Problema 2.1 [CLRS10]

Si consideri ll problema di fondere m ≥ 2 array ordinati di lunghezza k in un unico array ordinato.
Mostrare che gli m array possono essere fusi in tempo Θ(nlg m), dove n = m*k, sempre considerando il caso peggiore.

Problema 2.1 [CLRS10]

Si consideri Il problema di fondere m ≥ 2 array ordinati di lunghezza k in un unico array ordinato.
Mostrare che gli m array possono essere fusi in tempo Θ(nlg m), dove n = m*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 e il secondo elemento dell'array da cui si è preso il minimo, e così via.
Però calcolare il minimo tra m elementi ha un tempo di esecuzione Θ(m) e poiché andrebbe ripetuto n volte, si ottiene un tempo Θ(n*m).

Problema 2.1 [CLRS10]

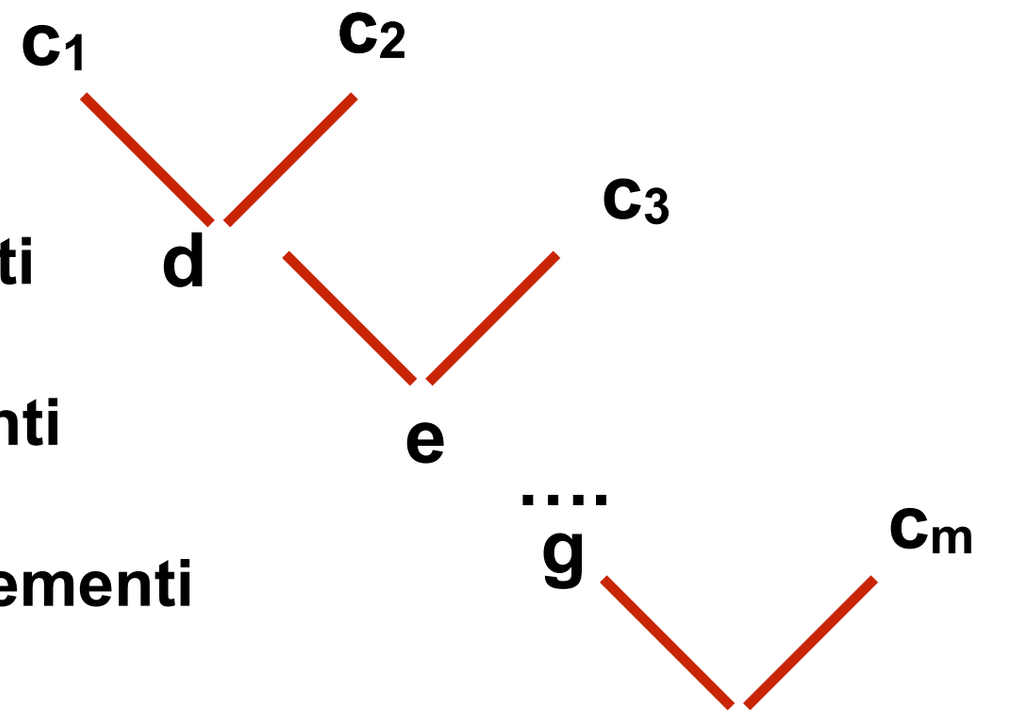
Un'altra possibilità, proposta da uno studente durante la lezione, è quella di fondere i primi due array e poi fondere il risultante array con il terzo e così via fino ad esaurimento degli array. Analizziamo questo algoritmo, arrangiando in un albero le fusioni:

c_i è un array di k elementi

d è un array di 2k elementi

e è un array di 3k elementi

g è un array di (m-1)k elementi



Al livello m-1 si fa una fusioni su 2 array di k elementi, quindi in tempo Θ(2k)
Al livello m-2 si fa una fusione 2 array, uno di k e uno di 2k elementi, quindi in tempo Θ(3k) …
Al livello 0 si fa una fusione su un array di (m-1)k elementi e uno di k, in tempo Θ(mk) = Θ(n). Sommando i tempi di tutte le fusioni si ottiene un tempo $\sum 2 \le i \le m$ Θ(ik) = Θ(m^2k) = Θ(mn). Quindi anche questo approccio va scartato.

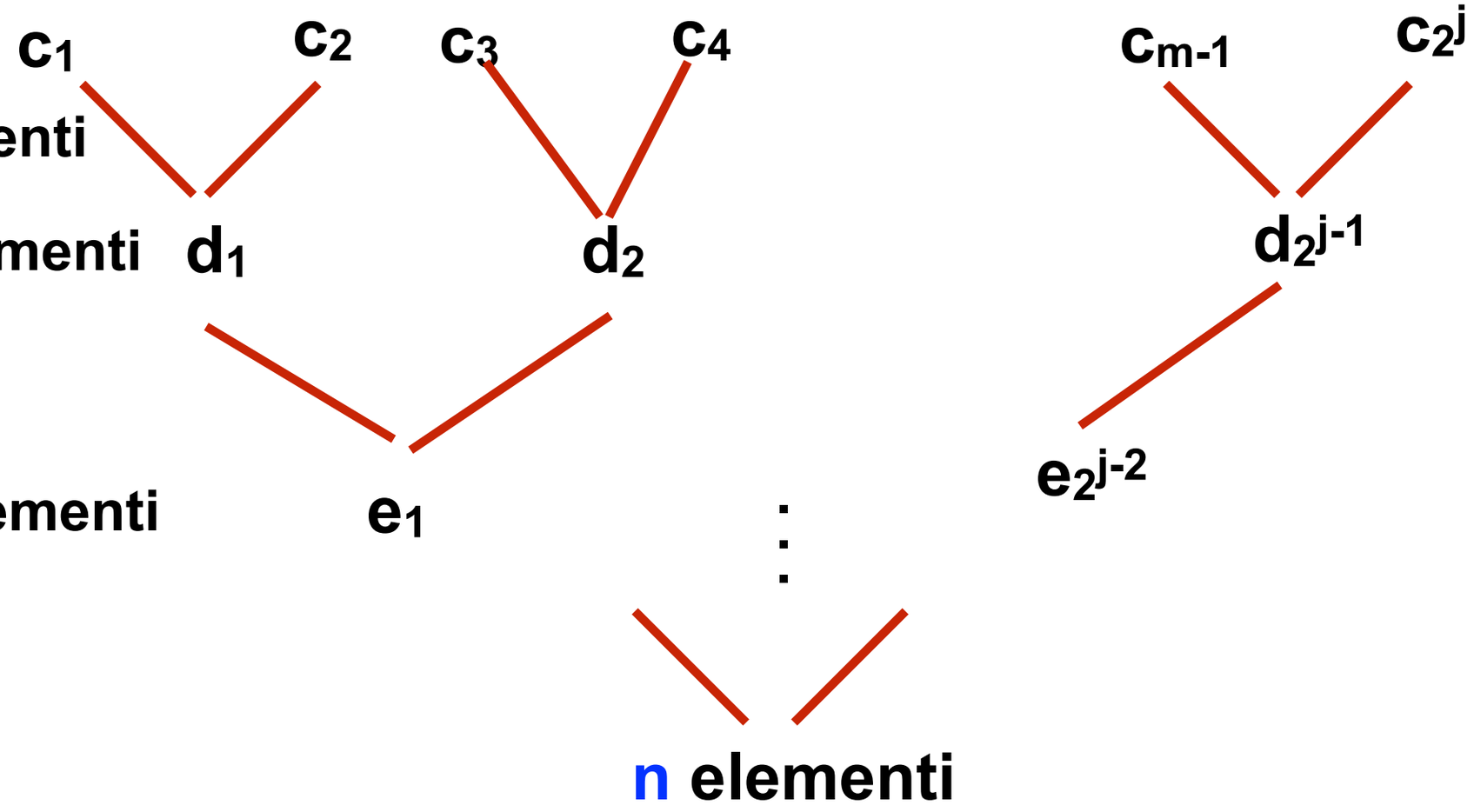
La merge su m arrays

Sia $m = 2^j$
e quindi $n = 2^j * k$

c_i è un array di k elementi

d_i è un array di 2k elementi

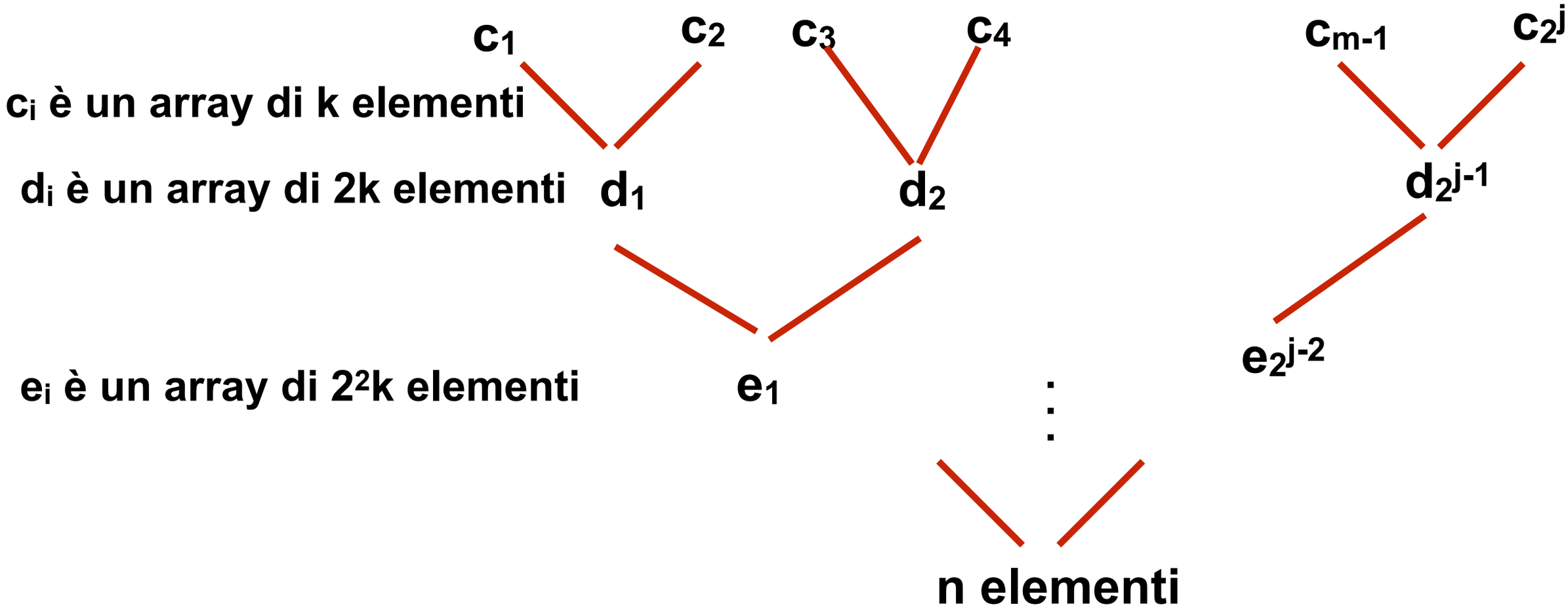
e_i è un array di 2^2k elementi



Al livello j-1 si fanno $2^{j-1}=m/2$ fusioni su 2 array di k elementi, quindi in tempo $O(m/2*2k) = O(m*k) =O(n)$
Al livello j-2 si fanno $2^{j-2}=m/4$ fusioni su 2 array di 2k elementi, quindi in tempo $O(m/4*4k) = O(m*k) =O(n)$
…
Al livello 0 si fa una fusione su 2 array di $2^{j-1}*k = n/2$ elementi, quindi in tempo $O(2*2^{j-1}*k) = O(n)$

La merge su n/k arrays

Sia m =2^j



Ogni livello comporta fusioni che complessivamente si eseguono in tempo O(n). L'altezza dell'albero è j = lg (m). Se $2^j \leq m < 2^{j+1}$ vorrà dire che l'ultimo array a destra su un livello i coinvolto nella fusione avrà meno di $2^{j-(i+1)}*k$ elementi. In conclusione in generale per completare la fusione si impiega un tempo Θ(n lg (m)), ottenuto sommando i tempi di tutte le fusioni.