UNIVERSITÀ DEGLI STUDI DI BERGAMO

DIPARTIMENTO DI INGEGNERIA
DELL'INFORMAZIONE E METODI MATEMATICI

Corso di Laurea in Ingegneria Informatica

Esame di **Informatica III B – Progettazione e algoritmi a.a. 2012/13**

I Appello Gennaio 2013

1. Quale delle seguenti affermazioni è vera? **[0,5 pt]**

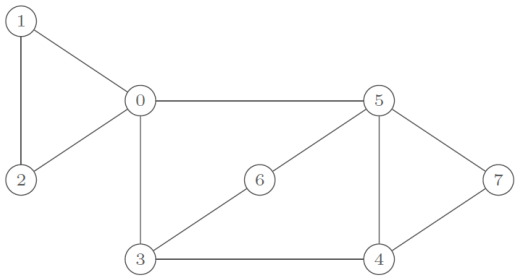
a. $n! = O(n!)$
b. $\log n = \Omega(n)$
c. $\log n = O(\log \log n)$
d. Nessuna delle precedenti è vera

2. Nell'algoritmo di ordinamento *Insertion sort*, nel caso migliore quant'è il numero di spostamenti effettuato? Motivare la risposta. **[1 pt]**

3. Data la seguente funzione ricorsiva foo, trovate la corrispondente relazione di ricorrenza e risolvetela utilizzando il *teorema Principale* (*Master*). **[1,5 pt]**

```
foo( n ){
        if (n < 10) return 1;
        else if (n < 1234){
                tmp = n;
                for (i = 1; i <= n; i = i+1)
                for (j = 1; j <= n; j = j+1)
                for (k = 1; k <= n; k = k+1)
                tmp = tmp + i * j * k;
        }
        else {
                tmp = n;
                for (i = 1; i <= n; i = i+1)
                  for (j = 1; j <= n; j = j+1)
                     tmp = tmp + i * j;
                }
         return tmp + foo( n/2 ) * foo( n/2 ) + foo( n/2 );
}
```

4. Dato il grafo non orientato in figura, **[1,5pt]**



a. determinare le componenti connesse;
b. calcolare l'albero BFS prodotto visitando in ampiezza (visita BFS) il grafo a partire dal nodo sorgente **s=0** e seguendo un ordine numerico;
c. classificare, inoltre, tutti gli archi in base all'albero BFS prodotto dalla visita.

5. Definire un algoritmo ricorsivo per invertire gli elementi di un array. Impostare poi l'equazione di ricorrenza per il tempo di esecuzione e risolverla. **[1,5 pt]**

6. Dato un array ordinato contenente n elementi di tipo intero. Progettare un algoritmo ricorsivo che data una chiave k restituisce il numero di occorrenze di k nell'array. L'algoritmo deve avere complessità tempo O(log n). **[4pt]**

Suggerimento: Una possibile soluzione è una variante dell'algoritmo di ricerca binaria.

SOLUZIONE:

1. **Risposta**: a.
2. **Risposta**:

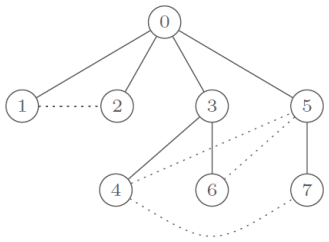
   Il caso migliore si ha quando l'array è già ordinato.

```
InsertionSort (A)
1.  for j=2 to n do
2.      x = A[j] //elemento da inserire nella sotto-sequenza ordinata A[1..j-1]
3.      i= j-1 //indice di scansione da destra verso sinistra (da j-1 ad 1)
4.      while i>0 and A[i] > x do
5.          A[i+1]= A[i]  //sposta di una posizione tutti gli elementi A[i] > x
6.          i= i-1
7.      A[i+1]=x //Inserisce x nella locazione che gli compete
```

   Il numero di spostamenti nel caso migliore è lineare: $\mathbf{S_{best}(n)=\Theta(n)}$. Infatti, il numero di spostamenti per ogni j-esima iterazione (j=2..n) sono solo 2 (per caricare/scaricare x). In totale:

$$S_{best}(n)=\sum_{j=2..n} 2= \Theta(n)$$

3. **Risposta**: La relazione di ricorrenza ha come caso base di costo costante il ramo then. Il secondo ramo else e le chiamate ricorsive della seconda istruzione return contribuiscono al costo "asintotico": $\mathbf{T(n) = 3\ T(n/2) + \Theta(n^2)}$. Quindi rientriamo nel terzo caso del teorema e si ha: $\mathbf{T(n) = \Theta\ (n^2)}$ per $\varepsilon>0$ e $3/4<= c <1$.
4. **Risposta**:
   a. Il grafo è connesso quindi esiste un'unica componente connessa.
   b. L'albero BFS:



   c. Tutti gli archi del grafo sono *archi dell'albero* BFS. Fanno eccezione:
      - (1,2) e (4,7): arco *dello stesso livello;*
      - (5,4) e (5,6): *archi tra livelli consecutivi*.
5. **Risposta**:

Omessa.

6. **Risposta:**

La soluzione è una variante della ricerca binaria: deve evitare di effettuare chiamate ricorsive su segmenti S dell'array che (a) contengono chiavi k e (b) l'elemento nell'array che precede S e quello che succede a S nell'array sono uguali a k.

```
conta( a, sx, dx, k):
if (sx > dx) return 0;
cx = (sx+dx)/2;
if (k < a[cx])
   return conta(a, sx, cx-1);
else if (k > a[cx])
   return conta(a, cx+1, dx);
else {
   c = 1;
   if (k == a[sx])
      c = c + (cx-sx);
   else
      c = c + conta(a, sx, cx-1);
   if (k == a[dx])
      c = c + (dx-cx);
   else
      c = c + conta(a, cx+1, dx);
   return c;
}
```

Risultati

Matricola	Cognome	Nome	Anno frequenza	Voto prova scritta (tra 6 e 10)
1002737	BUSI	ANDREA	2010 / 2011	7,5
1002715	CARMINATI	NICOLA	2010 / 2011	6
1007601	CATTANEO	STEFANO	2012 / 2013	7,5
1007603	DURELLI	ANDREA	2011 / 2012	9
1002637	FORLANI	MAURO	2010 / 2011	7
1007414	MARRAZZO	FRANCESCO	2011 / 2012	6
1002662	SANTACATTERINA	MARCO	2010 / 2011	6
1007582	SCOTTI	GABRIELE	2011 / 2012	6