Politecnico di Milano
Facoltà di Ingegneria dell'Informazione
Informatica 3
Prof. Giovanni Agosta
Prof. Pier Luca Lanzi
Prof. Marcello Restelli

Appello del 9 Luglio 2008

Risolvere i seguenti esercizi, scrivendo le risposte ed eventuali tracce di soluzione negli spazi disponibili.
Non consegnare altri fogli.

COGNOME E NOME (IN STAMPATELLO)

MATRICOLA

Spazio riservato ai docenti

Esercizio 1. Una agenzia gestisce N cartelloni publicitari. I clienti possono effettuare delle richieste di affissione specificando la durata dell'affissione in giorni e il cartellone sui cui effettuarla.
Se il cartellone è libero la richiesta viene immediatamente evasa, altrimenti verrà evasa non appena possibile.

Modelizzare gli N cartelloni come una risorsa condivisa in Java, fornendo le seguenti funzionalità:

- deve essere possibile specificare il numero di cartelloni N alla creazione della risorsa
- deve essere disponibile un metodo *request()* che consenta di specificare il codice cliente, il cartellone desiderato e la durata dell'affisione
- deve essere disponibile un metodo *update()* che permetta di aggiornare lo stato corrente dei cartelloni e rimuovere le richieste scadute (per semplicità, ipotizzare che questa metodo venga chiamato dal gestore del sistema una sola volta alla fine di ogni giornata)

Una volta sviluppata la soluzione rispondere alla seguente domanda: le richieste non evase immediatamente vengono gestite nell'ordine in cui sono ricevute? In caso affermativo giustificare la risposta. In caso negativo, spiegare come andrebbe modificata la soluzione

Esercizio 2. Determinare l'output del seguente programma Python.

```
# Esercizio Base: 5 punti
def f(x,y):
    return x+y*2

def g(x,y=[1]):
    return x>y

a=[1,2]
b=['a','b']

print g(f(a,b))
print f(b,a)
print g(f(a,b),f(b,a))

# Bonus: 2 punti
def h(y):
    def f(x):
        return x+y
    return f

f=h(a)
print f(b)
```

Esercizio 3. Dare dei bound asintotici per le seguenti ricorrenze, giustificandoli (5 punti).

α) $T(n) = T(n-3) + \lg n$

β) $T(n) = 7\,T(n/3) + n^2$

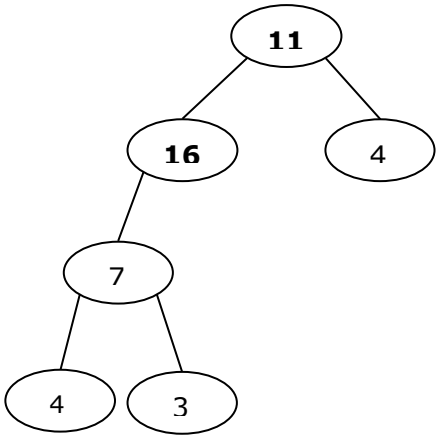
χ) $T(n) = 8T(n/2) + n^3$

Esercizio 4. Determinare il contenuto delle tabelle di hash di dimensione 16 risultanti dall'inserimento della sequenza di lettere I N F O R M A T I C A (ogni lettera è associata a una chiave numerica data dalla sua posizione nell'ordine alfabetico) nei seguenti casi:

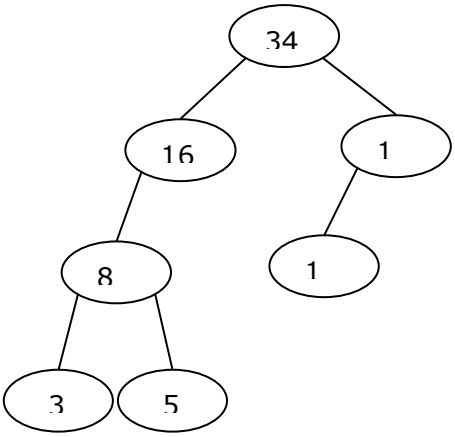
a) usando open hashing con liste non ordinate e funzione di hash: h(k) = 11*k mod 16
b) usando closed hashing con linear probing e funzione di hash: h(k) = 11*k mod 16
c) usando double hashing e funzione di hash primaria: h1(k) = 11*k mod 16 e funzione di hash secondaria h2(k) = (k mod 3) + 1

Esercizio 5. Si implementi un algoritmo in grado di stabilire se un albero binario gode della seguente proprietà: l'etichetta di tutti i nodi dell'albero (ad eccezione delle foglie) deve essere uguale alla somma fra tutte le etichette contenute nel sottoalbero sinistro e quelle contenute nel sottoalbero destro. Si assuma che tutte le etichette contenute nell'albero siano numeri interi positivi e la somma delle etichette di un albero vuoto sia pari a zero per definizione.
Valutare la complessità dell'algoritmo implementato nel caso pessimo (rispetto al numero di nodi n contenuti nell'albero binario).

Esempio



(a)



(b)

L'albero (a) non gode della proprietà richiesta per le etichette nei due nodi in grassetto, l'albero (b) invece gode della proprietà richiesta

Soluzioni

Esercizio 1

```
public class Boards {

	boolean  free[];
	String       id[];
	int          days[];

	public Boards(int n) {
		super();
		free = new boolean[n];
		id = new String[n];
		days = new int[n];
		for (int i = 0; i < n; i++) {
			free[i]=true;
			days[i]=0;
		}
	}

	synchronized void request(int board, String id, int days){

		while (free[board]==false){
			try {
				wait();
			} catch (InterruptedException e) {  e.printStackTrace();}
		}

		this.free[board]=false;
		this.id[board]=id;
		this.days[board]=days;
	}

	synchronized void update() {

		boolean freed = false;

		for (int i = 0; i < free.length; i++) {
			if (free[i]==false)
			{
				days[i]--;
				if (days[i]==0)
				{
					free[i]=true;
					freed=true;
				}
			}
		}

		if (freed)
			notifyAll();
	}
}
```

La soluzione non garantisce che le richieste vangano evase nell'ordine di arrivo. Per ottenere tale risultato è possibile utilizzare una coda per ciascun tabellone contenente gli ID dei clienti. Al risveglio ogni client dovrà verificare se è il primo in coda e solo in quel caso procederà con l'utilizzo della risorsa.

Esercizio 5

```
static int check(BinNode tree)
{
	int current = ((Integer) tree.element()).intValue();
	if (tree.isLeaf())
		return current;

	int leftSum, rightSum;

	if (tree.left()!=null)
		leftSum = check(tree.left());
	else
		leftSum = 0;

	if (tree.right()!=null)
		rightSum = check(tree.right());
	else
		rightSum = 0;

	if (leftSum == -1 || rightSum == -1 || leftSum+rightSum != current)
		return -1;
	else return current*2;
}
```

La complessità è sempre Θ(n) dal momento che è comunque necessario visitare ogni nodo dell'albero.