

Das AKS-Netzwerk: Sortieren in $O(\log N)$

André Höing

15.01.2005

Ausarbeitung zum Seminar:
PERLEN DER INFORMATIK WS 04/05

Zusammenfassung

Atjai et al haben ein Sortiernetzwerk mit einer Tiefe von $O(\log N)$ entwickelt. Zu diesem Netzwerk wird hier eine Reihe von Vereinfachungen und Verbesserungen vorgestellt, die es erlauben, eine Abschätzung für die Konstante vor dem $\log N$ zu ermöglichen. Allerdings ist die Konstante für einen sinnvollen Einsatz noch zu groß.

Inhaltsverzeichnis

1	Einleitung	2
2	Aufbau des Netzwerkes	3
3	Definitionen und Bausteine	3
4	Bedingungen für das Netzwerk	5
4.1	Analyse der Ebenen innerhalb des Baumes	5
4.2	Grenzsituationen	9
4.3	Letzte Schritte des Algorithmus	9
4.4	Benötigte Anzahl an Schritten	10
5	Runden auf Integer	10
6	Abschätzung der Konstanten	11
6.1	Verbesserungsmöglichkeiten für die Abschätzung	12
7	Zusammenfassung	12

1 Einleitung

Es soll ein Hardwarebaustein zur Sortierung einer Menge von Zahlen entwickelt werden. In diesen Baustein sollen die N Zahlen über N Leitungen eingegeben werden und nachher liegen die Zahlen dann sortiert auf den N Ausgangsleitungen. Dieser Baustein besteht aus einem so genannten Sortiernetzwerk.

Dieses besteht aus verschiedenen Ebenen. Jede dieser Ebenen besteht aus höchstens $N/2$ kleineren Bausteinen, den Komparatoren. Wichtig ist, dass alle Komparatoren einer Ebene (auch Vergleichsschritt genannt) gleichzeitig ausgeführt werden können, da jede Leitung an höchstens einem Komparator liegt. Nach einer Ebene befinden sich die N Elemente wieder auf den Leitungen, jedoch in einer permutierten Reihenfolge. Bei einem solchen Netzwerk spricht man von einem Sortiernetzwerk, wenn nach der letzten Ebene die Zahlen in sortierter Reihenfolge auf den Leitungen liegen. Die Tiefe eines Sortiernetzwerkes wird durch die Anzahl der Ebenen bestimmt, die das Netzwerk umfasst.

Ein einfaches Netzwerk mit der Tiefe N ist „odd-even Sort“. Hierbei werden bei jedem Schritt erst die ungeraden bzw. geraden Leitungen jeweils mit ihrem Nachbarn verglichen und sortiert. Ein gutes rekursives Netzwerk ist das Batcher Sortiernetzwerk mit einer Tiefe von $(\log N)^2/2$. Abbildung 1 zeigt diese Netzwerke schematisch. Für das Sortieren von 8 Zahlen zeigt Abbildung 1 Beispiele für diese beiden Netzwerke.

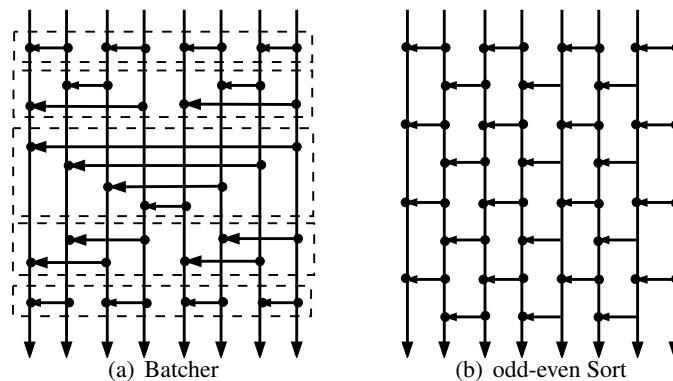


Abbildung 1: Beispiele für Sortiernetzwerke und $N = 8$

Im Jahr 1983 wurde von M. Atjai, J. Komolós und E. Szemerédi der Beweis für die Existenz eines Sortiernetzwerkes mit einer Tiefe von $O(\log N)$ erbracht [2][1]. Der Beweis ist jedoch sehr komplex, da sich die Absicht der Artikel auf den Beweis der Existenz konzentriert. Deshalb ist die dort entwickelte Konstante bei der Tiefenabschätzung sehr groß und ein so konstruiertes Netzwerk im tatsächlichen Einsatz nicht praktikabel. In dem Artikel von M.S. Paterson „Improved Sorting Networks with $O(\log N)$ Depth“ [5] wird versucht, den Schwerpunkt der Erkenntnisse von Atjai et al. nicht auf die Existenz, sondern auf die Abschätzung der Konstante zu legen. Hierfür wird die Konstruktion vereinfacht und so ein besser verständlicher Zugang zum Beweis ermöglicht. Trotzdem ist die hier entwickelte Konstante immer noch so groß, dass das Batcher[3] Netzwerk für alle praktikablen Größen weniger tief ist als das vorgestellte $O(\log N)$ -Netzwerk.

2 Aufbau des Netzwerkes

Das im Folgenden vorgestellte Netzwerk ist ein Beispiel für ein theoretisches Sortiernetzwerk mit der Tiefe $O(\log N)$.

Das Netzwerk beruht auf der Vorstellung eines Binärbaumes mit einem „Bag“, ein Platz für eine Menge von Zahlen, an jedem Knoten des Baumes. Zu Beginn des Sortierprozesses befinden sich die N Elemente innerhalb der Wurzel des Baumes. Wenn es in konstanter Tiefe möglich wäre, die Elemente in 2 Hälften zu zerteilen, so dass die kleineren $N/2$ -Elemente in der linken und die anderen in der rechten Hälfte liegen, welche jeweils an ein Kind des Binärbaumes geschickt werden, dann ist es möglich, die Elemente in $\log N$ Schritten zu sortieren. Das Partitionieren der Elemente kann jedoch nicht in konstant vielen Vergleichen geschehen, sondern benötigt $\Omega(\log N)$ Ebenen.

Deshalb benutzt die Idee, die in [2][1] vorgestellt wird, eine annähernd korrekte Partitionierung, die in konstanter Zeit erstellt werden kann, und zusätzlich wird in das Netzwerk eine Fehlerkorrektur eingebaut. Hierzu wird jeder Knoten des Baumes nicht nur in 2 sondern in 4 Teile aufgespalten. Abbildung 2 zeigt diese Aufspaltung.

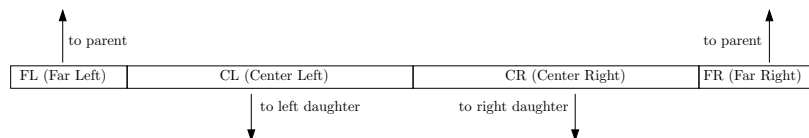


Abbildung 2: Partitionierung eines Bags

Die beiden mittleren Teile, also „Center Left“ und „Center Right“ werden im nächsten Schritt auf die beiden Töchter des Knotens verteilt, die beiden äußeren Teile, also „Far Left“ und „Far Right“ werden an den Vater des Knotens geschickt.

Die Elemente, die falsch weitergegeben wurden, sollten sich nach der Aufteilung in den beiden äußeren Teilen befinden, damit sie vom Vater aus in nächsten Schritt an die richtige Tochter geschickt werden. Mit jedem Schritt wandern mehr Elemente in Richtung Blätter des Baumes und somit zu ihrer endgültigen Position in der Sortierung.

Die Korrektheit des Netzwerkes wird in den folgenden Kapiteln durch eine Invariante gezeigt, die den maximalen Anteil an falsch sortierten Elementen innerhalb der einzelnen Knoten beschreibt. Dabei wird auch der Grad der Abweichung eines Elementes von seinem „eigentlichen“ Pfad durch den Baum berücksichtigt. Diese Abweichung wird „Strangeness“ genannt. Die in [5] und hier benutzte Definition dieser Strangeness, ist eine Vereinfachung der ursprünglichen Definition in [1].

Während der Berechnung der Tiefe werden verschiedene Parameter benötigt. Es wird jedes mal angegeben, welche Ungleichungen bei der Wahl der Parameter erfüllt sein müssen. Außerdem werden Beispiele für die Wahl der Parameter gegeben, mit denen schließlich eine Konstante vor dem $\log N$ berechnet werden kann.

3 Definitionen und Bausteine

Ein sehr wichtiger Baustein für ein solches Netzwerk ist der so genannte ϵ -Halbierer.

Definition 1 (ϵ -Halbierer)

Ein ϵ -Halbierer für m Elemente ist ein Vergleichsnetzwerk mit m Eingaben, dessen Ergebnis in einen linken und einen rechten Block aufgeteilt ist. Jeder dieser Blöcke ist gleich groß, hat also $m/2$ Elemente.

Für jede beliebige Eingabe, jedem $\epsilon > 0$ und einem $k \leq m/2$ gilt:

Die Anzahl der Elemente, die zu den k linken der Eingabe gehören, aber dennoch im rechten Block ausgegeben werden und die k rechten, die aber dennoch dem linken Block zugeordnet werden, ist kleiner als ϵk .

Ein ϵ -Halbierer kann, in Abhängigkeit von ϵ , in konstanter Tiefe gebaut werden[1].

Mit Hilfe des ϵ -Halbierers kann das nächst größere Bauteil des Sortieretzwerkes gebildet werden. Es ist ein $(\lambda, \epsilon, \epsilon_0)$ -Separator. Im eigentlichen Sinne ist der Separator ein Netzwerk aus mehreren ϵ_0 -Halbierern. Der Aufbau wird in Abbildung 3 vorgestellt. Der Separator liefert die Aufteilung des Bags in Teile, die an den Parent, bzw an die Töchter geschickt werden.

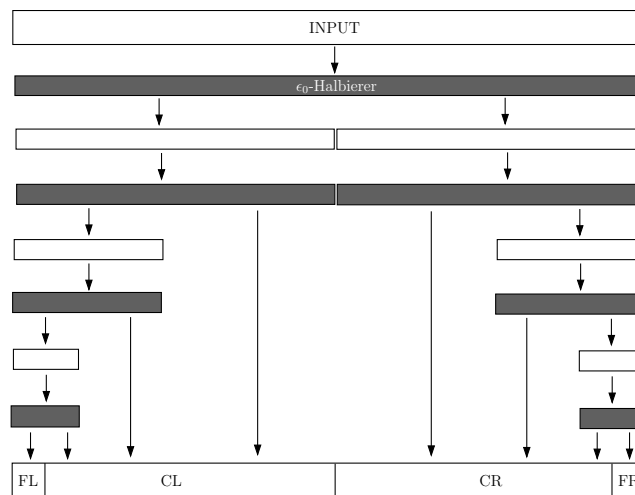


Abbildung 3: Aufbau eines $(\frac{1}{8}, \epsilon, \epsilon_0)$ -Separator

Definition 2 ($(\lambda, \epsilon, \epsilon_0)$ -Separator)

Ein $(\lambda, \epsilon, \epsilon_0)$ -Separator für m Elemente liefert eine Aufteilung der m Eingaben in vier Teile (FL, CL, CR, FR) mit den Größen $\lambda m/2$, $(1 - \lambda)m/2$, $(1 - \lambda)m/2$, $\lambda m/2$ und den Eigenschaften:

- für ein beliebiges k , $k \leq \lambda m/2$ ist die Anzahl der k linken Elemente der Eingabe, die nach der Aufteilung nicht in FL liegen, kleiner als ϵk .
- für ein beliebiges k , $k \leq \lambda m/2$ ist die Anzahl der k rechten Elemente der Eingabe, die nach der Aufteilung nicht in FR liegen, kleiner als ϵk .
- für ein beliebiges k , $k \leq m/2$ ist die Anzahl der k linken Elemente der Eingabe, die nach der Aufteilung nicht in FL und CL liegen, kleiner als $\epsilon_0 k$.

- für ein beliebiges k , $k \leq m/2$ ist die Anzahl der k rechten Elemente der Eingabe, die nach der Aufteilung nicht in FR und CR liegen, kleiner als $\epsilon_0 k$.

Ein solcher Separator kann mit Hilfe der ϵ_0 -Halbierer einfach mit einer konstanten Tiefe aufgebaut werden. Nach dem Bauplan (vgl. Abbildung 3) kann ein $(2^{-(p+1)}, p\epsilon_0, \epsilon_0)$ -Separator gebaut werden, wobei p die Anzahl der ϵ_0 -Halbierer-Ebenen ist. Diese produzieren $2p$ Blöcke für die Ausgabe. Die äußersten Blöcke werden zu FL und FR und die restlichen zu CL und CR vereinigt. Der 2. Parameter ist genau $p\epsilon_0$, da auf jeder Ebene der ϵ_0 -Anteil falsch sortiert werden kann.

Beispiel 1

Als Beispiel werden folgende Parameter gewählt: $p = 4$, $\lambda = 1/8$, $\epsilon_0 = 1/72$, $\epsilon = 1/18$.

4 Bedingungen für das Netzwerk

Das Netzwerk, kann, wie bereits erwähnt, als Binärbaum betrachtet werden. Jeder Knoten enthält einen Bag, mit einer bestimmten Kapazität von Elementen, die mit ihm partitioniert werden können. Meist ist während des Algorithmus immer abwechselnd eine Ebene des Baumes leer und die nächste voll. Die Kapazität der einzelnen Bags auf Ebene d des Baumes, kann mit $r \cdot A^d$ beschrieben werden. Dabei ist die Wurzel Ebene 0, A konstant und r eine mit jedem Schritt abnehmende Variable.

Beispiel 2

Im Beispiel wird $A = 3$ gewählt.

4.1 Analyse der Ebenen innerhalb des Baumes

Der Algorithmus wird schrittweise ausgeführt, beginnend mit Schritt 1. Bei einer ungeraden Schrittzahl sind die ungeraden Ebenen leer, bei einer geraden Zahl die geraden Ebenen. Bei jedem Schritt werden die vollen Bags des Baumes jeweils mit einem Separator geteilt und die Teile zu dem Vater, bzw. den Töchtern geschickt.

Die Kapazitäten der Bags werden Schritt für Schritt reduziert. Betrachte einen leeren Bag mit Kapazität b , der nach einem Schritt die neue Kapazität vb besitzt. Wie in Abbildung 4 dargestellt, ergibt sich:

$$vb = \underbrace{2\lambda bA}_{\text{von den Töchtern}} + \underbrace{(1-\lambda)b/2A}_{\text{vom Vater}}$$

Die Kapazität der Bags muss sich in jedem Schritt verringern, da die Elemente sich langsam in Richtung Blätter bewegen sollen. Deshalb muss $v < 1$ gelten. Also lautet die erste Bedingung für unser Netzwerk:

$$v = 2\lambda A + (1-\lambda)/(2A) < 1 \quad (1)$$

Die Kapazität eines Bags auf Ebene d des Baumes nach Schritt t kann also mit $c \cdot v^t \cdot A^d$ mit einer Konstante c berechnet werden.

Beispiel 3

Für die bereits gewählten Parameter gilt: $v = 43/48$.

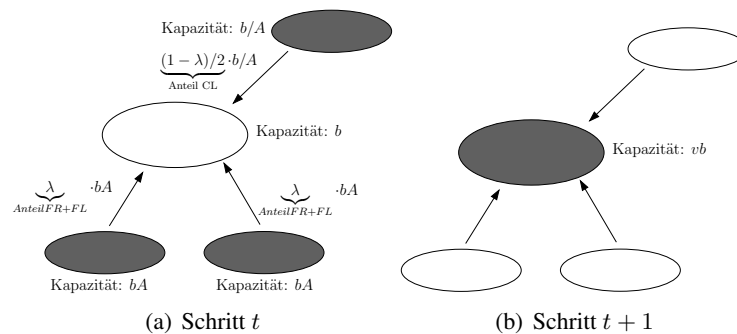


Abbildung 4: Kapazitäten während des Algorithmus

Jeder Bag des Baumes darf eigentlich immer nur eine von Beginn an vorbestimmte Menge der Eingabe enthalten (wie bei einer perfekten Sortierung). Die Wurzel darf alle Elemente enthalten, die linke Tochter nur die $m/2$ kleinsten Zahlen, die rechte die $m/2$ größten usw. Da der Separator nicht perfekt ist, können jedoch „falsche“ Elemente in einen Bag geraten. Solche Elemente haben dann einen Grad an „Strangeness“ erreicht. Dieser Grad wird über die Anzahl der Knoten bestimmt, die diese Zahl aufwärts zurücklegen muss, um wieder in einem Bag mit der korrekten Zahlenmenge zu landen. Wird also die Zahl das erste mal falsch sortiert erhält sie Strangeness 1, wird sie dann anstatt wieder ein Bag hoch, tiefer in den Baum geschickt, erhält sie Strangeness 2. Allgemein gelten folgende Regeln:

1. alle Elemente in der Wurzel haben die Strangeness 0.
2. die Strangeness wird um 1 verringert, wenn ein Element mit einer Strangeness größer 0 zum Vater geschickt wird.
3. wenn ein Element zu einer Tochter geschickt wird, erhöht sich seine Strangeness um 1, **außer** wenn seine Strangeness 0 ist **und** es zur korrekten Tochter geschickt wird.

Für jeden Bag B und einem Integer $j > 0$ kann nun zu jeder Zeit der Anteil an falsch sortierten Elementen mit Strangeness größer j bestimmt werden. Wir nennen dies $S_j(B)$. Damit das Netzwerk korrekt ist, darf dieser Anteil eine bestimmte Grenze nicht überschreiten:

$$S_j(B) < \mu \cdot \delta^{j-1} \quad \text{für alle } B \text{ und alle } j \geq 1 \quad (2)$$

μ bezeichnet dabei die Obergrenze für den Anteil der Elemente mit einer Strangeness von mindestens 1 oder mehr.

Beispiel 4

Beispielparameter: $\mu = 1/36$, $\delta = 1/40$

Wenn (2) gilt, dann können wir auch eine Aussage über die Anzahl der Elemente im nächsten Schritt für die einzelnen Bags machen. Wie bereits gezeigt, ist die Kapazität eines Bags im nächsten Schritt ($t + 1$) immer vb , wobei b die Kapazität im Schritt t ist. Weiter ist im Schritt t die Kapazität des Vaters b/A und die der Töchter bA . Im Schritt t können für $j > 1$ Elemente mit einer Strangeness von j oder mehr über 2 Wege in Bag B kommen. Einmal können es Elemente mit einer Strangeness von mindestens $j + 1$ sein, die von den Töchtern

hochgeschickt werden, und es können Elemente mit einer Strangeness von mindestens $j - 1$ sein, die vom Vater kommen. Wenn sichergestellt werden kann, dass FL und FR groß genug sind, um alle Elemente mit beliebiger Strangeness aufzunehmen, dann wird also maximal ein ϵ -tel dieser Elemente weiter falsch einsortiert und zum Bag B geschickt. Deshalb wird die Bedingung (3) gefordert.

$$\mu \leq \lambda/2 \quad (3)$$

FR bzw. FL haben jeweils die Größe $\lambda/2$ und es kann maximal einen Anteil von μ „Stranger“ geben. Mit (3) ist sichergestellt, dass im Extremfall alle Elemente mit einer Strangeness von 1 oder mehr in FL oder FR passen, um wieder auf der richtigen Weg gebracht zu werden.

Man kann nun die neue Strangeness $S'_j(B)$ von B im Schritt $t + 1$ folgendermaßen abschätzen:

$$S'_j(B) \cdot vb < \underbrace{2bA\mu\delta^j}_{\text{vom den Töchtern}} + \underbrace{\epsilon(b/A)\mu\delta^{j-2}}_{\text{vom Vater}} \quad \text{für } j < 1$$

Für diese Abschätzung von $S'_j(B)$ muss wiederum die Invariante (2) gelten:

$$\begin{aligned} 2bA\mu\delta^j + \epsilon(b/A)\mu\delta^{j-2} &< vb \cdot \mu\delta^{j-1} && | \text{ Kürzen von } \mu \text{ und } b \\ 2A\delta^j + \epsilon(1/A)\delta^{j-2} &< v \cdot \delta^{j-1} && | \cdot 1/\delta^{j-1} \\ 2A\delta + \epsilon(1/A)\delta^{-1} &< v && | \cdot \delta A \\ 2A^2\delta^2 + \epsilon &< v\delta A \end{aligned}$$

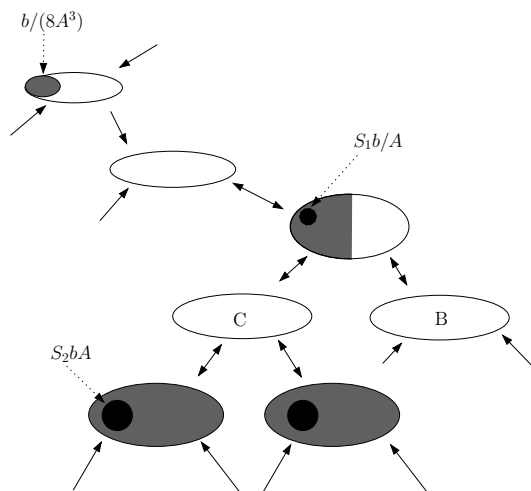
So muss eine weitere Bedingung für das Netzwerk gelten, damit die Anzahl der Stranger mit jedem Schritt abnimmt.

$$2A^2\delta^2 + \epsilon \leq v\delta A \quad (4)$$

Beispiel 5

Bei unseren bisherigen Beispielparametern: $A = 3$, $\delta = 1/40$, $\epsilon = 1/18$, $v = 43/48$ gilt diese Bedingung: $9/800 + 1/18 < 43/640$

Bisher haben wir die Einhaltung der Grenze nur für $j > 1$ gezeigt. Sie muss also noch für $S'_1(B)$ festgelegt werden. Dabei bleibt der erste Teil ($2bA\mu\delta^1$) korrekt, da weiterhin Elemente von den Töchtern geschickt werden. Es können jedoch noch weitere Fehlerquellen auftreten. Durch Elemente mit einer Strangeness größer 0 in den Knoten, kann es passieren, dass dort mehr als die Hälfte der Elemente eigentlich zu B 's Bruder C geleitet werden müssten. Alle Elemente, die mit Strangeness 0 in C gehören, werden in der Menge V zusammengefasst. Diese Menge verteilt sich anteilig der Größen der Bags wie in Abbildung 5 gezeigt auf den Baum. Die Elemente aus V liegen bei korrekter Sortierung in den Unterbäumen unter C , eine Hälfte von C 's Vater, $1/8$ von C 's Urgroßvater usw.

Abbildung 5: Fehlerquellen bei der Abschätzung von $S_1(B)$

Durch die mögliche „Überfüllung“ der einzelnen Bags mit falschen Elementen, kann es also passieren, dass Fehler auftreten. Diese Fehler werden mit dem Ausmaß der Überfüllung abgeschätzt:

- Töchter von C können maximal $S_2 b A$ Elemente außerhalb V enthalten, 2 Ebenen tiefer maximal $S_4 b A^3$ Elemente und so weiter. Also kann der Fehler in tieferen Ebenen mit $2S_2 b A + 8S_4 b A^3 + 32S_6 b A^5 + \dots < 2\mu\delta b A / (1 - 4\delta^2 A^2)$ abgeschätzt werden.
- Im schlimmsten Fall können Knoten über B keine Elemente mehr aus V enthalten und $b/(8A^3) + b/(32A^5) + \dots = b/(8A^3 - 2A)$ gilt.
- vom Vater können $S_1 b/A (< \mu b/A)$ falsch sortierte Elemente an B geschickt werden.

Zu diesen Fehlern kommen noch die neu (nach der Definition) fehlgeleiteten Elemente ($b\epsilon_0/(2A)$) des Vaters hinzu. So ergibt sich die Abschätzung:

$$S'_1(B) \cdot v < \underbrace{2\mu\delta A}_{\text{v. Kindern}} + \underbrace{2\mu\delta A / (1 - 4\delta^2 A^2)}_{\text{Überfüllung unter } C} + \underbrace{1 / (8A^3 - 2A)}_{\text{Überfüllung über } B} + \underbrace{\mu/A}_{S_1(B)\text{'s Vater}} + \underbrace{\epsilon_0 / (2A)}_{\text{vom Vater}}$$

Damit nun wieder Ungleichung (2) hält, muss eine weitere Bedingung erfüllt werden:

$$2\mu\delta A^2 + 2\mu\delta A^2 / (1 - 4\delta^2 A^2) + 1 / (8A^2 - 2) + \mu + \epsilon_0 / 2 \leq v\mu A \quad (5)$$

Beispiel 6

Mit der bisherigen Wahl von $\epsilon_0 = 1/72$ wird auch diese Bedingung erfüllt.

4.2 Grenzsituationen

Die Elemente wandern Schritt für Schritt immer weiter in die unteren Knoten des Baumes. Durch den Algorithmus kann eine nur fast gefüllte Ebene entstehen, die Ebenen darüber sind abwechselnd entweder leer oder gefüllt. Unter dieser Ebene sind alle Knoten des Baumes noch leer. Um immer nur eine fast volle Ebene zu haben, wird gefordert, dass zuerst die FL und FR Teile des Separators mit Elementen gefüllt werden und erst dann die restlichen Elemente auf die Töchter aufgeteilt werden. Damit dies zu erreichen wird, wird der Separator leicht modifiziert. Nach dem ersten ϵ_0 -Halbierer werden die beiden Hälften mit Hilfe von virtuellen Elementen wieder zu m Elementen aufgefüllt. Für die rechte Hälfte sind alle virtuellen Elemente kleiner als das kleinste echte Element und für die linke Hälfte umgekehrt. Bei Vergleichen mit einem virtuellen Element, werden diese wieder gelöscht und nur die echten Elemente weiter gegeben. So werden die echten Elemente zu den Außenseiten gedrückt und nur Elemente an die Töchter geschickt, wenn genügend vorhanden sind. Trotz der Modifikationen müssen die Strangeness-Invarianten gelten. Dies ist erfüllt, da durch diese Änderungen Elemente mit einer Strangeness von 1 oder mehr eher nach außen gedrückt werden, und so eher wieder in die richtigen Knoten geleitet werden.

Bei der Wurzel des Baumes sollte der Algorithmus am besten genau wie bei allen anderen Knoten funktionieren. Deshalb wird über ihr eine Art weiterer Knoten mit einer Menge von Elementen, die Reserve, betrachtet. Zwischen der Wurzel und diesem Knoten werden die Elemente wie bei jedem anderen Knoten ausgetauscht. In der Reserve werden keine Vergleiche angestellt. Die Strangeness Invariante ist per Definition mit einer Strangeness von 0 in der Wurzel erfüllt. Am Anfang des Algorithmus befinden sich $N(1 - 1/(4A^2))$ Elemente in der Wurzel und $N/(4A^2)$ Elemente in der Reserve.

4.3 Letzte Schritte des Algorithmus

Wie bereits festgestellt wandern die Elemente immer tiefer den Baum hinab. Die Wurzel kann, wenn sie klein genug ist, in 2 Bäume aufgespalten werden. In einem ungeraden Schritt sind alle ungeraden Bags leer und gleichzeitig kann die Anzahl S_{2j} der in der falschen Hälfte liegenden Elemente für die Ebene $2j$ abgeschätzt werden:

$$\underbrace{rA^{2j}}_{\text{Kapazität}} S_{2j} \stackrel{(2)}{<} rA^{2j}\mu\delta^{2j-1} = rA^2\mu\delta \cdot \underbrace{(A\delta)^{2j-2}}_{\text{wenn } \leq 1} \stackrel{(6)}{\leq} rA^2\mu\delta$$

Es muss also gelten:

$$A\delta \leq 1 \tag{6}$$

Jetzt kann der Zeitpunkt bestimmt werden, bei dem weniger als 1, also kein, Element in der falschen Hälfte unter der Wurzel liegt.

$$rA^2\mu\delta = 1 \quad \Rightarrow \quad r = 1/(A^2\mu\delta)$$

Beispiel 7

Mit $A = 3$ und $\delta = 1/40$ gilt: $A\delta = 3/40 \leq 1$ und somit ist der Schwellenwert $r = 1/(A^2\mu\delta) = 160$.

Somit kann der Baum an der Wurzel geteilt und für jede Hälfte kann eine Reserve und neue Wurzel entstehen. In bestimmten Abständen wird dieser Vorgang wiederholt, so dass ein Wald von unabhängigen Bäumen entsteht. Ab einer begrenzten Größe von Elementen kann ein Batchers Netzwerk [3] für eine korrekte Halbierung benutzt werden. Jeder Schritt kann daher mit der Tiefe für einen Separator und eines Batchers Netzwerkes abgeschätzt werden.

4.4 Benötigte Anzahl an Schritten

Die Kapazität der Bags auf Level $\log_2 N$ beträgt zu Beginn $\Theta(N \cdot A^{\log_2 N})$ am Ende $\Theta(1)$. Die Gesamtanzahl k an benötigten Schritten für den Algorithmus erfüllt $N \cdot A^{\log_2 N} \cdot v^k = \Theta(1)$. Also gilt:

$$k = \log_2 N \cdot \log(2A)/(-\log v) + O(1)$$

Beispiel 8

Im Beispiel: $\log(2A)/(-\log v) = \log(6)/(\log 48/43) < 17$

5 Runden auf Integer

Bisher wurden alle Abschätzungen für Mengen von Elementen durch reelle Zahlen ausgedrückt. Elemente können aber nicht geteilt werden und daher muss gezeigt werden, dass die Bedingungen auch bei ganzen Zahlen erfüllt bleiben.

Bei jedem Separator sollte die Anzahl an Elementen innerhalb eines Bags gerade sein. Dies kann einfach durch das Hinzufügen eines Elementes aus der Reserve beim Start berichtigt werden. Es werden nun einfach Regeln aufgestellt, um die Integer Größe eines Bags nicht zu weit von der idealen Größe abweichen zu lassen:

- Jeder idealerweise leerer Bag ist auch tatsächlich leer.
- Für jeden Unterbaum eines *nicht leeren* Knotens beträgt, wenn α die ideale Größe ist, die gerundete Größe $2\lceil\alpha/2\rceil$.
- Angenommen, der ideale Inhalt eines nicht leeren Knotens und seiner Unterbäume ist α und der ideale Inhalt seiner Enkel inkl. der Unterbäume sei β . Weiter sei die ideale Größe des Bags b und die gerundete $Z(b)$. Es gilt dann also $b = \alpha - 4\beta$. Nach dem bereits angegebenen Rundungsregeln gilt dann für den gerundeten Wert für die Kapazität eines Bags:

$$Z(b) = 2\lceil\alpha/2\rceil - 8\lceil\beta/2\rceil$$

Die Abweichung der Kapazitäten der Bags von der idealen Größe kann mit $b - 8 < Z(b) < b + 2$ angegeben werden.

- Bei der Wahl für die Größen von FL und FR muss gerundet werden. Dieser gerundete Wert muss im Intervall $[\lambda' m/2, m/2]$ (bei idealem $(\lambda', \epsilon, \epsilon_0)$ -Separator) liegen, damit die Abschätzungen der Fehler noch gelten. Es muss also sichergestellt werden, dass beim tatsächlichen Separator λ so gewählt wird, dass die Größe von FL und FR niemals kleiner $\lambda' Z(b)/2$ wird. Dies ist erfüllt, wenn gilt:

$$\lambda \geq \lambda' + 2/b \tag{7}$$

Beispiel 9

Mit $b \geq 160$ und $\lambda' = 1/8$ muss gelten: $\lambda \geq 0.1375$.

Es ist jederzeit möglich, FL und FR mit Elementen aus CL und CR aufzufüllen, um die gewünschten Größen der Bags zu erhalten, ohne die Fehlergrenzen zu verletzen.

6 Abschätzung der Konstanten

Für die Abschätzung wird ein Verallgemeinerung der Halbierer Notation vorgenommen. Für jedes $\epsilon > 0$ und α , $0 < \alpha \leq 1$, ist ein (ϵ, α) -Halbierer für $m = 2n$ Elemente genau wie ein (ϵ) -Halbierer definiert, außer dass die Schlüsseleigenschaft für eine Menge von k , $k \leq \alpha n$ anstatt $k \leq n$ gilt. Es liegen also von den αn rechtesten Elementen maximal $\epsilon \alpha n$ in der linken Hälfte und umgekehrt. So ist es möglich, eine Berechnung für die Tiefe des Halbierers zu machen, die unabhängig von n ist. Es gilt folgendes Lemma:

Lemma 1 (Existenz und Tiefe eines (ϵ, α) -Halbierers)

Für $0 < \epsilon \leq \frac{1}{2}$, $0 < \alpha \leq 1$ und $n > 0$, gibt es einen (ϵ, α) -Halbierer für $2n$ Elemente und einer Tiefe von $C(\alpha, \epsilon)$ mit

$$C(\alpha, \epsilon) = \lceil 1 + \frac{(h(\epsilon\alpha)) + (h((1-\epsilon)\alpha))}{-\epsilon\alpha \ln((1-\epsilon)\alpha)} \rceil$$

mit

$$h(x) = -x \ln x - (1-x) \ln x$$

Auf den Beweis des Lemmas wird hier nicht weiter eingegangen. Nachzuschlagen ist er in [5]. In dem Beweis wird eine leicht größere Tiefe C gezeigt, als es für die gezeigten Bedingungen nötig wäre, um die Analyse zu vereinfachen.

Damit die Halbierung möglichst genau ist, sollte ϵ möglichst klein sein. Konvergiert $\epsilon \rightarrow 0$, gilt:

$$C(1, \epsilon) \rightarrow -2 \ln \epsilon / \epsilon$$

und für ein festes α , $0 < \alpha < 1$:

$$C(\alpha, \epsilon) \rightarrow -h(\alpha) / (\epsilon \alpha \ln \alpha)$$

Damit kann die Tiefe des Sortiernetzwerkes ausgerechnet werden. Sie beträgt:

$$\underbrace{(\log_2 N \cdot \log(2A) / (-\log v))}_{\text{Anzahl Schritte}} \cdot C(1, \epsilon_0) \cdot p$$

Für jeden Schritt werden Separatoren mit einer Tiefe von p hintereinander geschalteten Halbierern benötigt.

Beispiel 10

Für die im Paper vorgestellten Parameter von $p = 4$, $\lambda = 1/8$, $\epsilon_0 = 1/72$, $A = 3$, $\mu = 1/36$ und $\delta = 1/34$ kann die Tiefe des Netzwerkes mit einer Zahl knapp unter $50.000 \log_2 N$ angegeben werden.

6.1 Verbesserungsmöglichkeiten für die Abschätzung

Durch Veränderungen der Parameter kann eine Verkürzung des Netzwerks erreicht werden. Mit den Parametern $p = 4$, $\lambda = 1/8$, $\epsilon_0 = 3/223$, $A = 2, 7$, $\mu = 1/16$ und $\delta = 1/34$ kann die Konstante unter 30.000 reduziert werden.

Durch eine weitere genauere Analyse des Separators wird eine weitere Verbesserung erzielt. Hier werden nur einige Ideen erläutert, weitere sind in [5] nachzulesen. Der erwartete Fehler ϵ des Separators ist die Folge der einzelnen Fehler auf jeder der p Ebenen. Sei ϵ_i der Fehler auf Ebene i und somit $\epsilon = \epsilon_1 + \epsilon_2 \cdot \dots \cdot \epsilon_p$. Weiter halbiert sich die Anzahl der Elemente und damit auch die Anzahl der falsch sortierten Elemente um die Hälfte nach jedem Halbierer. Damit kann die Anzahl der Stranger mit $\frac{\mu}{1/2^i}$ angegeben und so die Tiefe der jeweils benötigten Halbierer genauer bestimmt werden. Die Gesamttiefe eines Separator beträgt dann nicht mehr $p \cdot C(1, \epsilon_0)$, sondern:

$$C(2\mu, \epsilon_1) + C(4\mu, \epsilon_2) + \dots + C(2^p \mu, \epsilon_p)$$

7 Zusammenfassung

Das Paper [5] liefert eine Zusammenfassung von [2] und [1], um den Beweis und die Tiefenberechnung eines $O(\log N)$ Sortiernetzwerkes zu vereinfachen. Nach und nach wurden verschiedene Bedingungen mit unterschiedlichen Parametern eingeführt, die gelten müssen, um die Korrektheit des Sortierers gewährleisten zu können. Die Parameter können schon durch leichte Veränderungen grosse Auswirkungen auf die Tiefe haben. Durch eine sehr genaue Analyse und geschickter Wahl der Parameter konnte die Konstante schon auf einen Wert unter 6500 gedrückt werden, jedoch ist diese immer noch viel zu hoch. Trotz der in der O -Notation guten Laufzeit ist ein Batchers [3] Sortiernetzwerk in allen praktikablen Fällen noch immer schneller. Bis heute konnte die Konstante nur auf 1830 [4] verkleinert werden.

Es bleibt zu hoffen, dass durch weitere Ideen und Verbesserungen die Konstante weiter verringert werden kann und damit das Netzwerk benutzbar wird.

Literatur

- [1] AJTAI, M., J. KOMLOS und E. SZEMEREDI: *An $O(n \log n)$ sorting network*. In: *STOC '83: Proceedings of the fifteenth annual ACM symposium on Theory of computing*, Seiten 1–9. ACM Press, 1983.
- [2] AJTAI, M., J. KOMLOS und E. SZEMEREDI: *Sorting in $c \log n$ parallel steps*. *Combinatorica*, 3(1):1–19, 1983.
- [3] BATCHER, KENNETH E.: *Sorting Networks and Their Applications*. In: *AFIPS Spring Joint Computing Conference*, Seiten 307–314, 1968.
- [4] CHVATAL, V.: *Lecture notes on the new AKS sorting network*. Technischer Bericht, November 1992.
- [5] PATERSON, M. S.: *Improved Sorting Networks with $O(\log n)$ Depth*. Technischer Bericht, 1987.