Vorlesung 5

Das ANSV-Problem

Gegeben n Elemente $a_1, \ldots, a_n$, bestimme zu jedem Element die nächsten größeren Elemente (seine „Dominatoren“) links und rechts von ihm (falls sie existieren) (Abb. 5.1).

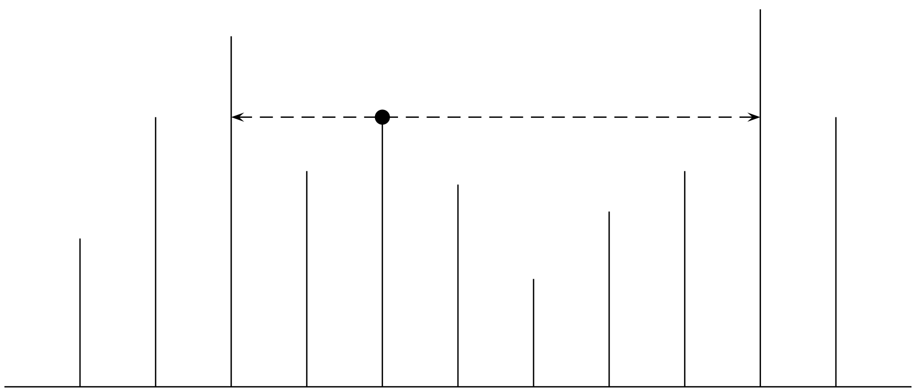

Abbildung 5.1

ANSV = all nearest smaller value (aber wir benutzen „größer“ statt „kleiner“)

Lemma 5.1: *ANSV-Probleme der Größe n können auf einer CREW PRAM mit n Prozessoren in $O(\log n)$ Zeit gelöst werden.*

Beweis: O.B.d.A. sei n eine Zweierpotenz und die Eingabeelemente paarweise verschieden. Wir speichern die Elemente $a_1, \ldots, a_n$ von links nach rechts an den Blättern eines vollständigen binären Baumes. Dann berechnen wir für jeden Knoten das Maximum über die Blätter in seinem Teilbaum (leicht) (Abb. 5.2).

Wir setzen jetzt einen Prozessor auf jedes Blatt a_i. Der Prozessor läuft im Baum zum rechten (z.B.) Dominator von a_i, wobei er die im Baum abgespeicherte Information zum

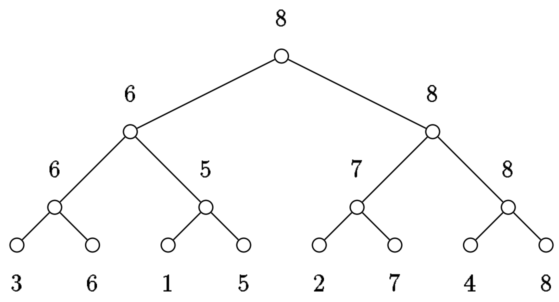

Abbildung 5.2

„Navigieren“ benutzt:

Zuerst läuft er hoch, bis er von links in einen Knoten trifft, dessen rechter Sohn einen Wert $> a_i$ hat. Dann läuft er wieder hinunter, wobei er sich so weit wie möglich links hält, aber nur Knoten mit Werten $> a_i$ besucht.

Das Blatt, an welchem er ankommt, ist der rechte Dominator von a_i (Abb. 5.3).

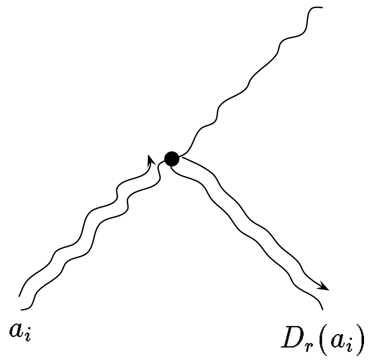

Abbildung 5.3

∎

Satz 5.2: *ANSV-Probleme der Größe n können auf einer CREW PRAM in $O(\log n)$ Zeit mit $O(n)$ Operationen gelöst werden.*

Bemerkung: Hier ist das Optimalisieren ziemlich schwierig.

Beweis: Wir teilen die n Elemente in $O(n/\log n)$ Gruppen der Größe $O(\log n)$ und lösen das lokale Problem innerhalb jeder Gruppe sequentiell (Übungsaufgabe 6.1).

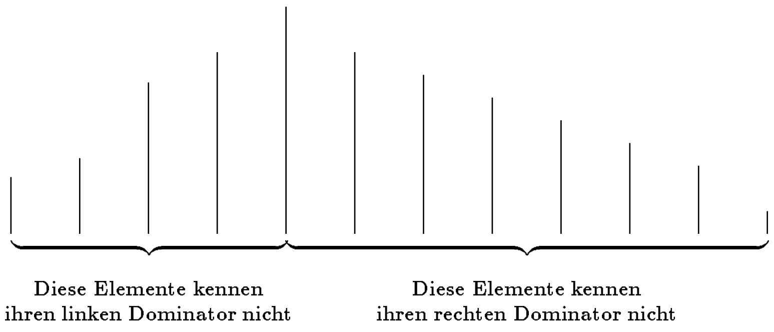

Abbildung 5.4

Jetzt sieht jede Gruppe aus wie in Abb. 5.4 (NB: Nach Komprimieren mit Präfix-Summieren).

Löse jetzt das ANSV-Problem, das durch die „Bergspitzen"(die Gruppenmaxima) gegeben ist, mit Hilfe des nicht-optimalen Algorithmus (Abb. 5.5).

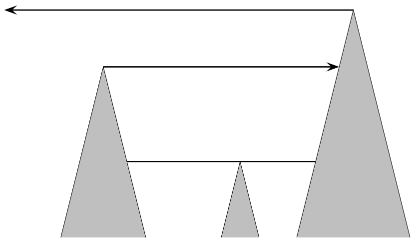

Abbildung 5.5

Wo finden sich die rechten Partner einer rechten Bergseite (Abb. 5.6)?

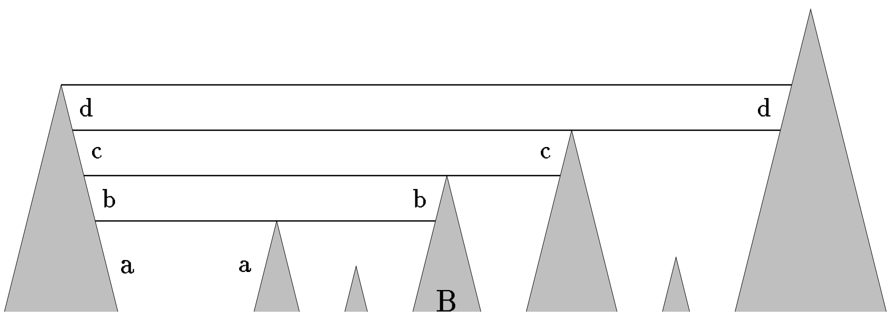

Abbildung 5.6

Durch Mischen gleichgenannter Segmente (a mit a, usw.), die ja sortiert sind, können wir alle jeweiligen Dominatoren bestimmen. Jedes Mischproblem hat Größe $O(\log n)$, so daß es sequentiell in Zeit $O(\log n)$ gelöst werden kann. Wir müssen jetzt nur noch die Mischprobleme identifizieren und einen Prozessor auf jedes Mischproblem ansetzen.

Es gibt nur $O(n/\log n)$ Mischprobleme, denn jedes Mischproblem wird durch eine Bergspitze nach oben beendet (wodurch sollte es sonst beendet werden?). Wir können leicht einen Prozessor auf jede Bergspitze setzen, der dann das betreffende Problem löst (einen links, einen rechts). Dieser Prozessor kennt den eigenen Berg, aber auch den Partnerberg, mit dem gemischt werden soll, denn das ist einfach der Dominatorenberg des eigenen Berges. Er weiß zunächst nicht, welches Stück mit welchem Stück zu mischen ist. Dazu muß er den höchsten Berg zwischen sich selbst und dem Partnerberg kennen (Bsp.: Für das Mischen der c-Stücke muß man den Berg B kennen). Dieser Berg in der Mitte kann sich aber selbst melden, denn er hat als einziger

$$\text{Linker Dominatorberg (Rechter Dominatorberg)} = \text{Eigener linker Dominatorberg.}$$

Jeder Berg, für den diese Bedingung zutrifft, meldet sich also bei seinem rechten Dominatorberg, dessen Prozessor dann sein Mischen ausführen kann (meldet sich kein Berg, ist das Mischproblem nicht nach unten beschränkt). ∎

Schneller geht es nicht auf der CREW PRAM. Auf der CRCW PRAM können wir schneller sein.

Mit n^2 Prozessoren kann das Problem in konstanter Zeit gelöst werden. Wir versuchen es daher wieder mit folgender Struktur:

(1) Bilde $O(\sqrt{n})$ Gruppen zu je $O(\sqrt{n})$ Elementen.

(2) Löse das Problem rekursiv innerhalb jeder Gruppe.

(3) Löse das durch die Gruppenmaxima definierte ANSV-Problem.

(4) Berechne Dominatoren für alle Elemente, die noch keine haben (d.h. die Dominatoren liegen außerhalb der eigenen Gruppe).

Die Rekursionstiefe ist $O(\log\log n)$. Mit n Prozessoren können wir Schritt (3) in konstanter Zeit ausführen. Danach sind wir in der Berglandschaft von oben. Aber Vorsicht: Wollen wir schneller als $\Theta(\log n)$ sein, können wir die Elemente, die schon Dominatoren gefunden haben, nicht mit Hilfe von Präfix-Summieren wegwerfen — das wäre zu langsam. Diese Elemente behindern uns beim Mischen, denn sie verletzen die Monotonieeigenschaft.

Lösung des Problems: Ersetze jede linke Bergseite durch die Folge ihrer Präfix-Maxima; jetzt wächst sie in der Tat monoton. Ersetze analog jede rechte Bergseite durch die Folge ihrer Suffix-Maxima (Abb. 5.7).

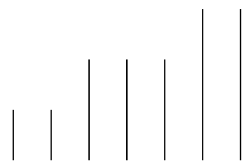

Abbildung 5.7

Präfix-Maxima können in $O(\log\log n)$ Zeit berechnet werden (Übungsaufgabe 3.4). Jetzt können wir entsprechende Segmente mischen; da das Mischen den Rang in der anderen Folge bestimmt, liefert uns das immer noch die gewünschte Information (Überlegen Sie!).

Jede Rekursionsstufe kann somit in Zeit $O(\log\log n)$ ausgeführt werden. Das liefert uns einen Algorithmus mit Laufzeit $O((\log\log n)^2)$; optimaler Speedup kann wie im Beweis von Satz 5.2 erreicht werden.

Wir können die Laufzeit weiter drücken. Aber jetzt wird's kompliziert.

Satz 5.3: *ANSV-Probleme der Größe n können auf einer CRCW PRAM in $O(\log\log n)$ Zeit mit $O(n)$ Operationen gelöst werden.*

Beweis: Die Struktur des $O((\log\log n)^2)$-Zeit-Algorithmus kann man sich als Baum vorstellen. Jeder Knoten im Baum repräsentiert ein Teilproblem bzw. ein Teilarray des Eingabearrays. Die Wurzel repräsentiert das ursprüngliche Problem bzw. das gesamte Eingabearray.

Die Blätter werden mit den Eingabeelementen identifiziert. Ein Blatt (bzw. Eingabeelement) **gehört** zu einem Knoten, wenn es in dessen Unterbaum vorkommt.

Der Baum wird von der Wurzel her aufgebaut, aber dann von den Blättern zur Wurzel abgearbeitet. Auf jeder Stufe im Baum verbrauchen wir $\Theta(\log\log n)$ Zeit, um Mischprobleme zu lösen.

Die Grundidee des schnelleren Algorithmus ist es, sämtliche auftretende Mischprobleme **gleichzeitig** zu lösen. Das sieht zunächst erstaunlich aus, denn die Ausgabe eines Mischvorgangs beeinflußt die Eingabe des nächsten Mischvorgangs.

Nehmen wir an, wir haben n Elemente und $\Theta(n\log\log n)$ Prozessoren. Damit haben wir einen Prozessor für jeden Knoten und für jedes Blatt dieses Knotens.

Dann können wir leicht für jeden Knoten v das folgende berechnen:

1. Das Maximum der Blätter von v.
2. Die Präfix-Maxima dieser Blätter.
3. Die Suffix-Maxima dieser Blätter.