

22/01/2018

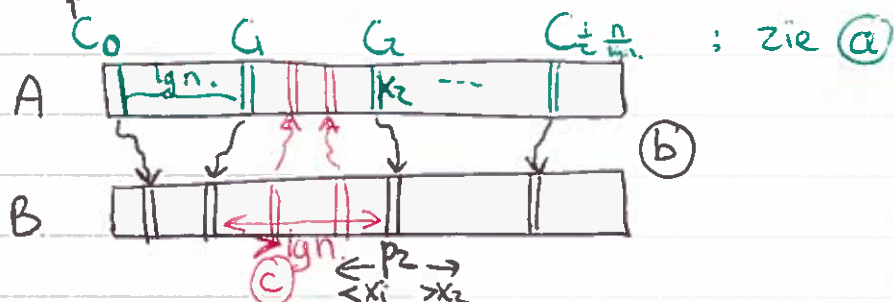
## Optimale Merge in Logaritmische Span

Stel als doel: Optimale, snelle merge.

Dwz:  $\lg n$  span en  $n$  Work.

Parallellisme is  $\frac{n}{\lg n}$  dus adagio:

Probeer  $\frac{n}{\lg n}$  cores aan het werk te houden!



Stel A en B hebben elk  $\frac{n}{2}$  elementen en zijn gesorteerd. Ik schakel  $\frac{n}{2 \lg n}$  cores in en maak ze elk verantwoordelijk voor een blok van  $\lg n$  elementen uit A.

@ Voor  $i = 0 \dots \frac{n}{2 \lg n}$ ,

laat  $C_i$  beginnen met element  $x_i = A_{i \cdot \lg n}$ .

Het eerste dat  $C_i$  doet is een Binary Search met  $x_i$  in B, zie (b)

$$p_i = \text{BinSearch}(x_i, B)$$

Hierna is bekend hoeveel elementen er zijn die  $\leq x_i$  zijn:  $i \cdot \lg n$  in A

+  $p_i$  in B.

Dus: Taak voor Core  $i$  om te mergen  
vanaf  $i \cdot \lg n$  in A met  
vanaf  $p_i$  in B naar  
vanaf  $i \cdot \lg n + p_i$  in C.

De lengte van dit merge-taakje:  
exact  $\lg n$  elementen uit A  
gemiddeld  $\lg n$  elementen uit B

De gemiddelde omvang van de merge-taak is dus  $O(\lg n)$  en kan in  $\lg n$  span door Core  $C_i$  worden gedaan!

De BinSearch is ook  $\lg n$ , dus totaal  $O(\lg n)$  tyd voor elk van de  $\frac{1}{2} \frac{n}{\lg n}$  cores. Precies wat we willen!

Probleem nog: de merge-taakjes zijn wel  $\lg n$  gemiddeld maar niet  $\lg n$  worst case!

Zie Vb waar  $C_i$  wel  $\lg n$  elt<sup>n</sup> uit A heeft, maar veel meer uit B.

Als Core  $i$  merkt dat zijn merge-taak te lang gaat duren roept hy hulp in!

Kyk naar de lengte van je B-stuk:

als  $p_{i+1} - p_i > K \cdot \lg n$ ,

dan vraag je  $K$  extra "cores" (taken) aan.

Je gaat de extra core  $K_j$  laten zoeken met  $B_{p_i+j \cdot \lg n}$  in A (zie c)

Dit verdeelt je taak in  $K+1$  stukjes die zowel aan de A- als de B-kant max  $\lg n$  elt<sup>n</sup> hebben.

Feitje (even zelf bewijzen): aantal extra  $K$ -cores  $\leq \frac{1}{2} \frac{n}{\lg n}$

Totale tyd:

- constant voor  $x_i$  bepalen (a)
- $\lg n$  voor de BinSearch (b)
- constant voor bepalen extra cores
- $\lg n$  voor BinSearch (c)
- $\lg n$  voor mergen

Totaal dus  $O(\lg n)$  tyd op  $O(\frac{n}{\lg n})$  taken.  
De Optimale Merge in  $\lg n$  tyd!