Esercizi per il Corso di Algoritmica 2 (a.a. 2014/15)

Roberto Grossi
Dipartimento di Informatica, Università di Pisa
grossi@di.unipi.it

4 ottobre 2014

Sommario

Vengono raccolti i problemi discussi in classe e collegati agli argomenti a lezione. Tali problemi vengono approfonditi e osservati da più punti di vista, anche sbagliati, in quanto l'errore è funzionale all'apprendimento di situazioni complesse. La motivazione risiede nel fatto che interessa sviluppare il percorso che conduce alla soluzione (piuttosto che la soluzione stessa), sotto la guida del docente in base alle idee proposte dagli studenti. La soluzione non viene qui fornita per i motivi suddetti: è preferibile venire a ricevimento dal docente.

1. [Ordinamento in memoria esterna] Nel modello EMM (external memory model), mostrare come implementare il k-way merge ($(k+1)B \leq M$), ossia la fusione di k sequenze individualmente ordinate e di lunghezza totale N, con costo I/O di $O(N/B)$ dove B è la dimensione del blocco. Minimizzare e valutare il costo di CPU. Analizzare il costo del merge sort (I/O complexity, CPU complexity) che utilizza tale k-way merge.

2. [Esecuzione della permutazione in memoria esterna] Dati due array A e π, dove A contiene N elementi (non importa quali) e π contiene una permutazione di $\{1, \ldots, N\}$, descrivere e analizzare nel modello EMM un algoritmo ottimo per costruire un terzo array C di N elementi tale che $C[\pi[i]] = A[i]$ per $1 \leq i \leq N$.

3. [Limite inferiore per la permutazione] Estendere l'argomentazione utilizzata per il limite inferiore al problema dell'ordinamento in memoria esterna a quello della permutazione: dati N elementi $e_1, e_2, \ldots, e_N$ e un array π contenente una permutazione degli interi in $[1, 2, \ldots, N]$, disporre gli elementi secondo la permutazione in π. Dopo tale operazione, la memoria esterna deve contenerli nell'ordine $e_{\pi[1]}, e_{\pi[2]}, \ldots, e_{\pi[N]}$.

4. [Limite inferiore per la ricerca] Considerare il miglior algoritmo A possibile (passato, presente o futuro) che effettua la ricerca di una chiave mediante confronti in un insieme statico S di N elementi che obbediscono a una relazione d'ordine $<$. L'algoritmo può

effettuare un'elaborazione preliminare (preprocessing) per organizzare a suo piacimento le chiavi di S utilizzando la relazione $<$. Mostrare che la successiva ricerca di una qualsiasi chiave effettuata da A mediante confronti con $<$ richiede $\Omega(\log_B N)$ I/O (trasferimenti di blocchi di taglia B) nel modello EMM.

5. [Ricerca implicita in memoria esterna] Dato un array statico A di N chiavi in memoria esterna, descrivere come organizzare le chiavi dentro A permutandole attraverso un opportuno preprocessing, in modo che sia possibile effettuare successivamente la ricerca di una chiave con $O(\log_B N)$ trasferimenti di blocchi, chiaramente con un tempo di CPU che rimanga $O(\log N)$. Discutere il costo di tale preprocessing in memoria esterna.

6. [Ordinamento per distribuzione mediante splitter] Utilizzando il fatto che è possibile trovare $\sqrt{M/B}$ splitter per un insieme di N elementi con $O(N/B)$ trasferimenti di blocchi, descrivere come creare $\sqrt{M/B}+1$ sottoinsiemi di chiavi, separati dagli splitter. (Nota: con un solo splitter, questo equive ad avere la classica partizione del quicksort dell'input in due sottoinsiemi). Utilizzando tale distribuzione, progettare un algoritmo di ordinamento (alternativo al mergesort) in memoria esterna che richeda $O(N/B \log_{M/B} N/B)$ trasferimenti di blocchi.