Informatica Generale II - Prova teorica

A.A. 20052006

Esame: 27 aprile 2007
Codice: VGHP

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

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

2. Qual'è l'altezza dell'albero rappresentato in figura 1?

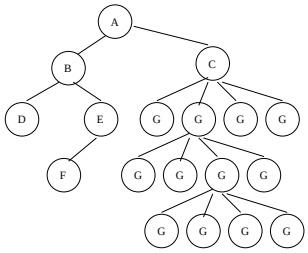

Figura 1: Albero

   (a) nessuno degli altri valori
   (b) 5
   (c) 4
   (d) 3
   (e) non rispondo

3. Quale è la lunghezza del cammino dell'albero in figura 2

   (a) 3

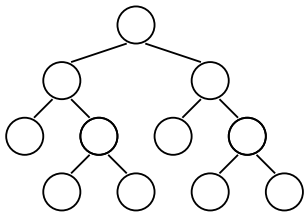

Figura 2:

(b) 23
(c) 11
(d) 22
(e) non rispondo

4. Le operazioni di base di stack e code sono:

   (a) push e pop
   (b) push e pop per stack, put e get per code
   (c) put e get
   (d) push e pop per code, put e get per stack
   (e) non rispondo

5. Si consideri la rappresentazione di un grafo mediante *matrice delle adiacenze* (sia n il numero di nodi). Quale tra le seguenti affermazioni è *falsa*?

   (a) tale rappresentazione richiede un'occupazione di memoria proporzionale al numero massimo di archi del grafo;
   (b) tale rappresentazione richiede un'occupazione di memoria proporzionale al numero di nodi del grafo;
   (c) l'operazione di accesso ai successori di un nodo richiede l'accesso ad n elementi della matrice;
   (d) l'elemento nella riga i e nella colonna j della matrice è pari a 1 se nel grafo rappresentato c'è un arco dal nodo i al nodo j, è pari a 0 nel caso contrario;
   (e) non rispondo

6. Si consideri il seguente frammento di codice:

```
struct Tipo1 {
 Tipo2 a;
};

struct Tipo2 {
 Tipo1 a;
};
```

   (a) è errato perché Tipo1 e Tipo2 contengono due campi con lo stesso nome.
   (b) è errato perché Tipo2 non ha un costruttore di default.
   (c) è errato perché Tipo1 e Tipo2 non sono istanziabili.
   (d) è corretto, ma Tipo2 non è istanziabile.
   (e) non rispondo

7. Si analizzi il seguente codice:

```
void foo(int A[], int N) {
  for (int i=0; i < N-1; ++i) {
    int min = i;
    for (int j=i+1; j < N; ++j)
      if (A[j] < A[min]) min = j;
    swap(A[i], A[min]);
  }
}
```

Esso implementa l'algoritmo:

(a) Mergesort
(b) Quicksort
(c) Bubblesort
(d) Selection sort
(e) non rispondo

8. Si consideri l'algoritmo di ordinamento per fusione (mergesort):

   (a) si avvale dell'algoritmo di fusione tra due array ordinati (merge) che ha complessità lineare. Tale algoritmo di fusione viene attivato un numero di volte pari a $\mathcal{O}(log\, n)$ e da ciò si ha che la complessità dell'algoritmo mergesort è $\mathcal{O}(n * log\, n)$;
   (b) si avvale dell'algoritmo di fusione tra due array ordinati (merge) che ha complessità $\mathcal{O}(1)$. Tale algoritmo di fusione viene attivato un numero di volte pari a $\mathcal{O}(log\ n)$ e da ciò si ha che la complessità dell'algoritmo mergesort è $\mathcal{O}(log\, n)$;
   (c) si avvale dell'algoritmo di fusione tra due array ordinati (merge) che ha complessità $\mathcal{O}(n^2)$. Tale algoritmo di fusione viene attivato un numero di volte pari a $\mathcal{O}(log\, n)$ e da ciò si ha che la complessità dell'algoritmo mergesort è $\mathcal{O}(n^2 * log\, n)$;
   (d) si avvale dell'algoritmo di fusione tra due array ordinati (merge) che ha complessità $\mathcal{O}(log\ n)$. Tale algoritmo di fusione viene attivato un numero di volte pari a $\mathcal{O}(n)$ e da ciò si ha che la complessità dell'algoritmo mergesort è $\mathcal{O}(n * log\, n)$;
   (e) non rispondo

9. Supponendo di aver definito una funzione `funzEsame` con la seguente definizione:

```
void funzEsame (int i, int & j) {
  j += i;
}
```

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

   (a)
```
int i=10, j=5;
funzEsame(5, j);
```
   (b)
```
int i=10, j=5;
funzEsame(i+1, j);
```
   (c)
```
int i=10, j=5;
funzEsame(i, j);
```
   (d)
```
int i=10, j=5;
funzEsame(i, j+1);
```
   (e) non rispondo

10. Dato il grafo in figura 3:

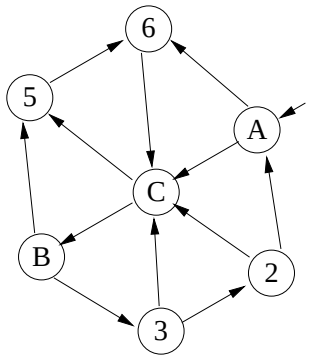

Figura 3:

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