FONDAMENTI DI INFORMATICA II – Algoritmi e Strutture dati

19 settembre 2016 - ANNO ACCADEMICO 2015/16

1	2	3	4	5
5	7	7	7	7

Esercizio 1

Sia dato il seguente min-heap (un min-heap è uno heap in cui ogni nodo è minore o uguale dei suoi figli e le operazioni di up e down cambiano di conseguenza):

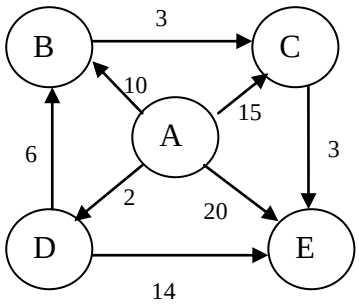
[10 20 15 30 20 40 17]

mostrare lo stato dello stesso e le chiamate ad up e down:
A) dopo l'inserzione dell'intero 9
B) dopo l'estrazione di un elemento dal min-heap ottenuto al passo A
C) dopo l'estrazione di un elemento dal min-heap ottenuto al passo B

A 9 10 15 20 20 40 17 30	**up(7), up(3), up(1), up(0)**
B 10 20 15 30 20 40 17	**down(o), down(1), down(3)**
C 15 20 17 30 20 40	**down(o), down(2)**

Esercizio 2

a) Descrivere l'algoritmo di Dijkstra: a cosa serve, il suo funzionamento, la sua complessità. (3)
b) Applicarlo al grafo in figura con il nodo A come nodo di partenza. (4)



Q	A	B	C	D	E
A, B, C, D, E	0 | -	inf | -	inf | -	inf | -	inf | -
B, C, D E	0 | -	10 | A	15 | A	2 | A	20 | A
B, C, E	0 | -	8 | D	15 | A	2 | A	16 | D
C, E	0 | -	8 | D	11 | B	2 | A	16 | D
E	0 | -	8 | D	11 | B	2 | A	14 | C

Esercizio 3

Calcolare la complessità in funzione di **n**>0 dell'istruzione

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

con le funzioni **f** e **g** definite come segue:

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

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

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

Stima del tempo di f

Primo for
numero iterazioni = O(n)
complessità di un'iterazione = costante
tempo del for = O(n)

Secondo for
numero iterazioni for = $O(n^4)$
complessità di un'iterazione = costante
tempo del for = $O(n^4)$

$T_f(1) = a$
$T_f(n) = b\,n^4 + T_f(n-1)$ $\quad O(n^5)$

$R_f(1)=a$
$R_f(n) = n^8 + R_f(n-1)$ $\quad O(n^9)$

Stima del tempo di g:
numero iterazioni del for: $R_f(m)= O(m^9)$
complessità di un'iterazione: $T_f(m) = O(m^5)$
tempo del for: $O(m^{14})$

tempo di g
$T_g(1)=$ cost
$T_g(m)= c\cdot m^{14} + T_g(m/2)$
T_g è $O(m^{14})$

Tempo di y=g(f(n)) :

Tf(n) + Tg(n^9) = O(n^5)+O(n^(9*14)) = O(n^126)

Esercizio 4
Sia dato un albero binario ad etichette intere. Scrivere una funzione che, per ogni nodo somma all'etichetta la differenza fra il numero di discendenti di sinistra e il numero di foglie di destra. La complessità della funzione deve essere O(n), con n numero di nodi dell'albero.

```
int somma(Node* t, int & foglie) {
  if (!t)
    {foglie=0; return 0; }
  if ( !t-left && !t->right)
     {foglie=1; return 1; }

  int nodi_l, nodi_r, foglie_l, foglie_r ;
  nodi_l = somma(t->left, foglie_l);
  nodi_r = somma(t->right, foglie_d);
  t->label+=nodi_l-foglie_r ;
  foglie= foglie_l+foglie_r ;
  retutn nodi_l+nodi_r+1 ;
}
```

Esercizio 5 Sia dato il seguente programma c++.

```
class A {
protected:
       int a;
public:
       A(){a=8;}
       void stampa (){ cout <<  a; }
};

class B: public A {
protected:
       int a;
public:
       B(){a=9;}
       void stampa (){ cout <<  a; }
};

class C: public A {
protected:
       int a;
public:
       C(){a=10;}
*      void stampa (){ cout <<  a; }
};

class D: public C {
public:
       D(){a=11;}

};

int  main(){
       B *obj1= new B;
       D *obj2= new D;
       C *obj3= new C;
       obj1->stampa();
       obj2->stampa();
        obj3->stampa();
}
```

1. Indicare l’uscita del programma (3)

a) così come è scritto
b) eliminando la linea asteriscata

a) 9 11 10
b) 9 8 8

2. Spiegare la eventuale differenza fra i due casi. (2)

Nel secondo caso la funzione stampa che viene chiamate da obj2 e da obj3 è quella della classe A, poiché né la classe C né la classe D ridefiniscono questa funzione e quindi viene chiamata la funzione ereditata da A.

3. Spiegare cosa vuol dire “classe astratta”. (2)

E’ una classe che contiene almeno una funzione virtuale pura e per questo non può essere instanziata.