ANNO ACCADEMICO 2010/2011- 24 giugno 2011

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

Esercizio 1

Scrivere una funzione ricorsiva che stampa i nodi di uno heap in ordine simmetrico.

```
void visita (const int* a, int i=0, int last) {
    if (i> last) return;
    visita(a, 2*i+1, last);
    cout << a[i];
    visita(a, 2*i+2, last);
 }
```

Esercizio 2

Applicare l'algoritmo di Dijkstra al grafo in figura, per trovare i cammini minimi dal nodo A a qualsiasi altro nodo del grafo.

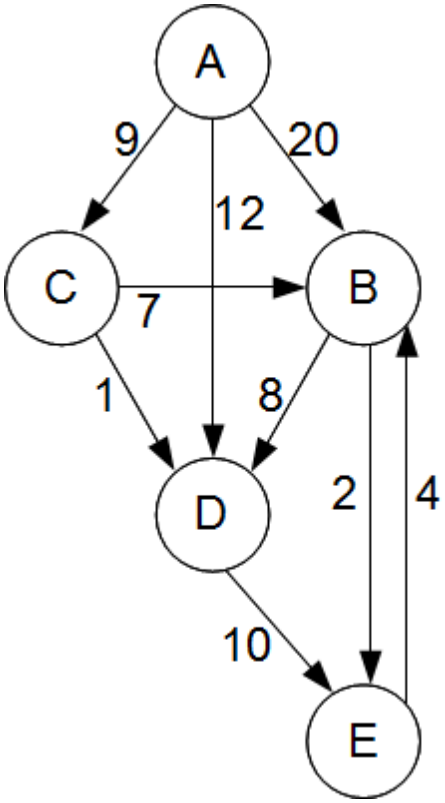

Tabella dist | pred

Q	A	B	C	D	E
A,B,C,D,E	0 | inf	inf | -	inf | -	inf | -	inf | -
B,C,D,E	0 | inf	20 | A	9 | A	12 | A	inf | -
B,D,E	0 | inf	16 | C	9 | A	10 | C	inf | -
B,E	0 | inf	16 | C	9 | A	10 | C	20 | D
E	0 | inf	16 | C	9 | A	10 | C	18 | B

Esercizio 3

Calcolare la complessità del for in funzione di n.
`for (int i=0; i <= f(g(n)); i++) cout << i;`
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++;
  return a*a+ g (x-1);
       }
```

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

Funzione g

Numero di iterazioni del for: n
Complessità della singola iterazione: O(1)
Complessità del for: =O(n)

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

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

Funzione f

Numero di iterazioni del for: n^3
Complessità della singola iterazione: O(n^4)
Complessità del for: =O(n^7)

$T_f(0)= d$
$T_f(n)= cn^7+ 2T_f(n/2)$ $T_f(n)$ è $O(n^7)$

$R_f(0)= d$
$R_f(n)= c+ 4R_f(n-1)$ $R_f(n)$ è $O(n^2)$

Calcolo for del blocco:
numero iterazioni: $f(g(n))= f(n^3) = O(n^6)$
Complessità della singola iterazione: $T_g(n) + T_f(n^3) = O(n^2) + O(n^{21}) = O(n^{21})$
Complessità del for: $O(n^{27})$

Esercizio 4

Scrivere una funzione ricorsiva che, dato un albero binario a etichette intere, somma ad ogni nodo il numero di nodi interni (cioé non foglie) appartenenti al suo sottoalbero destro.

```
int somma_interni (Node* t) {
    if (!t) return 0;
    if (!t->left && !t->right) return 0;
    int l = somma_interni(t->left);
    int r = somma_interni(t->right);
    t->info += r;
    return l + r + 1;
    }
```

Esercizio 5

Scrivere una funzione ricorsiva **int piu_figli(const Node* t)** che, dato un albero generico memorizzato con la memorizzazione figlio-fratello, restituisce il massimo numero di figli dei nodi dell'albero.

```
int piu_figli (const Node * t) {
    if (!t) return 0;
    int a=conta(t->left);
    int l= piu_figli(t->left);
    int r= piu_figli(t->right);
    return max(a,l,r);
    }


int piu_figli (const Node * t) {
    if (!t) return 0;
    return 1 + conta(t->right);
    }
```

SOLUZIONE ALTERNATIVA:

```
int piu_figli (const Node* t, int& numFrat) {
    if (!t) { return 0; numFrat = 0; }
    int numFigli;
    int l = piu_figli(t->left, numFigli); numFigli++;
    int r = piu_figli(t->right, numFrat); numFrat++;
    return max(numFigli,l,r);
    }
```