Algoritmica 2 Appello 1: 11/01/2010
SOLUZIONE

Esercizio 1

Come noto il problema di scheduling di *n* processi indipendenti su *m* processor ammette un algoritmo *epsilon*-approssimato noto come LPT (longest processing time). Descrivere l'algoritmo; dare un esempio numerico della sua applicazione diverso da quello riportato nella dispensa; indicare quanto è il valore teorico di *epsilon* (teorema di Graham).

Vedi dispensa del corso, pagine 567-568**.**

Esercizio 2

Sia dato un insieme n di chiavi estratte da un universo U di dimensione |U|.

a) Descrivere come si progetta un Bloom Filter di m bits, e quale è il numero ottimo di funzioni hash per memorizzare un insieme di n chiavi in questo BloomFilter.

b) Commentare sulla comprimibilità del Bloom Filter del punto precedente (ossia uno che usa un numero ottimo di funzioni hash). [Suggerimento: Considerare la probabilità di avere 0 o 1 in una cella, e poi calcolare l'entropia del Bloom Filter visto come stringa binaria.]

c) Si considerino due insiemi A,B $\subseteq$ U aventi dimensione |A| e |B|, rispettivamente, memorizzati su due macchine MA e MB. Progettare un algoritmo distribuito che calcola l'insieme A $\cap$ B considerando DUE scenari diversi: algoritmo che usa 1 sola comunicazione tra A e B, e algoritmo che usa 2 comunicazioni tra A e B. Per ciascuno di questi 2 scenari indicare il numero di bit scambiati tra le due macchine e se la risposta calcolata è corretta sicuramente o in probabilità, e in quest'ultimo caso calcolare/discutere l'errore commesso.

Per i punti (a) e (c) si consulti la lezione sul BF.
Per il punto (b) basta osservare che se k è settato pari al numero ottimo di funzioni hash, dati n e m, allora la probabilità che una cella del BF sia 0 o 1 è pari a 1/2, per cui BF ha entropia 1e quindi risulta incomprimibile.

Esercizio 3

a) Si indichino le principali proprietà della struttura dati suffix tree di una sequenza o di un insieme di sequenze. La descrizione sia breve, ma completa: definizione della struttura dati, complessità in tempo della costruzione, complessità in spazio.

b) Si descriva a parole (il più possibile rigorosamente e brevemente) in che modo e a quali costi, un pattern P lungo m possa essere cercato in un testo T lungo n facendo uso del suffix tree.

a) Si veda ad esempio la definizione a pag.90 del materiale su Suffix Tree (file tre.pdf).
Complessita' della costruzione: lineare nella dimensione del testo.
Complessita' in spazio: lineare nella dimensione del testo.

b) Si veda ad esempio la descrizione a pagina 89 del file tre.pdf.
Complessita' della ricerca: lineare nella dimensione del pattern.

Esercizio 4

Si consideri l'algoritmo Greedy Dual Size che opera su una memoria che contiene all'inizio le 7 pagine: A, L, G, O, D, U, E. Ogni pagina p è caratterizzata da un costo c(p) che corrisponde alla posizione della lettera nell'alfabeto italiano (es. c(A) = 1) e da una size s(p) = (21 – c(p)) mod 3 +1. Si mostri l'esecuzione dell'algoritmo, cioè lo stato della memoria per la sequenza σ = ESAME.

Situazione iniziale memoria:

p	A	L	G	O	D	U	E

h(p) 1/3 10/3 7/2 13/3 4/3 19/3 5/2
L=0
E: non comporta cambiamenti
S: c(S) /h(S) = 17/2, minh = h(A)= L =1/3 h(S) = 53/6
p S L G O D U E
h(p) 53/6 10/3 7/2 13/3 4/3 19/3 5/2
A: c(A)/h(A) =1/3 minh = h(D) = L = 4/3 h(A) = 5/3
p S L G O A U E
h(p) 53/6 10/3 7/2 13/3 5/3 19/3 5/2
M: c(M)/h(M)= 11/2, c'e solo uno slot di memoria libera, minh=h(A)=L=5/3, h(M)=43/6
p S L G O M U E
h(p) 53/6 10/3 7/2 13/3 43/6 19/3 5/2
E: p S L G O M U E
h(p) 53/6 10/3 7/2 13/3 5/3 19/3 17/2

Esercizio 5
È dato un testo T composto da N parole e contenuto in memoria esterna. Per semplicità, T viene rappresentato mediante un array di N parole della stessa lunghezza constante, con un'occupazione di memoria pari a N/B blocchi. Ogni operazione di I/O trasferisce un blocco di memoria contenente B parole di Te la memoria principale può contenere M parole di T.

Descrivere brevemente (ma rigorosamente) un algoritmo efficiente per trovare le parole *distinte* che occorrono in T per ciascuna delle seguenti ipotesi, utilizzando il modello di memoria esterna (EMM) per analizzarne la complessità:

(a) non c'è alcuna conoscenza a priori sul numero K di parole distinte contenute in T;
(b) K è noto e vale K < M;
(c) parte facoltativa: K è noto e vale K = 2 x M [suggerimento: usare un albero di grado M/B e buffer di memoria].

(a) Ordinare le parole e poi effettuare una scansione per riportare una sola occorrenza per parola. Costo: sort(N) = O(N/B log_{M/B} N/B)

(b) Mantenere in memoria principale le parole distinte che si trovano durante una scansione: questo è possibile perché K < M. Costo: scan(N) =O(N/B)

(c) Come (b) solo che le parole distinte via via trovate durante la scansione vanno mantenute in un buffer tree di altezza log_{M/B} K/B =O(log_{M/B} M/B) = O(1). Costo scan(N) = O(N/B). L'albero è composto di una radice di M/B pagine usate come buffer e di altrettante foglie. S invece si usano i B-tree, il costo diventa O(N). In alternativa, si può ricopiare il testo e dividerlo in N/M gruppi di M parole consecutive: con una scansione è possibile mantenere l'invariante che gli ultimi due gruppi mantengono le parole distinte finora scandite (poiché k = 2M). Quando si esamina il gruppo successivo di M parole, si carica tutto in memoria e si ordina lasciando le parole distinte, e si effettua il merge con quelle distinte finora trovate.