Politecnico di Milano
Facoltà di Ingegneria dell'Informazione
Informatica 3. Prof. Ghezzi, Lanzi e Morzenti
Seconda Prova in Itinere
30 Giugno 2004

COGNOME E NOME (IN STAMPATELLO)

MATRICOLA

Risolvere i seguenti esercizi, scrivendo le risposte ed eventuali tracce di soluzione negli spazi disponibili. NON CONSEGNARE ALTRI FOGLI.

(spazio per i docent

Esercizio 1. Sia dato un grafo G = (V, E) non orientato e connesso, avente n = |V| > 0 vertici ed m = |E| archi. Si dimostri che deve necessariamente valere la relazione $m \geq n - 1$. Si suggerisce di fornire una dimostrazione per induzione sul numero dei vertici del grafo.

Risp. Caso Base n = 1, m = 0 e` banalmente verificata.

Ipotesi induttiva: m(n) >= n-1 o anche m(n)+1 >= n
Tesi: m(n+1) >= (n+1) – 1

Dimostrazione:
se aggiungo un vertice al grafo perche` sia connesso devo aggiungere almeno un arco da quel vertice a un vertice preesistente. Quindi m(n+1) >= m(n)+1. Da cio` mettendo insieme la tesi m(n+1) >= m(n)+1 >= n cioe` prendendo il primo e l'ultimo termine della disuguaglianza m(n+1) >= n che e` la tesi.

Esercizio 2.
Si ipotizzi di voler risolvere un problema che consiste nello stabilire se una determinata sequenza di N numeri reali possiede una certa proprietà.
Il seguente programma, scritto in un ipotetico linguaggio simile a C, descrive un algoritmo per risolvere tale problema; gli elementi della sequenza vengono letti e memorizzati nell'array a in un ordine sul quale non è possibile fare alcuna ipotesi, e non si può quindi assumere che l'array risulti ordinato.

```
float  a[N];
float  x;
boolean found;
int i;
for (i = 1; i <= N; i++ )  read(a[i]);
for (i = 1; i <= k(N); i = h(i) ) {
   read(x);
   found = search(x, a);  // cerca l'elemento x nell'array a
   subp(a, found, x);
}
```

Alcune parti dell'algoritmo non sono completamente precisate. Riguardo alle parti non precisate dell'algoritmo si facciano le seguenti ipotesi:
- il calcolo della funzione k(N) abbia un costo costante, indipendente dal valore di N;
- si indichi con g(N) la complessità temporale del sottoprogramma subp(a, found, x) come funzione della lunghezza della sequenza.

Si supponga che un'analisi approfondita del problema abbia portato a stabilire un *limite inferiore* alla sua complessità, espresso dalla funzione inf(N).
Indicare, per ognuna delle seguenti combinazioni di valori o proprietà per k(N), h(i), g(N) e inf(N), la risposta alle seguenti due domande, fornendo una sintetica spiegazione

a. qual è la complessità asintotica dell'algoritmo?
b. il problema risulta aperto o chiuso sulla base dell'algoritmo fornito?

1. k(N) = N, h(i) = i+1, g(N) = $\Theta(N)$, inf(N) = $\Theta(N^2)$

a. $\Theta(N^2)$

b. chiuso

2. k(N) = N, h(i) = i+1, g(N) = $\Theta(N^2)$, inf(N) = $\Theta(N^2)$

a. $\Theta(N^3)$

b. aperto

3. $k(N) = N$, $h(i) = 2*i$, $g(N) = \Theta(N)$, $inf(N) = \Theta(N \log N)$
a. $\Theta(N \log N)$

b. chiuso

4. $k(N) = 2^N$, $h(i) = 2*i$, $g(N) = \Theta(\log N)$, $inf(N) = \Theta(N \log N)$
a. $\Theta(N^2)$

b. aperto

5. $k(N) = 2^N$; $h(i) = 2$, se $i=1$ e $h(i)=i*i$ se $i>1$; $g(N) = \Theta(1)$; $inf(N) = \Theta(\log N)$
a. $\Theta(N \log N)$

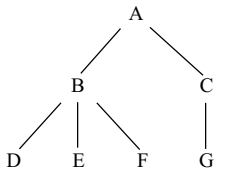
b. aperto

In quali tra i casi precedenti è possibile trovare una semplice variante dell'algoritmo presentato tale per cui il problema, che risulta aperto considerando l'algoritmo originale, diventa chiuso una volta ideata la variante? Fornire una descrizione della variante dell'algoritmo e una breve motivazione della risposta.

Risp. Nel caso 4 ($k(N) = 2^N$, $h(i) = 2*i$, $g(N) = \Theta(\log N)$, $inf(N) = \Theta(N \log N)$) il secondo ciclo for viene ripetuto N volte, la complessità di ogni ripetizione è $\Theta(N)$ a causa della procedura search (che deve effettuare una ricerca sequenziale perché l'array non è ordinato); tuttavia ordinando in precedenza l'array (costo $\Theta(N \log N)$) il costo del corpo del secondo ciclo diventa $\Theta(\log N)$ adottando una ricerca binaria, e quindi la complessità dell'intero algoritmo diventa $\Theta(N \log N)$, uguale al limite inferiore, e quindi il problema diventa chiuso.

Esercizio 3.

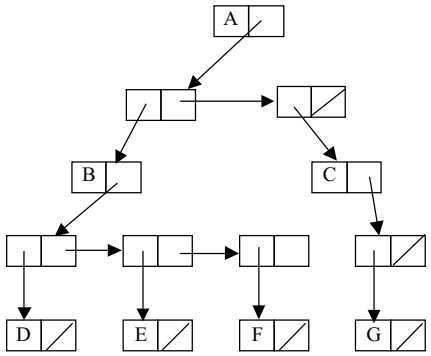
Valutare l'occupazione aggiuntiva di spazio (*space overhead*) negli alberi 5-ari (in cui ogni nodo ha al massimo 5 figli) nel caso in cui questi siano memorizzati secondo le due tecniche seguenti, mostrate in figura (si ipotizzi per semplicità che il valore memorizzato nel nodo dell'albero occupi lo stesso spazio di un riferimento a un altro nodo dell'albero)



(a)

sx		pad	dx
1	A		
3	B	0	2
6	C	0	
	D	1	4
	E	1	5
	F	1	
	G	2	

(b)



(c)

1. in un array, in cui ogni elemento dell'array che corrisponde a un nodo dell'albero contiene, oltre al valore memorizzato nel nodo, tre riferimenti ad altri nodi dell'albero (al primo figlio a sinistra, a padre e al fratello destro) sotto forma di indici nell'array stesso, come mostrato in figura (b)

Risp. I riferimenti occupano il 75% della memoria complessiva, come risulta evidente dal fatto che in ogni elemento dell'array tre parti su quattro sono occupate da riferimenti.

2. mediante puntatori, con ogni nodo collegato a una lista di figli, come mostrato in figura (c).

Risp. Si osservi che, a eccezione della radice, ogni nodo che contiene un valore dell'albero contiene anche un riferimento e inoltre è associato a un'ulteriore coppia di riferimenti; quindi, trascurando la radice, il sovraccarico è esattamente di 3/4, mentre considerando la radice esso si avvicina tanto più a tale valore quanto più l'albero è sviluppato.

Come varia l'occupazione aggiuntiva nel caso in cui il numero massimo di figli non sia 5 ma 7? Giustificare la risposta.

Risp. Il valore del sovraccarico è del tutto indipendente dal numero massimo di figli per ogni nodo.

Esercizio 4.
Data una tabella di hash di lunghezza M=12, si supponga di dover inserire (nell'ordine indicato) le chiavi: 6, 15, 27, 3, 35, con la funzione di hash h(k) = k mod M e usando la tecnica di *double hashing* per la risoluzione delle collisioni, con $h_2(k) = 1+(k \bmod 6)$. Si illustrino i risultati dell'inserimento, riportando il dettaglio dei calcoli effettuati.
Si indichi se la scelta del valore di M, della funzione di hash e della funzione h_2 è vantaggiosa o se al contrario presenta degli inconvenienti, motivando la risposta, eventualmente con degli esempi.

Risp.

6 mod 12 = 6 => 24 va in posizione 6
15 mod 12 = 3 => 15 va in posizione 3
27 mod 12 = 3 => posizione 3 occupata; p(27, 1) = 1*(1+(27 mod 6))=4; (3+4) mod 12 = 7, va in posizione 7
3 mod 12 = 3 => posizione 3 occupata; p(3, 1) = 1*(1+(3 mod 6))=4; (3+4) mod 12 = 7, posizione 7 occupata; p(3, 2) = 2*4=8; (3+8) mod 12 = 11 va in posizione 11
35 mod 12 = 11 => posizione 11 occupata; p(35, 1) = 1*(1+(35 mod 6))=6; (11+6) mod 12 = 5, va in posizione 5

La scelta dei parametri è infelice, perché M è un numero "altamente non primo" e inoltre 5 dei 6 possibili valori di h2 sono divisori di M. Infatti, se si tenta di inserire una qualsiasi chiave k tale che k mod 12 = 3, si verifica che la "probe sequence" si chiude su sé stessa (3 =>7=>11=>3…), rendendo impossibile l'inserzione, anche se la tabella è lontana dall'essere piena