Informatica Generale II - Prova teorica

A.A. 2005/2006

Esame: 23 giugno 2006
Codice: YWST

1. Quale tra le seguenti affermazioni relative alla ricerca binaria è *falsa*?

   (a) essendo n il numero di elementi nell'array, nel caso peggiore il numero di confronti da effettuare è il più piccolo m che soddisfa la relazione seguente: $2^m >= n$
   (b) la complessità della ricerca binaria non dipende dal particolare elemento da individuare;
   (c) la complessità della ricerca binaria è $\mathcal{O}(log\ n)$
   (d) essendo n il numero di elementi nell'array, dopo j confronti la dimensione dell'array su cui si deve continuare la ricerca è $n/(2^j)$;
   (e) non rispondo

2. 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) l'operazione di accesso ai successori di un nodo richiede l'accesso ad n elementi della matrice;
   (b) 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;
   (c) tale rappresentazione richiede un'occupazione di memoria proporzionale al numero di nodi del grafo;
   (d) tale rappresentazione richiede un'occupazione di memoria proporzionale al numero massimo di archi del grafo;
   (e) non rispondo

3. Un array allocato tramite una `new`

   (a) può essere localizzato nell'area di memoria di stack, oppure nell'area di heap;
   (b) è sempre localizzato in un area di memoria detta stack;
   (c) è sempre localizzato nell'area di memoria di heap;
   (d) non va mai deallocato esplicitamente;
   (e) non rispondo

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

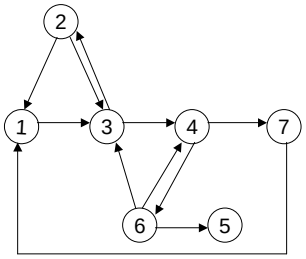

Figura 1:

   (a) 1342756
   (b) Nessuna risposta è accettabile;

(c) 1346572
(d) 1342675
(e) non rispondo

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

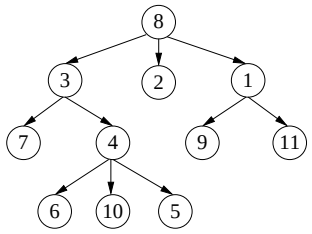

Figura 2:

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

6. Si suppoga di voler implementare un ADT stack di dimensione fissata. Quale di queste affermazioni è *falsa*?

(a) se implementato con array piuttosto che con lista concatenata, lo stack occupa meno spazio in memoria
(b) le operazioni di inserimento ed estrazione hanno uguale complessità asintotica: costante
(c) la complessità asintotica delle operazioni di inserimento ed estrazione di un elemento è maggiore se lo stack è implementato con lista concatenata invece che con array
(d) se implementato con array piuttosto che con lista concatenata, l'array occupa piu' spazio in memoria
(e) non rispondo

7. Data la seguente dichiarazione:

```
double vett[5];
```

Le istruzioni `sizeof(vett)` e `sizeof(double)`

(a) la dimensione del puntatore all'array e la dimensione in byte di un puntatore a una variabile di tipo `double`
(b) restituiscono rispettivamente la dimensione dell'intero array e il numero di byte necessari a rappresentare una variabile di tipo `double`
(c) sono rispettivamente corretta ed errata perché l'operatore `sizeof()` non può ricevere come parametro il nome di un tipo
(d) restituiscono rispettivamente la dimensione del puntatore all'array e la dimensione di un singolo elemento dell'array
(e) non rispondo

8. Dato l'albero n-ario in figura 3:

(a) Nessuna risposta è accettabile.
(b) la sua altezza è: 3
(c) la sua altezza è: 4
(d) la sua altezza è: 2
(e) non rispondo

9. Si analizzi il seguente codice:

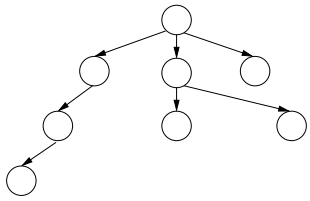

Figura 3:

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

10. Si consideri il seguente frammento di codice:

```
struct Tipo2;

struct Tipo1 {
 Tipo2* a;
};

struct Tipo2 {
 Tipo1 a;
 int b;

 Tipo2(int _b) {
   b = _b;
 }
};
```

(a) è corretto e compilabile.
(b) è errato perché Tipo2 non ha un costruttore di default.
(c) è corretto, ma Tipo2 non è istanziabile.
(d) è errato a causa di una mutua dipendenza tra Tipo1 e Tipo2.
(e) non rispondo