008AA – ALGORITMICA E LABORATORIO

Appello del 13 luglio 2010

Cognome Nome: **N. Matricola:** **Corso: A B**

Esercizio 1. *(4+3 punti)* Data la seguente funzione ricorsiva `foo`, trovare la corrispondente relazione di ricorrenza e risolverla utilizzando il teorema principale.

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

Soluzione: L'equazione è $T(n) = 3T(n/2) + O(n^2)$. Maggiorando $O(n^2)$ con cn^2, otteniamo $\alpha = 3$, $\beta = 2$ e $f(n) = cn^2$. Esiste $\gamma = 3/4 < 1$ tale che $\alpha f(n/\beta) = 3/4n^2 = \gamma f(n)$, per cui $T(n) = O(f(n)) = O(n^2)$.

Cognome Nome: **N.Matr:**

Esercizio 2. *(7+2 punti)* Dati due nodi u e v in un albero binario, definiamo la loro distanza $D(u, v)$ come il minimo numero di archi (dell'albero) che li collegano. Dato un albero binario T e un intero positivo d, progettare un algoritmo che calcoli quante coppie di nodi u e v soddisfano entrambe le condizioni:

- la loro distanza è $D(u, v) = d$;
- u è antenato di v o viceversa.

È possibile usare strutture di dati di appoggio e puntatori al padre (anche se questo non è strettamente necessario per risolvere il problema). Analizzare la complessità dell'algoritmo proposto in termini del numero n di nodi presenti nell'albero.

Soluzione $O(n^2h) = O(n^3)$, dove h è l'altezza dell'albero. Effettua una visita dell'albero e memorizza tutti i suoi nodi in un array V di n puntatori. Per ogni $i < j$, verifica che il nodo $V[i]$ sia antenato o discendente di $V[j]$ usando i puntatori al padre. In tal caso, conta quanti archi sono stati attraversati e verifica che il conteggio sia pari a d. Il costo di questa verifica è $O(h)$, che viene ripetuta per le $O(n^2)$ coppie di nodi.

Soluzione $O(n^2)$. Effettua una visita dell'albero e memorizza tutti i suoi nodi in un array V di n puntatori, insieme alla loro profondità in un array P, e alla loro numerazione in visita anticipata in un array N.[1] Per ogni $i < j$, verifica che il nodo $V[i]$ sia antenato o discendente di $V[j]$ usando gli intervalli in $N[i]$ e $N[j]$. In tal caso, la loro distanza è data da $|P[i] - P[j]|$, per cui è sufficiente verificare che $d = |P[i] - P[j]|$.

Soluzione $O(n)$, facile da implementare. Se due nodi u e v soddisfano le due condizioni indicate nel testo dell'esercizio, allora uno dei due deve essere a profondità maggiore o uguale a d. Il viceversa è anche vero: se un nodo ha profondità maggiore o uguale a d, alla deve esistere un suo antenato per cui i due nodi soddisfano le due condizioni indicate nel testo. Utilizzando tale corrispondenza, basta effettuare una visita dell'albero binario e contare quanti nodi sono a profondità maggiore o uguale a d. Se si vogliono contare la coppie (u, v) e (v, u) come ordinate, basta raddoppiare tale conteggio.

[1] Ricordiamo che l'array N contiene intervalli $[a, b]$ di interi, dove a indica il momento in cui il nodo è visitato per la prima volta e b quello in cui è visitato l'ultima volta. Non ci possono essere intervalli che si sovrappongono parzialmente: o sono disgiunti oppure uno contenuto nell'altro. In questo modo, un nodo è antenato di un altro nodo se e solo se i loro rispettivi intervalli sono uno contenuto nell'altro.

Cognome Nome: **N.Matr:**

Esercizio 3. *(1+5+2 punti)* Si consideri la gestione di tabelle hash a indirizzamento aperto mediante scansione lineare.

- Definire la formula da utilizzare per la scansione lineare, per una data funzione hash $h(k)$.
- Scrivere lo pseudocodice per il seguente inserimento di una chiave k: se la posizione correntemente esaminata risulta occupata da un'altra chiave k', allora k prende il posto di k' in tale posizione e si continua con l'inserimento di k' a partire dalle posizioni successive (in modo circolare).
- Dimostrare che la dimensione dei *cluster* primari (agglomerati) così ottenuti non cambia rispetto all'inserimento standard visto a lezione.

Soluzione: lo pseudocodice inserisce k al posto di k', spostando di una posizione in avanti (in modo circolare se siamo alla fine dell'array) tutti i successivi elementi del cluster. Nell'algoritmo standard, k va a finire in fondo al cluster, ma comunque la dimensione del cluster non cambia.

Cognome Nome: **N.Matr:**

Esercizio 4. *(1+4+2+3 punti)* Si consideri un grafo diretto G, rappresentato mediante le seguenti liste di adiacenza (con i vertici numerati a partire da 1):

```
1: 2, 3, 4
2: 4
3: 4, 6
4: 6
5: 6
6: 4
```

(a) Disegnare G sul foglio.

(b) Indicare l'ordine con cui i vertici di G sono scoperti dalle visite BFS e DFS (DFS_visit), che costruiscono un albero di copertura, a partire dal vertice 1.

(c) Disegnare ciascuno degli alberi BFS e DFS sul foglio.

(d) Per ogni arco di G, indicare la classificazione indotta dalla DFS (in avanti, all'indietro, trasversale, ...). [Nota: il punto (d) non deve essere svolto da chi è iscritto alla classe 26.]

Soluzione standard. L'unica particolarità è quella di accorgersi che l'arco $(5, 6)$ non viene attraversato dalla visita.