Correction Networks

Seminararbeit

Institut für Datenverarbeitungsanlagen

Carsten Buschmann

Kleine Kreutzstr. 9

38118 Braunschweig

Matrikelnummer 2540845

Inhalt

1. Einleitung

Das Sortieren ist eines der fundamentalsten Probleme der Informatik, das vielfältig auftritt und insbesondere bei großen Datenmengen äußerst langwierig sein kann. Daher wurden in der Vergangenheit immer wieder intensive Anstrengungen unternommen, den Aufwand der verschiedenen Verfahren zu reduzieren.

Ein klassisches Verfahren ist die Anwendung eines Vergleichernetzwerkes. Neben einer langen Tradition zeichnet Vergleichernetze ihre potentielle Hardwarerealisierung aus. Ein Ansatz zur Verkürzung von Sortierzeiten ist die Berücksichtigung von Vorsortierungen. Geht man davon aus, dass eine Folge schon „fast ganz“ sortiert ist, und maximal eine bestimmte Anzahl t „Fehleinträge“ hat, lassen sich zum Teil beschleunigte Verfahren einsetzen.

Korrekturnetze (engl. correction networks) kombinieren die beiden oben beschriebenen Ideen. Bis vor Kurzem hatten die besten bekannten Korrekturnetze eine Zeitkomplexität von $4\log N + O(\log^2 t \log\log N)$. Der in dieser Arbeit vorgestellte Ansatz der Fibonacci-Netze ([1]) erzielt eine Reduzierung des Faktors vor log N auf etwa 1,44.

In den folgenden beiden Abschnitten werden zunächst verschiedene Definition mit Bezug auf Sortiernetze und Korrekturnetze vorgestellt. In Kapitel 4 wird dann ein Vergleichernetz konstruiert, das einen Fehler korrigieren kann. Im nächsten Teil wird dann ein Netz vorgestellt, das die Streuung von t Fehlern auf ein bestimmtes Maß reduzieren kann. Im sechsten Kapitel wird dann gezeigt, wie das Gesamtkorrekturnetz mit der angestrebten Tiefe konstruiert werden kann.

2. Sortiernetze

Vergleichernetze und speziell auch Sortiernetze arbeiten auf einer Menge von *Registern*, deren Anzahl als *Problemgröße* oder *Eingabegröße* anzusehen ist. Diese Register tragen als Index eine Folge von Integern. Dabei können, wenn die Indexfolge eine Länge größer als eins hat, das erste Element der Folge als Zeilenindex und die restlichen Elemente als Spaltenindex verstanden werden.

Ein Vergleichernetz ist genau dann ein *Sortiernetz*, wenn es beliebige Eingaben sortieren kann.

Bei der Anwendung eines Sortiernetzes werden die in den Registern abgelegten Werte durch mehrfaches Tauschen von je zwei Registerinhalten so angeordnet, dass die lexikographische Ordnung der Registerinhalte der der jeweiligen Registerindizes entspricht, die diese Werte enthalten.

Das Tauschen von Registerwerten erfolgt mittels so genannter *Vergleicher*, die in grafischen Darstellungen von Sortiernetzen durch Pfeile symbolisiert werden. Abbildung 2.1 zeigt beispielhaft ein sogenanntes bitonisches Sortiernetz. Ein Vergleicher kann, wenn für das Sortieren notwendig, die Werte zweier Register, die er verbindet, vertauschen. Der größere (=“schwerere“) Wert wandert in Richtung der Pfeilspitze.

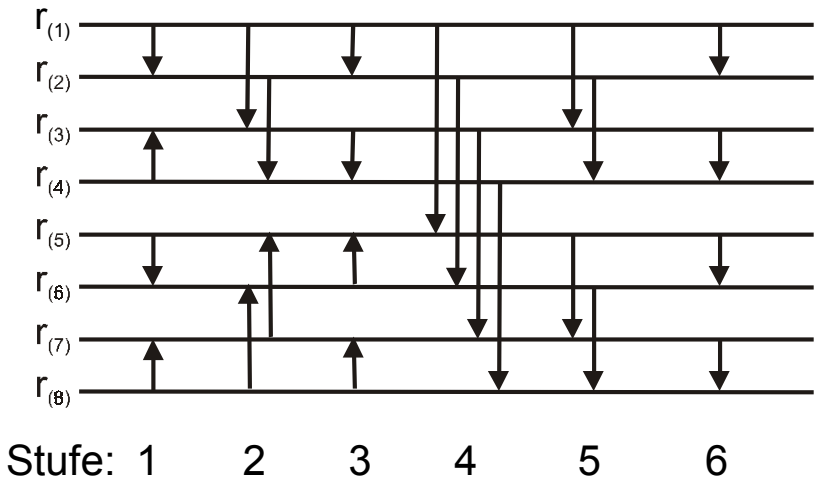

Abbildung 2.1: Bitonisches Suchnetz

Bei Vergleichern, die unterschiedliche Register verbinden, spricht man von unabhängigen Vergleichern. Sie können somit parallel arbeiten und in sogenannten *Stufen* (auch engl. layers) organisiert werden. Die Anzahl der notwendigen, hintereinander geschalteten Stufen

wird als *Tiefe* eines Vergleichernetzwerkes bezeichnet und kann als zeitliche Komplexität verstanden werden.

Eine wichtige Hilfe für die Beurteilung von Sortiernetzen stellt das *0-1-Prinzip* dar, das 1973 von Knuth entwickelt wurde. Es besagt, dass ein Vergleichernetz genau dann beliebige Schlüsselfolgen sortiert, wenn es alle Folgen sortiert, die nur aus Nullen und Einsen bestehen. Daraus folgt, dass ein Vergleichernetzwerk genau dann ein Sortiernetz ist, wenn es alle 0-1-Folgen sortiert.

Um den Beweis zu erbringen, dass ein Vergleichernetz ein Sortiernetz für N Register ist, reicht es somit, den Nachweis für 2^N statt für N! mögliche Folgen zu führen.

Ein weiterer Vorteil ist, dass eine grafische Beweisführung bzw. Veranschaulichung der Funktionsweise von Sortiernetzen möglich wird. Dazu werden die Register durch Kästchen symbolisiert, die schwarz gefärbt sind, wenn sie eine 1 einhalten, bzw. weiß, wenn sie eine 0 enthalten. Von dieser Darstellung wird im weiteren Verlauf dieser Arbeit mehrfach Gebrauch gemacht werden, um komplexe Sachverhalte grafisch nachvollziehbar darzustellen.

3. Korrekturnetze

Um die Konzepte von Korrekturnetzen verdeutlichen zu können, soll zunächst auf die Störungen in der Sortierung von Eingabefolgen eingegangen werden, die korrigiert werden sollen. Dabei geht man davon aus, dass sich in einer sortierten Folge eine kleine Anzahl von falsch einsortierten Elementen befindet. Gründe dafür können beispielsweise defekte Vergleicher in einem zuvor angewendeten Sortiernetz oder Änderungen an den Daten selbst sein.

Konkreter formuliert heißt eine 0 (1) in einer 0-1-Folge *gestört*, wenn sie zu einer 1 (0) geändert wurde. Eine 0-1-Folge heißt *t-gestört*, wenn sie aus einer sortierten Folge durch das Stören von höchstens t Einträgen hervorgeht. Eine t-gestörte Folge, in der nur Nullen gestört sind, heißt *t-partiell-gestört*. Abbildung 3.1 zeigt eine t-partiell-gestörte Folge für $t \geq 2$.

Abbildung 3.1: t-partiell-gestörte Folge

Die *Dirty Area* einer Folge ist die kleinste Menge von Registern, so dass alle Register unterhalb (Register mit kleineren Indizes) nur Nullen enthalten und alle Register oberhalb (Register mit kleineren Indizes) nur Einsen enthalten. Abbildung 3.2 verdeutlicht dieses Konzept grafisch.

Abbildung 3.2: Dirty Area

Aufbauend auf die obigen Definitionen für Folgen schließen sich nun Begriffsbestimmungen für die sie korrigierenden Netze an. So ist ein Vergleichernetz genau dann ein *t-Korrekturnetz*, wenn es alle t-gestörten Folgen sortiert. Ein Vergleichernetz ist genau dann *t-partielles-Korrekturnetz*, wenn es alle t-partiell-gestörten Folgen sortiert. Ein Netz ist ein *(t,Δ)-partielles Korrekturnetz*, wenn es die Größe der Dirty Area einer t-partiell-gestörten Folge auf maximal Δ Register reduziert.

4. Eine gestörte Position

In diesem Abschnitt soll ein 1-partielles Korrekturnetz betrachtet werden. Aufgrund der Symmetrie folgt die Behandlung einer gestörten 1 direkt aus diesem Fall. An dem vorgestellten Fibonaccinetz kann man aufgrund seiner Einfachheit leicht den Aufbau und die Funktionsweise nachvollziehen und beweisen. Bei den später vorgestellten, weit komplexeren Netzen, die auf den gleichen Prinzipien beruhen, wird dann auf formale Konstruktion und Nachweis verzichtet.

Zunächst soll die rekursive Definition der namensstiftenden Fibonaccizahlen in Erinnerung gerufen werden:

$$f_0 = f_1 = 1$$
$$f_k = f_{k-1} + f_{k-2}$$

Weiter werden φ_k und ψ_k ähnlich definiert:

$$\varphi_0 = \varphi_1 = \psi_0 = \psi_1 = 1$$

$$\psi_k = \psi_{k-1} + \varphi_{k-2}$$

φ_k = größte ungerade Zahl $\leq \psi_k$

Aus diesen Definitionen ergibt sich, dass ψ_k (und somit auch φ_k) etwas langsamer wächst als die Fibonaccizahlen.

Sei *LG(N)* die kleinste natürliche Zahl k mit $\varphi_k \geq$ N. Anschaulich verhält sich LG zu φ_k ähnlich wie eine aufgerundete Logarithmusfunktion zu einer Exponentialfunktion. Dabei gilt bezüglich des Wachstums

$$\text{LG(N)} \sim \log_{(\frac{1+\sqrt{5}}{2})} n = \frac{1}{\log_2 \frac{1+\sqrt{5}}{2}} \log n = \alpha \log n \,.$$

Dies folgt direkt aus der Ungleichung

$$f_{k-1} \leq \varphi_k \leq f_k ,$$

den Gesetzen zum Basiswechsel von Logarithmen sowie deren Proportionalität (aus [2], Formel 1.17)

$$\log_a x = M \log_b x \quad \text{mit} \quad M = \log_a b = 1 / \log_b a$$

und der geschlossen Definition der Fibonaccizahlen (aus [2], Formel 5.154)

$$f_k = \frac{1}{\sqrt{5}} \left[\left(\frac{1+\sqrt{5}}{2} \right)^k - \underbrace{\left(\frac{1-\sqrt{5}}{2} \right)^k}_{<1} \right].$$

LG(N) wird im weiteren ein Maß für die Tiefe d des konstruierten Netzes F_N sein, wobei N die Anzahl der Registern $r_{(1)}, ...,r_{(N)}$ ist. F_N besteht aus d = LG(N) Stufen $L_1, ..., L_d$. In der Definition der einzelnen Stufen

$$L_p = \{[2i + p : 2i + p + \varphi_{d-p}]\}$$

stellt [i:j] einen Vergleicher dar, der von $r_{(i)}$ nach $r_{(j)}$ zeigt, p nimmt die Werte 1,…, d und i solche ganzzahligen Werte an, dass sich gültige Registerindizes ergeben. Abbildung 4.1 zeigt ein solches Netzwerk F_8.

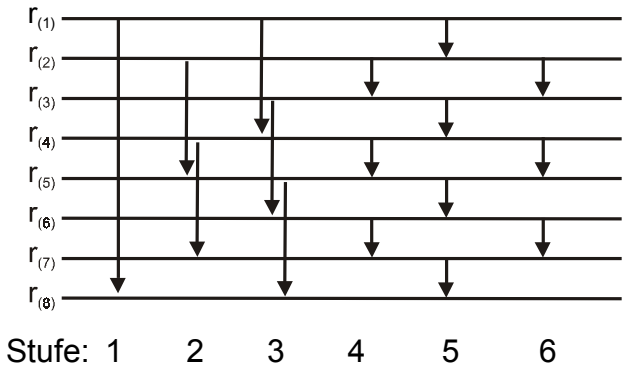

Abbildung 4.1 Fibonacci-Korrekturnetz

Der Beweis der Funktionsfähigkeit erfordert zwei weitere Definitionen: Das Register mit dem größten Index, dass eine 0 enthält, heißt *Grenze*. Der *Abstand* der verschobenen 1 zur Grenze ist die Differenz des Indexes des Registers mit der 1 und der Grenze. F_N reduziert diesen Abstand sehr effektiv. Der Nachweis, dass F_N wirklich funktioniert, wird durch Beweis der folgenden Aussage geführt:

Nach Anwendung von l Stufen von F_N ist der Abstand zwischen der verschobenen Eins und der Grenze kleiner als ψ_{d-l+1}. Wenn diese 1 sich in $r_{(i)}$ befindet, und die Parität von l und i gleich ist, ist dieser Abstand kleiner als φ_{d-l}.

Der Beweis erfolgt per Induktion über l: für l=0 ist die Aussage offensichtlich, da $\varphi_d \geq N$. Stimmt die Aussage für l – 1, so folgt sie für l: Sei i der Registerindex, in dem die verschobene Eins vor der Anwendung der Stufe l stehlt.

Wenn die Eins von Stufe l nicht bewegt wird, sind zwei Fälle zu unterscheiden. In Fall 1 ist die Parität von l und i verschieden. Es gilt, dass der Abstand kleiner als $\varphi_{d-(l-1)} = \varphi_{d-l+1} \leq \psi_{d-l+1}$ ist. Im zweiten Fall ist die Parität von i und l gleich. Daraus folgt, dass der Abstand zur Grenze kleiner als φ_{d-l} sein muss, da die Eins sonst konstruktionsgemäß von einem Vergleicher erfasst worden wäre.

Wird die Eins in Stufe l bewegt, haben i und l die gleiche Parität, und die Eins wandert um φ_{d-l} Schritte an die Grenze heran, wobei der Abstand vor Anwendung von Stufe l kleiner als ψ_{d-l+2} war. Somit ist der Abstand danach kleiner als

$$\psi_{d-l+2} - \varphi_{d-l} = \psi_{d-l+1}.$$

Somit bleibt der Abstand der Eins zur Grenze immer kleiner als oben angegeben. Die Eins bleibt immer in der sogenannten *Correction Area*, von der aus es noch möglich ist, die Grenze zu erreichen.

Insgesamt ergibt sich, dass F_N ein 1-Korrekturnetz mit einer ungefähren Tiefe von $\alpha \log N$ ist.

Im Folgenden soll noch kurz das Verhalten von F_N für den Fall beschrieben werden, dass sich mehrere Einsen nicht an der richtigen Stelle befinden. Seien zu einem bestimmten Zeitpunkt nicht weniger als t verschobene Einsen in der Correction Area. Dann sind nach Anwendung einer weiteren Stufe mindestens noch t/2 verschobene Einsen in der Correction Area. Da jetzt verschobene Einsen an einem Vergleicher aufeinandertreffen können, fällt im ungünstigsten Fall (alle verschobenen Einsen treffen aufeinander) die Hälfte der Einsen aus der Correction Area heraus. Nach Anwendung von s Stufen müssen also mindestens noch $t/2^s$ Einsen in der Correction Area sein.

5. Partielles Korrekturnetz

Es soll nun aufgezeigt werden, wie aus 1-partiellen-Korrekturnetzen ein (t, $ct(\log N)^{c_s \log t}$)-partielles-Korrekturnetz C fast gleicher Tiefe konstruiert wird, wobei c_s eine Konstante in Abhängigkeit von s ist. Abbildung 5.1 zeigt eine schematische Darstellung von C. Aufbau und Funktionsweise sollen hier lediglich grob qualitativ geschrieben werden, für eine formalere Darstellung sei auf [1] verwiesen.

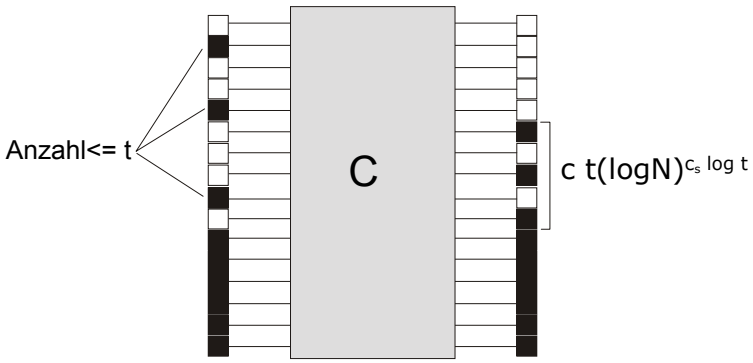

Abbildung 5.1: Schematischer Aufbau von C

Die Register werden mit Paaren indiziert, so dass man sich eine matrixförmige Anordnung vorzustellen hat. Zeilen und Spalten stehen dabei in einem bestimmten Verhältnis. Abbildung 5.2 zeigt diese Ausgangssituation.

Abbildung 5.2: Ausgangssituation

C besteht aus zwei Teilen. Der erste Teil verwendet für jede Zeile einen der in [3] beschriebenen Selektoren, die die jeweils t größten Werte einer Zeile in die obersten t Register einer Zeile transportieren (Abbildung 5.3). Dieser Teil hat eine Tiefe von $c_s \log t \log\log N$ mit $c_s \sim -\frac{1}{\ln(1-1/2^s)} \sim 2^s$.

Abbildung 5.3: Verschobene Einsen in den t letzten Spalten

Der zweite Teil von C besteht aus parallel arbeitenden Netzen F_N in jeder Spalte. Die Grundidee dabei ist, dass die maximal $t(1-1/2^s)$ Einsen, die die Correction Area nach s Stufen verlassen haben können, alle s Stufen durch eine zusätzliche Stufe in andere F_Ns (d.h. andere Spalten) transferiert werden, die vorher keine verschobenen Einsen enthielten. Abbildung 5.4 zeigt für s = 2 seitlich schematisch die Anordnung der Stufen, wobei die Stufen der F_N durch Pfeile, die Transferstufen durch graue Balken symbolisiert werden.

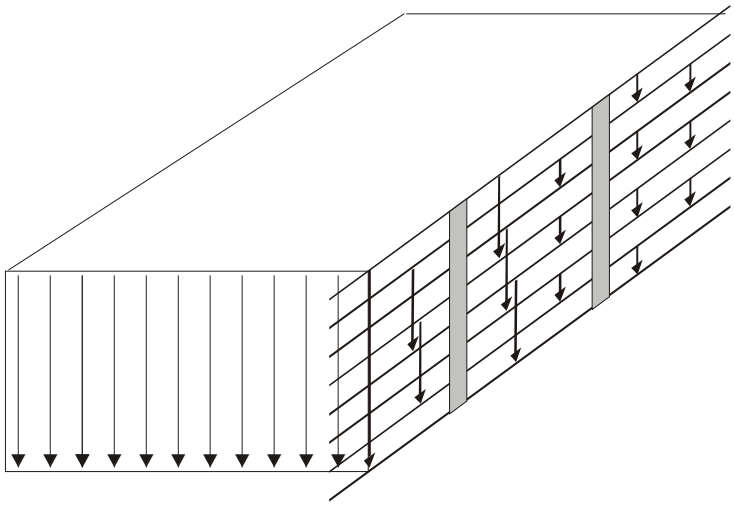

Abbildung 5.4: Aufbau des zweiten Teils von C

In den Vergleicherstufen vor der ersten Transferstufe wird nun begonnen, die verschobenen Einsen zu sortieren. Dabei verlassen einige Einsen (grau) die Correction Area (gepunktet, s. Abbildung 5.5)

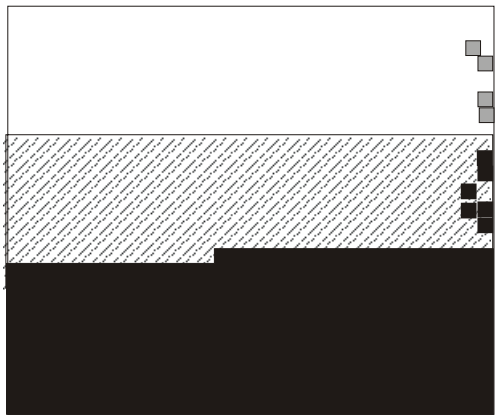

Abbildung 5.5: Register vor der ersten Transferstufe

In der ersten Transferstufe kommen Vergleicher zum Einsatz, die eben diese Einsen, die die Correction Area verlassen haben, in die Correction Area von Spalten transferieren, die bisher keine verschoben Einsen enthielten (Abbildung 5.6).

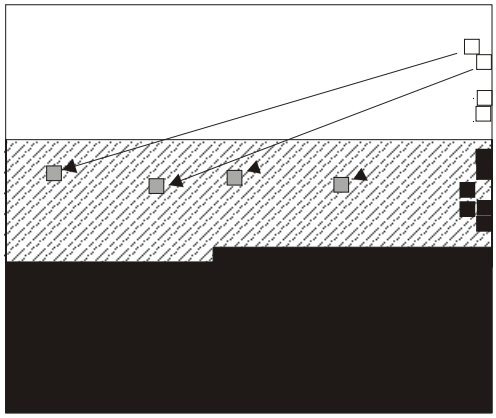

Abbildung 5.6: Erste Transferstufe

Die erste Transferstufe stellt somit die Ausgangsssituation für die nächsten Vergleicherstufen her (Abbildung 5.7).

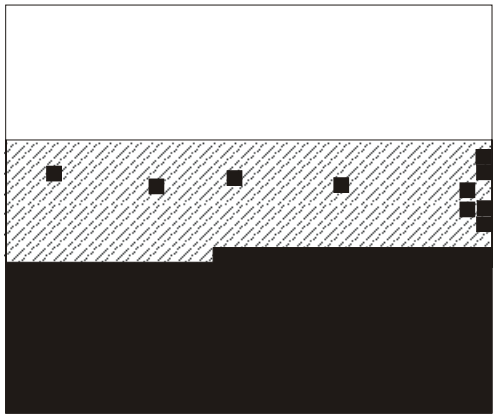

Abbildung 5.7: Situation nach der ersten Transferstufe

Vor der zweiten Transferstufe haben wiederum einige Einsen die nun nochmals kleiner gewordene Correction Area verlassen, sie werden nun wieder in die Correction Area von Spalten transportiert, die bisher keine verschobenen Einsen enthielten (Abbildung 5.8).

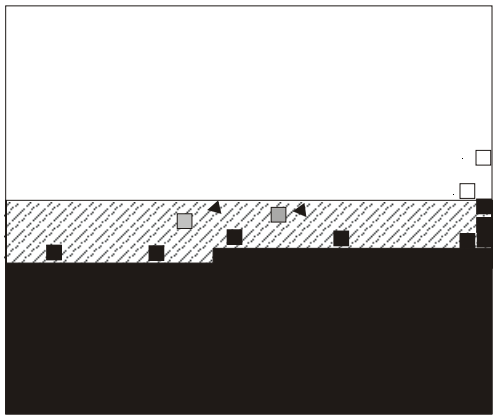

Abbildung 5.8: Transfers der nächsten Transferstufe

Dieses Verfahren wiederholt sich, bis schließlich die Dirty Area auf maximal 3 Zeilen reduziert ist (Abbildung 5.9).

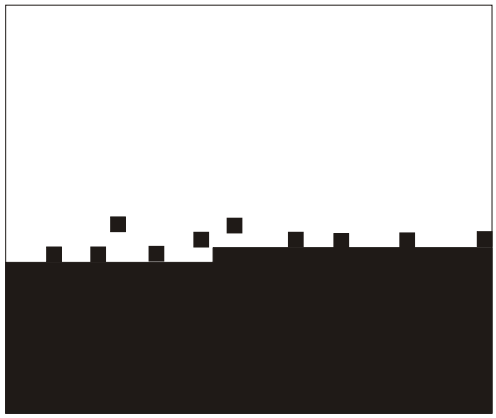

Abbildung 5.9: Registerinhalt nach Anwendung von C

Der zweite Teil von C hat eine Tiefe von $\alpha(1+\frac{1}{s})\log N$, so dass sich eine Gesamttiefe von

$$\alpha(1+\tfrac{1}{s})\log N + c_s \log t \log\log N$$

ergibt.

6. Fibonacci-Korrekturnetz

In diesem Teil wird aufgezeigt, wie man, wenn ein partielles Korrekturnetz vorliegt, ein Korrekturnetz nahezu der gleichen Tiefe konstruieren kann.

Zunächst soll gezeigt werden, wie man möglichst effektiv eine kleine Dirty Area „bereinigt“. Angenommen, es gibt ein t-Korrekturnetz X_N, dass für Eingabegrößen N eine Tiefe von $\delta(N, t)$ aufweist. Dann gibt es ein Vergleichernetzwerk, das jede t-gestörte Eingabe mit einer Größe der Dirty Area von höchstens Δ Registern sortiert. Dieses Netz hat eine Tiefe von $2\delta(2\Delta, t)$.

Dieses Netz hat Register mit Indizes (1), ..., (N). Es besteht aus zwei Teilnetzen $X_{2\Delta}$, die je eine Tiefe von $\delta(2\Delta, t)$ haben. Das erste Teilnetz „bearbeitet“ die Register S_{2i}, das zweite die Register S_{2i+1}, wobei i solche ganzzahligen Werte annimmt, dass sich gültige Indizes ergeben:

$$S_{2i} = \{r_{2i\Delta+1}, r_{2i\Delta+2}, ..., r_{2i\Delta+2\Delta}\}$$

$$S_{2i+1} = \{r_{(2i+1)\Delta+1}, r_{(2i+1)\Delta+2}, ..., r_{(2i+1)\Delta+2\Delta}\}$$

Dieses Cleaningnetz bereinigt die Dirty Area, da diese in mindestens einem S_i enthalten ist. Abbildung 6.1 zeigt den Aufbau des Netzes beispielhaft für N=12 und Δ=3.

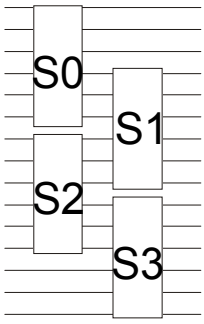

Abbildung 6.1: Aufbau des Cleaningnetzes

Nach der Beschreibung der vorangegangenen Korrekturnetze folgt nun die Hauptaussage dieses Abschnittes. Es wird gezeigt, dass es reicht, ein (t, Δ)-partielles-Korrekturnetz Y kleiner Tiefe und einem vernünftig kleinen Δ und ein t-Korrekturnetz X von nicht allzu großer Tiefe zu kombinieren, um ein t-Korrekturnetz mit ähnlicher Tiefe wie der von Y zu konstruieren. Diese Rückführung wird auch Refinement-Lemma genannt:

Angenommen, es gibt ein (t, Δ_1)-partielles-Korrekturnetz Y_N der Tiefe $\delta'(N, t)$ mit $\Delta_1=\Delta(N, t)$ und ein t-Korrekturnetz X_N der Tiefe $\delta(N, t)$. Für jedes natürlichzahlige M und jede

Eingabegröße N gibt es dann ein t-Korrekturnetz der Tiefe $\delta(M,t)+\delta'(\frac{Nt}{M},t)+2\delta(\frac{4M\Delta_2}{t}+2M,t)$ mit $\Delta_2=\Delta(Nt/M, t)$.

Der Beweis dieser Aussage wird konstruktiv erbracht. Seien die Registerindizes Paare (i,j) mit $i \in \{1, ..., N/M\}$ und $j \in \{1, ..., M\}$. Das Netz besteht aus drei Teilen.

Im ersten Teil wendet man N/M Kopien von X_M in jeder Zeile parallel an. Dies erfordert $\delta(M,t)$ Stufen. Das Ergebnis ist, dass die verschobenen Einsen in die letzen t Spalten wandern und die verschobenen Nullen in die ersten t Spalten wandern (Abbildung 6.2).

Abbildung 6.2: Ergebnis des ersten Teils

Im zweiten Teil werden zwei Kopien von $Y_{Nt/M}$ verwandt. Die erste Kopie wird genau gespiegelt, um mit den verschobenen Nullen umgehen zu können, und auf die ersten t Spalten angewandt. Die zweite Kopie wird auf die letzen t Spalten angewandt. Dieser Teil hat eine Tiefe von $\delta'(\frac{Nt}{M},t)$ Stufen, und die Größe der Dirty Area wird auf höchstens $M(2\frac{\Delta}{t}+1)=2\frac{M\Delta}{t}+M$ Register reduziert.

Der dritte Teil verwendet das Cleaningnetz auf Basis von X_N für die Dirty Area obiger Größe und hat somit eine Tiefe von $2\delta(\frac{4M\Delta_2}{t}+2M,t)$.

Die folgende Aussage zeigt, wie man das Refinement-Lemma zur Konstruktion von Korrekturnetzen kleiner Tiefen verwenden kann: Für jedes natürlichzahlige s gibt es ein t-Korrekturnetz der Tiefe $\alpha(1+\frac{1}{s})\log N+c_s(\log t \log\log N)^2$ für eine Konstante c_s in Abhängigkeit von s. Zum Beweis verwendet man das Refinement-Lemma unter Verwendung von $Y_N = C$ und des Batcher-Netzwerkes (einem bitonischen Netz aus [4]) als X_N sowie mit M = t log N. Mit $t=o(2^{\sqrt{\log N}/\log\log N})$ und $s=\log\log\log N$ erhält man ein t-Korrekturnetz der Tiefe

$$\alpha(1+\tfrac{1}{\log\log\log N})\log N + c\log^2 t\log\log^4 N \sim \alpha\log N\,.$$

Damit ist die angestrebte Konstruktion eines t-Korrekturnetzes einer Tiefe von ungefähr α log N abgeschlossen.

7. Abbildungsverzeichnis

8. Literatur

[1] G. Stachwiak, Fibonacci Correction Networks, SWATT 2000, SNCS 1851, S. 535-548, 2000

[2] I.N. Bronstein et al., Taschenbuch der Mathematik, Frankfurt am Main, 1997

[3] A.C. Yao, Bounds on Selection Networks, SIAM J. Comput. 9, S. 566-582, 1980

[4] K.E. Batcher, Sorting Networks an their Application, AFIPS Conf. Proc. 32, S. 307-314, 1968