Informatica Generale II - Prova teorica

A.A. 20052006

Esame: 27 aprile 2007
Codice: XRUQ

1. Supponendo di aver definito una funzione `funzEsame` con la seguente intestazione:

   ```
   void funzEsame (double *i, double *j);
   ```

   L'invocazione della suddetta funzione `funzEsame` avrà la forma:

   (a)
   ```
   double i=1.0, j=0.5;
   funzEsame(&i, &j);
   ```
   (b)
   ```
   double i=1.0, j=0.5;
   funzEsame(*i, *j);
   ```
   (c)
   ```
   double i=1.0, j=0.5;
   funzEsame();
   ```
   (d)
   ```
   double i=1.0, j=0.5;
   funzEsame(i, j);
   ```
   (e) non rispondo

2. Quali degli alberi in figura 1 hanno una lughezza del cammino maggiore di 15?

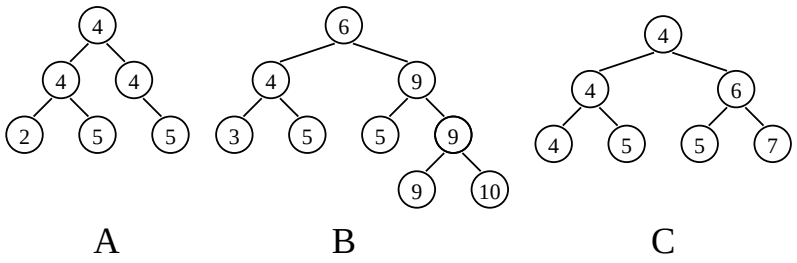

Figura 1:

   (a) C
   (b) A
   (c) B,C
   (d) B
   (e) non rispondo

3. Il costruttore è una funzione speciale che:

   (a) viene definita all'interno di una struttura e ha lo scopo di allocare dinamicamente memoria nello heap per i campi della struttura;
   (b) viene definita all'interno di una struttura, può essere identificata tramite un qualunque identificatore valido e deve ritornare un tipo identico al tipo della struttura;
   (c) viene definita all'interno di una struttura, ha come identificatore lo stesso della struttura, non ha tipo di ritorno e deve inizializzare tutti i campi della struttura;
   (d) viene definita all'interno di una struttura, ha come identificatore lo stesso nome della struttura, non ha tipo di ritorno e serve per inizializzare i campi della struttura;

(e) non rispondo

4. Si consideri un grafo: sia n il numero di nodi e m il numero di archi. Quale tra le seguenti affermazioni è *falsa*?

   (a) l'algoritmo DFS richiede tempo proporzionale a $(n + m)$;
   (b) l'algoritmo BFS si ottiene dall'algoritmo DFS iterativo sostituendo la Coda con una Stack;
   (c) l'algoritmo BFS si ottiene dall'algoritmo DFS iterativo sostituendo lo Stack con una Coda;
   (d) l'algoritmo BFS richiede tempo proporzionale a $(n + m)$
   (e) non rispondo

5. Si consideri il limite superiore asintottico $\mathcal{O}(f(n))$ alla complessità di un problema. Quale tra le seguenti affermazioni è *falsa*:

   (a) esistono tre opportune costanti a, b, c tali che il numero di istruzioni $t(n)$ che vengono eseguite nel caso peggiore con input di dimensione n verifica per ogni $n > c$ la seguente relazione: $t(n) < a * f(n) + b$;
   (b) La notazione $\mathcal{O}(f(n))$ fornisce una valutazione approssimata per eccesso della complessità di un algoritmo;
   (c) un algoritmo ha complessità $\mathcal{O}(f(n))$ se, per ogni *n* e per qualche input di dimensione *n*, l'algoritmo impiega una quantità di risposte proporzionali a $f(n)$;
   (d) esiste almeno un algoritmo di soluzione avente complessità $\mathcal{O}(f(n))$;
   (e) non rispondo

6. Dato il grafo in figura 2:

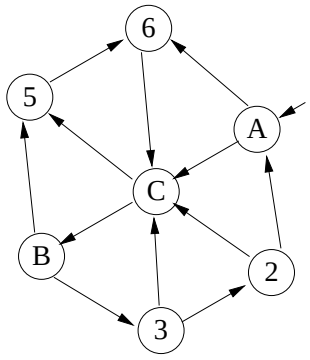

Figura 2:

   (a) La visita in ampiezza produce la seguente sequenza d'uscita: A 6 C 5 3 B 2
   (b) La visita in ampiezza produce la seguente sequenza d'uscita: A C B 3 2 5 6
   (c) Nessuna delle risposte è accettabile;
   (d) La visita in ampiezza produce la seguente sequenza d'uscita: A 6 C B 5 3 2
   (e) non rispondo

7. L'algoritmo selection sort ha complessità asintotica:

   (a) $\mathcal{O}(N^{2/3})$
   (b) $\mathcal{O}(N^2)$
   (c) nessuna delle altre risposte proposte è corretta
   (d) $\mathcal{O}(NlogN)$
   (e) non rispondo

8. Il seguente codice:

```
int a[5];
++(*((&(*(a+3))) - 1));
```

(a) ha l’effetto di incrementare di 1 il secondo elemento dell’array `a`
(b) è sbagliato perché `a` è un array e non un puntatore;
(c) ha l’effetto di incrementare di 1 il quarto elemento dell’array `a`
(d) ha l’effetto di incrementare di 1 il terzo elemento dell’array `a`
(e) non rispondo

9. Quale tipo di attraversamento d’albero implementa il seguente codice?

```
void foo-order(Node* l, void visit(Node*))
  StackPtr s = new Stack();

  Push(s, i);
  while(! StackIsEmpty(s)) {
    Node * h = Pop(s);

    if ((h->left == NULL) && (h->right == NULL)) visit(h);
    else Push(s, new Node(h->data, true));
    if (h->right != NULL) Push(s, h->right);
    if (h->left != NULL) Push(s, h->left);
    if (h->flag == true) delete h;
  }

  delete s;
}
```

(a) preorder
(b) inorder
(c) postorder
(d) level-order
(e) non rispondo

10. Una pila è:

(a) un multiinsieme di elementi in cui ogni eliminazione ha per oggetto l’elemento inserito per primo;
(b) un multiinsieme di elementi gestiti secondo la politica *lifo* (last in first out);
(c) un insieme di elementi gestiti secondo la politica *lifo* (last in first out);
(d) un multiinsieme di elementi gestiti secondo la politica *fifo* (first in first out);
(e) non rispondo