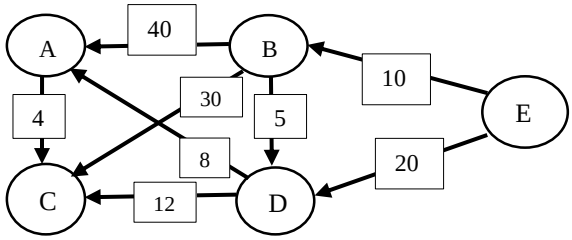
Algoritmi e Basi di dati – Modulo di Algoritmi e Strutture dati

23 febbraio 2017

1	2	3	4	5
7	6	7	7	6

Esercizio 1

a) Descrivere l'algoritmo di Dijkstra.
b) Applicarlo al grafo seguente con nodo di partenza E e indicare i cammini minimi



Q	**A**	**B**	**C**	**D**	**E**
A, B, C, D, E	**Inf | -**	**Inf | -**	**Inf | -**	**Inf | -**	**0 | -**
A, B, C, D	**Inf | -**	**10 | E**	**Inf | -**	**20 | E**	**0 | -**
A, C, D	**50 | B**	**10 | E**	**40 | B**	**15 | B**	**0 | -**
A, C	**23 | D**	**10 | E**	**27 | D**	**15 | B**	**0 | -**
C	**23 | D**	**10 | E**	**27 | D**	**15 | B**	**0 | -**
Cammini minimi	**E B D A**	**E B**	**E B D C**	**E B D**	

Esercizio 3
Calcolare la complessità dell'espressione g(f(n)) + f(g(n)) in funzione di n (indicando le relazioni di ricorrenza di tempo e risultato per ogni funzione) con le funzioni f e g definite come segue:

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

```
int g(int x) {
int b=0;
  if (x<=1) return 5;
  for (int i=1, i<=x*x;i++)
     b+=i;
  return b + g(x-1);
}
```

$T_f(1)= k_1$
$T_f(n)= k_2\, n^3 + 2T_f(n/2)$ $T_f(n)$ è $O(n^3)$

$R_f(1)= k$
$R_f(n)= kn^5 + 4R_f(n/2)$ $T_f(n)$ è $O(n^5)$

Calcolo $T_g(n)$
for:
numero iterazioni: = O(n^2)
Complessità singola iterazione: $T_f(n) = O(1)$
complessità del for: $O(n^2)$

$T_g(1)= k_1$
$T_g(n)= k_2\, n^2 + T_g(n-1)$ $T_g(n)$ è $O(n^3)$

$R_g(1)= 5$
$R_g(n)= k\, n^4 + R_g(n-1)$ $T_g(n)$ è $O(n^5)$

Tempo di g(f(n))= Tempo per il calcolo di f(n) + tempo per il calcolo di g(n^5) =

O(n^3)+O(n^15)= O(n^15)

Tempo di f(g(n))= Tempo per il calcolo di g(n) + tempo per il calcolo di f(n^5) =

O(n^3)+O(n^15)= O(n^15)

Tempo per l'esecuzione di g(f(n)) * f(g(n)) = O(n^15) + O(n^15) = O(n^15)

Esercizio 4

Dato un albero generico memorizzato figlio-fratello con etichette intere, scrivere una funzione che, dati tre interi p, q e r, aggiunge al nodo con etichetta r un primo figlio con etichetta p e un ultimo figlio con etichetta q.

```
void aggiungi (Node* t, int p, int q, int r) {
   Node* a=findNode (r,t);
   if (! a)  return;
   Node * b=a->left;
   a->left= new Node;
   a->left->right=b;
   a->left->label=p;
   for (Node* c=a->left; c->right; c=c->right);
   c->right= new Node;
   c->right->label=q;
   c->right->left=0;
   c->right->right=0;
}
```

Esercizio 5

Spiegare il meccanismo delle funzioni e classi modello.
Indicare l'uscita del seguente programma.

```
class uno {
     int a;
public:
     uno(){ a=1; cout << a << endl;}
     void valore(){ a++; cout << a << endl;}
};
class due {
     int a;
public:
     due(){ a=2; cout << a << endl;}
     void valore(){ a++; cout << a << endl;}
};
template<class tipo1, class tipo2>
class tre {
     static int b;
     tipo1 x;
     tipo2 y;
public:
     tre (){ cout << 3 << endl << b << endl; };
     void valore(){
                    x.valore();
                    y.valore();

}
```

```
};
template<class tipo1, class tipo2>
int tre<tipo1, tipo2>::b;

int main (){
tre<uno,uno> obj1;
tre<uno, due> obj2;
obj1.valore();
obj2.valore();
}
```

```
1
1
3
0
1
2
3
0
2
2
2
3
```

Esercizio 2

Descrivere gli algoritmi di ordinamento fatti a lezione e confrontarli fra loro.