Problema 2.1 [CLRS10]

Insertion sort nel merge sort su array piccoli.
Benché merge sort venga eseguito nel caso peggiore in tempo $\Theta(n\lg n)$ ed insertion sort impieghi un tempo $\Theta(n^2)$, sempre considerando il caso peggiore, i fattori costanti dell'insertion sort lo rendono più veloce per valori piccoli di n.
Quindi ha senso usare insertion sort dentro il merge sort quando i sottoproblemi diventano sufficientemente piccoli. Si consideri una modifica del merge sort in cui n/k sottoliste di lunghezza k sono ordinate usando l'insertion sort e sono poi combinate usando il meccanismo standard di fusione, con k che deve essere determinato.

a. Mostrare che le n/k sottoliste, ciascuna di lunghezza k nel caso peggiore possono essere ordinate dall' insertion sort con un tempo $\Theta(nk)$.

Poiché l'insertion sort ha un tempo di esecuzione nel caso peggiore in $\Theta(k^2)$ per ordinare una lista di k elementi, e visto che abbiamo n/k liste da ordinare in totale avremo $n/k * \Theta(k^2)$, quindi $\Theta(nk)$ nel caso peggiore.

Problema 2.1 [CLRS10]

Insertion sort nel merge sort su array piccoli
Benché merge sort venga eseguito nel caso peggiore in tempo Θ(nlgn) ed insertion sort impieghi un tempo $\Theta(n^2)$, sempre considerando il caso peggiore, i fattori costanti dell'insertion sort lo rendono più veloce per valori piccoli di n.
Quindi ha senso usare insertion sort dentro il merge sort quando i sottoproblemi diventano sufficientemente piccoli. Si consideri una modifica del merge sort in cui n/k sottoliste di lunghezza k sono ordinate usando l'insertion sort e sono poi combinate usando il meccanismo standard di fusione, con k che deve essere determinato.

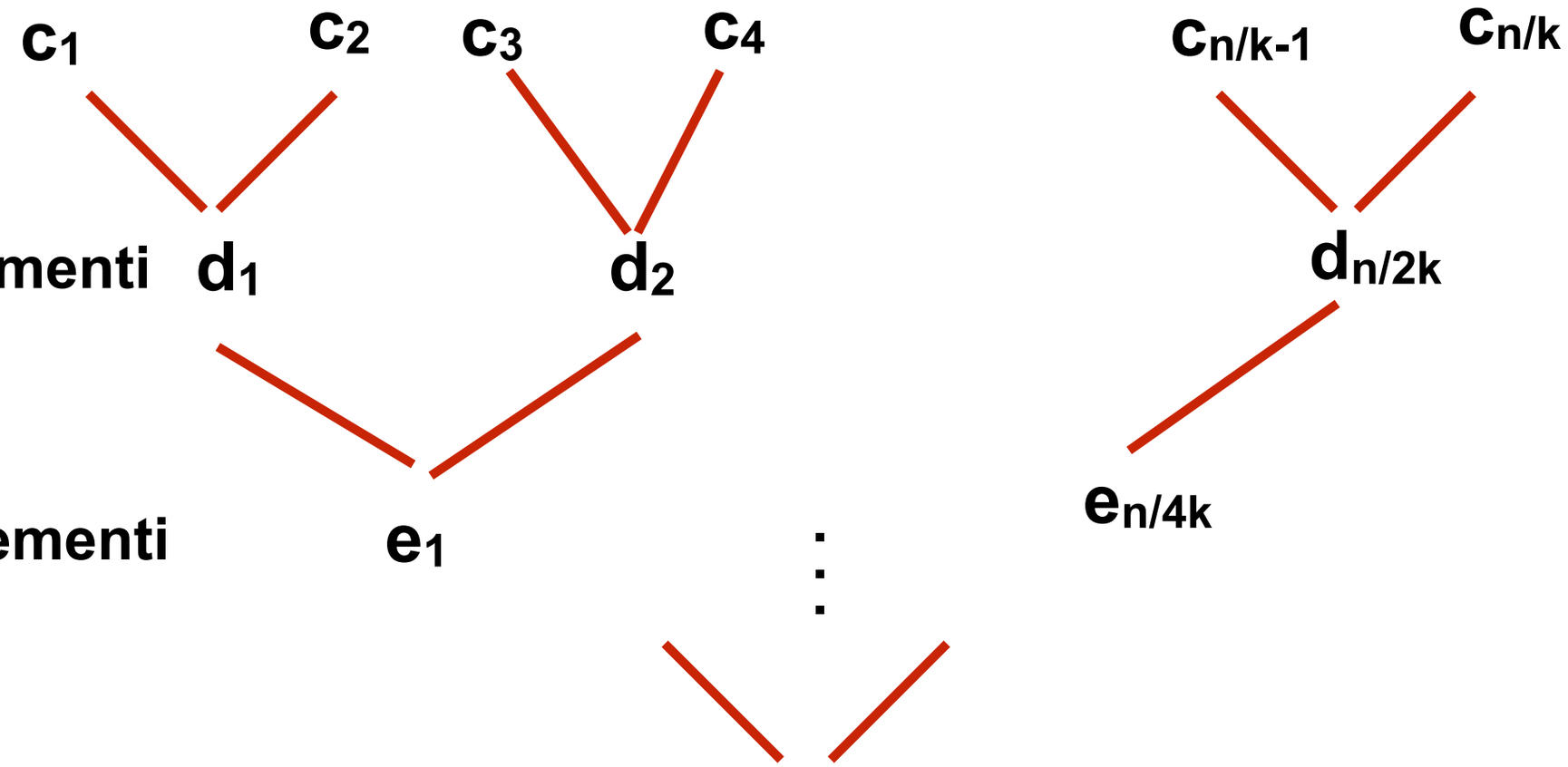
b. Mostrare che le sottoliste possono essere fuse in tempo Θ(nlg(n/k)), sempre considerando il caso peggiore.

Il modo più naturale di estendere la fusione di più array ordinati in un unico array ordinato sarebbe quello di calcolare il minimo tra tutti i primi elementi degli array ordinati e copiarlo in un nuovo array, ripetendo poi questo passo tra i primi elementi in ogni array da fondere e non ancora copiati nel nuovo array. Però calcolare il minimo tra n/k elementi ha un tempo di esecuzione Θ(n/k) e poiché andrebbe ripetuto n volte, si ottiene un tempo $\Theta(n^2/k)$.

La merge su n/k arrays

Sia $n/k = 2^j$

c_i è un array di k elementi



Si fondono gli array a due a due, poi i risultanti ancora a due a due, fino ad arrivare ad ottenere gli n elementi.
Su ogni livello complessivamente si hanno n elementi coinvolti nella fusione, e l'altezza dell'albero è pari all'esponente h di 2 tale che $2^h k = n$, da cui $2^h = n/k$ e quindi $h = j = \lg n/k$.
In conclusione per completare la fusione si impiega un tempo $\Theta(n \lg(n/k))$

Problema 2.1 [CLRS10]

Insertion sort nel merge sort su array piccoli
Benché merge sort venga eseguito nel caso peggiore in tempo Θ(nlgn) ed insertion sort impieghi un tempo $\Theta(n^2)$, sempre considerando il caso peggiore, i fattori costanti dell'insertion sort lo rendono più veloce per valori piccoli di n. Quindi ha senso usare insertion sort dentro il merge sort quando i sottoproblemi diventano sufficientemente piccoli. Si consideri una modifica del merge sort in cui n/k sottoliste di lunghezza k sono ordinate usando l'insertion sort e sono poi combinate usando il meccanismo standard di fusione, con k che deve essere determinato.

c. Dato che l'algoritmo modificato viene eseguito, nel caso peggiore, in un tempo Θ(nk + nlg(n/k)), qual è il più grande valore asintotico (notazione Θ) di k come funzione di n, per cui l'algoritmo modificato ha lo stesso tempo di esecuzione asintotico del merge sort standard?

d. Come, nella pratica, dovrebbe essere scelto il valore di k?

Deve essere k = Θ(lg n), infatti in tal caso Θ(nk + nlg(n/k)) = Θ(nlg n)

In pratica si dovrebbe scegliere k sperimentalmente trovando il valore di n per cui l'insertion sort è più veloce del mergesort.
Questo dipende dalle costanti coinvolte, per esempio se $n \leq 58$ allora risulta $n^2 \leq 10\ n \lg n$.