Informatica Generale II - Prova teorica

A.A. 20052006

Esame: 27 aprile 2007
Codice: PQTP

1. Si consideri la rappresentazione di un grafo mediante *lista delle adiacenze* (sia n il numero di nodi e m il numero di archi). Quale tra le seguenti affermazioni è *falsa*?

   (a) l'operazione di accesso a tutti i successori di un nodo può essere effettuata in modo più efficiente che non se fosse stata usata la rappresentazione mediante matrice delle adiacenze;
   (b) la verifica dell'esistenza di un arco da un nodo i a un nodo j si può effettuare in modo meno efficiente che non se fosse stata usata la rappresentazione mediante matrice delle adiacenze;
   (c) tale rappresentazione richiede un'occupazione di memoria proporzionale al numero di nodi del grafo;
   (d) tale rappresentazione richiede un'occupazione di memoria proporzionale a $(n + m)$;
   (e) non rispondo

2. Quale tipo di attraversamento d'albero implementa il seguente codice?

```
void foo-order(Node* l, void visit(Node*))
{
  Coda q;
  put(q, l);
  while (coda_piena(q)) {
    l = get(q);
    visit(l);
    if (l->left != NULL) put(q, l->left);
    if (l->right != NULL) put(q, l->right);
  }
}
```

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

3. In quale ordine partendo dal nodo 1 vengono visitati i nodi del grafo in figura 1 da un algoritmo di visita in profondità?

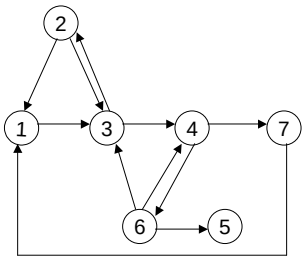

Figura 1:

   (a) 1346572

(b) Nessuna risposta è accettabile;
(c) 1342756
(d) 1342675
(e) non rispondo

4. Il costruttore è una funzione speciale che:

   (a) 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;
   (b) 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;
   (c) 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;
   (d) viene definita all'interno di una struttura e ha lo scopo di allocare dinamicamente memoria nello heap per i campi della struttura;
   (e) non rispondo

5. La seguente funzione `foo`:

```
void foo(Node * x, Node * y) {
  y->next = x->next;
  x->next = y;
}
```

   (a) inserisce la lista puntata da y dopo il nodo x
   (b) inserisce la lista puntata da x dopo il nodo y
   (c) concatena due liste concatenate x e y
   (d) inserisce il nodo y tra il nodo x e il successore di x
   (e) non rispondo

6. Si consideri la seguente funzione:

```
int* funz(int dim) {
  int* punt;
  punt=new int[dim];
  for (int i=0; i < dim; ++i) punt[i]=i;
  return punt;
}
```

   La memoria allocata all'interno della funzione `funz` mediante l'operatore `new`:

   (a) rimane allocata fin quando il programma non termina o finché non venga esplicitamente deallocata tramite una istruzione `delete []`
   (b) rimane allocata fin quando il programma non termina o finché non viene esplicitamente deallocata tramite l'istruzione `delete`
   (c) non può più venire deallocata poiché l'operatore `delete` non compare all'interno della funzione stessa
   (d) viene deallocata quando il controllo passa all'esterno della funzione `funz`
   (e) non rispondo

7. Si supponga di avere la sequenza di numeri *1,6,4,2,3*. Dopo *3* iterazioni soltanto di Bubblesort su tale sequenza, il risultato sarà:

   (a) 1,6,2,3,4
   (b) 1,2,3,6,4
   (c) 1,2,6,3,4
   (d) 1,2,6,4,3
   (e) non rispondo

8. Dato l'albero n-ario in figura 2:

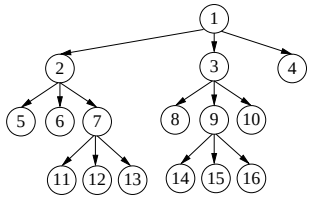

Figura 2:

(a) La visita in preordine dell'albero produce in uscita la seguente sequenza: 1 2 5 6 7 11 12 13 3 8 9 10 14 15 16 4
(b) Non si può effettuare la visita in postordine di un albero non binario;
(c) La visita in preordine dell'albero produce in uscita la seguente sequenza: 1 2 5 6 7 11 12 13 3 8 9 14 15 16 10 4
(d) La visita in preordine dell'albero produce in uscita la seguente sequenza: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
(e) non rispondo

9. Si consideri l'algoritmo di fusione (merge) tra due array ordinati aventi n elementi ciascuno:

(a) ha complessità lineare poiché ogni volta che si inserisce un elemento nell'array risultato si esegue un numero costante di operazioni;
(b) ha complessità $\mathcal{O}(n^2)$;
(c) si avvale di questo algoritmo l'ordinamento per fusione che funziona nel seguente modo: si sceglie a caso un elemento di pivot sulla base del cui valore si divide in due parti l'array, di seguito ciascuna parte viene ricorsivamente ordinata e infine i due array vengono fusi;
(d) si implementa sempre tramite programmazione ricorsiva;
(e) non rispondo

10. Quali degli alberi in figura 3 *sono* alberi binari di ricerca, mentre gli altri non lo sono?

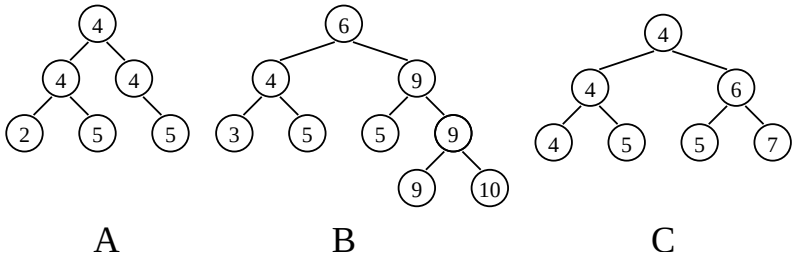

Figura 3:

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