Die Seidel-Sharir-Analyse für den Union-Find-Algorithmus

Lemma 1. *Bei einer Grundmenge mit n Elementen und Vereinigung nach Rang gibt es für jedes $r \geq 0$ höchstens $n/2^r$ Knoten, die irgendwann während des Algorithmus Rang r haben.*

Beweis. Wir machen vollständige Induktion nach r. Für $r = 0$ ist die Aussage offensichtlich. Damit ein Knoten vom Rang r entsteht, muss es vorher zwei Wurzeln vom Rang $r-1$ geben, die miteinander verbunden werden. Eine dieser Wurzeln wird aber dabei ein Kind der anderen und ändert danach ihren Rang nicht mehr. □

1 Werdegang eines Knotens bei Vereinigung nach Rang und Pfadkompression

Jeder Knoten v startet als Wurzel mit Rang 0 und ohne Kinder. Solange v Wurzel bleibt, kann der Rang von v durch UNIONs mit gleichrangigen Knoten schrittweise um 1 steigen. Während dieser Phase kann v auch zusätzliche Kinder bekommen, sowohl durch UNION-Operationen als auch durch die Pfadkompressionen bei FIND-Operationen.

Sobald v bei einer UNION-Operation als Kind an einen gleich- oder höherrangigen Knoten gehängt wird, ändert sich der Rang r von v nicht mehr. Ab dann hat v immer einen Elternknoten $p(v)$, und der Rang des Elternknotens $p(v)$ ist immer größer als r. Der Rang von $p(v)$ kann im Lauf der Zeit noch wachsen, entweder weil der Elternknoten durch UNION-Operationen schrittweise seinen Rang erhöht, oder weil v durch Pfadkompressionen einen neuen Elternknoten mit größerem Rang bekommt.

2 Elimination der UNION-Operationen

Sei F ein Wald. Ein *Pfad* P in F ist eine Folge v_1, v_2, …, v_k von $k \geq 1$ Knoten, sodass $v_{i+1} = p(v_i)$ ist, für $i = 1, \ldots, k-1$.

Eine *Kompression* in F ist gegeben durch einen Pfad in F. Wir ersetzen die Elternzeiger aller Knoten v_1, v_2, …, v_{k-2} durch v_k. Im Gegensatz zur Pfadkompression, wie sie beim UNION-FIND-Algorithmus ausgeführt wird, muss v_k keine Wurzel sein. Die *Kosten* der Kompression definieren wir als die Anzahl $\max\{k-2, 0\}$ der Zeigeränderungen.

Eine *Folge von Kompressionen* $C = (P_1, P_2, \ldots, P_m)$ für einen Wald F ist wie folgt definiert. Für $i = 0, \ldots m$ bezeichnen wir mit F_i den Wald nach Ausführen der ersten i Kompressionen. Es ist also $F_0 = F$. Jeder Pfad P_i muss ein Pfad im Wald F_{i-1} sein, und wenn man die Kompression P_i in F_{i-1} ausführt, erhält man F_i.

Die *Kosten* $\text{cost}(C)$ der Folge C sind definiert als die Summe der Kosten der P_i.

Das folgende Lemma nützt aus, dass wir den Endknoten einer Kompression beliebig angeben können. Dadurch wird es möglich, die UNION-Operationen vor die FINDs zu schieben. Das hat bei der Analyse den Vorteil, dass jeder Knoten zu Beginn der FIND-Operationen einen festen Rang hat, der sich nicht mehr ändert.

Lemma 2. *Gegeben sei eine Folge X von UNION-FIND Operationen, die m FINDs enthält. Wir beginnen mit der Partition $\{\{1\}, \{2\}, \ldots, \{n\}\}$ und verwenden Vereinigung nach Rang mit Pfadkompression. Dann existiert ein Wald F und eine Folge C von m Kompressionen für F, sodass folgendes gilt:*

(i) *Die Gesamtlaufzeit von X ist $O(m + n + \text{cost}(C))$.*

Außerdem kann man jedem Knoten im Wald F einen festen Rang $\in \{0, \ldots, \lfloor \log_2 n \rfloor\}$ zuordnen mit folgenden Eigenschaften:

(ii) *Der Rang jedes Knotens ist während der gesamten Laufzeit kleiner als der seines jeweiligen Elternknotens, sobald ein solcher existiert, und wenn der Knoten im Laufe des Algorithmus einen neuen Elternknoten bekommt, so muss dieser einen größeren Rang haben als der bisherige Elternknoten.*

(iii) *Es gibt höchstens $n/2^r$ Knoten mit Rang $\geq r$.*

Beweis. Führe zunächst nur die UNION-Operationen in X durch. Sei F der resultierende Wald. Den Rang einen Knotens definieren wir als die Höhe in F; das ist der endgültige Rang am Ende des UNION-FIND-Algorithmus. Eigenschaft (ii) folgt aus den Ausführungen in Abschnitt 1; man muss sich nur überlegen, dass der Wechsel auf endgültige Ränge nichts an diesen Eigenschaften ändert. Eigenschaft (iii) folgt aus Lemma 1.

Nun konstruieren wir die Folge C. Dazu betrachten wir jede Operation FIND(u) in X. Wenn v das Ergebnis von FIND(u) ist, fügen wir zu C den Pfad von u nach v im aktuellen Wald hinzu. Offenbar ist C eine gültige Folge von m Kompressionen für F.

Nun zur Laufzeit: X enthält höchstens $n-1$ UNIONs, von denen jedes konstante Zeit braucht. Für jedes der m FINDs benötigen wir jeweils konstante Laufzeit plus die Zeit zum Ablaufen des Pfades. Letzteres ist durch $\text{cost}(C)$ abgedeckt. Somit folgt (i). □

3 Teile und herrsche: Aufspalten nach Rängen

Wir werden allgemein einen Wald F mit seiner Knotenmenge identifizieren, und wir bezeichnen die Anzahl der Knoten mit $|F|$.

Definition. *Bei einer* Zerlegung *eines Waldes F wird der Wald für eine vorgegebene Schranke s in F_- und F_+ aufgespalten, wobei der* untere Wald F_- *die Knoten mit Rang höchstens s enthält und der* obere Wald F_+ *die Knoten mit Rang größer als s. Für einen Knoten v von F_-, dessen Elternknoten $p(v)$ ursprünglich in F_+ liegt, wird $p(v) := \perp$ gesetzt. Das heißt, v wird zu einer Wurzel im unteren Wald F_- gemacht.*

Was machen wir mit einer Kompression P, die die Grenze zwischen F_- und F_+ überschreitet? Wir müssen sie in zwei Teile P_- und P_+ trennen. Der neue Elternzeiger für die Knoten in P_- müssen wir beim Komprimieren ersatzweise auf $\perp$ setzen. Wir erweitern also den Begriff einer Kompression:

Ein *abgeschnittener Pfad* ist, ähnlich wie ein "normaler" Pfad eine Folge $v_1, v_2, \ldots, v_k$ von Knoten, sodass v_{i+1} der Elternknoten von v_i ist, für $i = 1, \ldots, k-1$, mit der zusätzliche Eigenschaft, dass v_{k-1} eine Wurzel in F ist und daher den Elternzeiger $v_k = \perp$ hat. Ein abgeschnittener Pfad muss mindestens einen echten Knoten enthalten, das heißt, es muss $k \geq 2$ sein. Bei der Kompression eines abgeschnittenen Pfades ersetzen wir die Elternzeiger aller Knoten $v_1, v_2, \ldots, v_{k-2}$ folgerichtig durch $v_k = \perp$.

Die Kompressionsfolgen C, die wir betrachten, werden auch abgeschnittene Pfade enthalten. Die *Länge* $\ell(C)$ von C definieren wir aber als die *Anzahl der nicht abgeschnittenen Pfade* in C. Die Kosten einer Kompression mit einem abgeschnittenen Pfad sind als 0 definiert. Die abgeschnittenen Pfade spielen also weder bei den Kosten noch bei der Länge eine Rolle. (Da in der Reduktion von Lemma 2 keine abgeschnittenen Pfade verwendet werden, bleibt Lemma 2(i) mit dieser erweiterten Definition von $\text{cost}(C)$ gültig.)

Die Kosten kann man auch so charakterisieren: Die Kosten $\text{cost}(C)$ einer Kompressionsfolge C zählen, wie oft ein Knoten v einen neuen Elternzeiger $p(v) \neq \perp$ bekommt.

Jetzt kommt das entscheidende Zerlegungslemma.

Lemma 3. *Sei F ein Wald und C eine Folge von Kompressionen für F. Sei F_+, F_- eine Zerlegung von F. Dann existieren Kompressionsfolgen C_+ für F_+ und C_- für F_-, sodass gilt:*

(i) $\ell(C) = \ell(C_+) + \ell(C_-)$

(ii) $\text{cost}(C) \leq \text{cost}(C_+) + \text{cost}(C_-) + |F_-| + \ell(C_+)$

Beweis. Wir gehen die Kompressionen in $P \in C$ einzeln durch. Falls $P \subseteq F_- \cup \{\bot\}$ ist, fügen wir P zu C_- hinzu, und falls $P \subseteq F_+ \cup \{\bot\}$ ist, fügen wir P zu C_+ hinzu. Es bleiben die Pfade, die sowohl Knoten in F_- als auch in F_+ enthalten. Für solche Pfade P fügen wir $P_+ = P \cap (F_+ \cup \{\bot\})$ zu C_+ hinzu, und wir fügen $P_- = (P \cap F_-) \cup \{\bot\}$ zu C_- hinzu. Der letzte Knoten von $P \cap F_-$ muss ja eine Wurzel in F_- sein, und P_- wird durch Anhängen des $\bot$-Knotens zu einem abgeschnittenen Pfad gemacht. Da dieser abgeschnittener Pfad für $\ell(C_-)$ nicht zählt, bleibt die Gesamtlänge der Folge in Summe gleich, und es gilt (i).

Die Folgen C_+ und C_- modellieren C nach. Jeder Zeiger von F_- zu einem Knoten außerhalb von F_- wird zu einem $\bot$-Zeiger, und die Kompression der entstehenden abgeschnittenen Pfade tut genau das Richtige.

Nun beweisen wir (ii). Die Kosten $\text{cost}(C)$ zählen die folgenden Ereignisse, die bei einer Kompression passieren können: (Sie sind unten tabellarisch zusammengefasst.) (a) ein Knoten aus F_+ bekommt einen neuen Elternknoten in F_+; (b) ein Knoten aus F_- bekommt einen neuen Elternknoten in F_-; (c) ein Knoten in F_-, dessen Elternknoten in F_- liegt, bekommt einen neuen Elternknoten in F_+; (d) ein Knoten in F_-, dessen Elternknoten in F_+ liegt, bekommt einen neuen Elternknoten in F_+. Wegen Lemma 2(ii) sind das die einzigen Möglichkeiten. Das Ereignis (c) kann einem Knoten aus F_- nur einmal zustoßen; es wird daher von dem Term $|F_-|$ abgedeckt. Die Ereignisse (a) werden auf der rechten Seite durch $\text{cost}(C_+)$ gezählt, die Ereignisse (b) durch $\text{cost}(C_-)$. Das Ereignis (d) kann nur bei einem nicht abgeschnittenen Pfad $P \in C$ auftreten, der in zwei Teile P_- und P_+ geteilt wird, und dann gibt es höchstens einen betroffenen Knoten, nämlich den letzten Knoten in F_-. Dieses Ereignis tritt also höchstens $\ell(C_+)$-mal auf. □

		ursprünglich		nach der Trennung		ursprüngliche Kosten
	v	$p(v)$ davor	$p(v)$ danach	$p(v)$ davor	$p(v)$ danach	sind abgedeckt durch
(a)	F_+	F_+	F_+	F_+	F_+	$\text{cost}(C_+)$
(b)	F_-	F_-	F_-	F_-	F_-	$\text{cost}(C_-)$
(c)	F_-	F_-	F_+	F_-	$\bot$	$\|F_-\|$
(d)	F_-	F_+	F_+	$\bot$	$\bot$	$\ell(C_+)$
	F	$\bot$	$\bot$	$\bot$	$\bot$	(kostet nichts)

4 Immer bessere Schranken

4.1 Stufe 0

Beobachtung 1. *Sei F ein Wald mit n Knoten, in dem jeder Knoten höchstens Rang r hat. Dann gilt für jede Kompressionsfolge C für F, dass $\text{cost}(C) \leq r \cdot n$ ist.*

Beweis. Jedesmal, wenn ein Knoten v bei einer Kompression einen neuen Elternknoten $\neq \bot$ erhält, hat dieser einen höheren Rang als der alte Elternknoten von v. Das kann nur r-mal pro Knoten passieren. □

4.2 Stufe 1

Setze $s := \lceil \log_2 r \rceil$ und betrachte die Zerlegung von F in $F_- = F_{\leq s}$ und $F_+ = F_{>s}$. Lemma 3(ii) ergibt

$$\text{cost}(C) \leq \text{cost}(C_-) + \text{cost}(C_+) + |F_-| + \ell(C_+).$$

Nun wollen wir die Größen in dieser Ungleichung einsetzen. Zunächst gilt wegen Lemma 2(iii) für die Anzahl der Knoten im oberen Wald $|F_+| \leq n/2^{s+1} \leq n/r$. Also folgt nach Beobachtung 1, dass $\text{cost}(C_+) \leq r \cdot |F_+| \leq n$ ist. Außerdem ist $|F_-| \leq n$, da es nur n Knoten gibt, und $\ell(C_+) = \ell(C) - \ell(C_-)$ wegen Lemma 3(i). Somit erhalten wir

$$\begin{aligned}\text{cost}(C) - \ell(C) &\leq \text{cost}(C_-) + \text{cost}(C_+) + |F_-| - \ell(C_-) \\ &\leq \text{cost}(C_-) - \ell(C_-) + 2n.\end{aligned}$$

Nun ist aber F_- ein Wald mit Rang höchstens $s = \lceil \log r \rceil$. Daher können wir das gleiche Spiel mit F_- und $s' = \lceil \log s \rceil = \lceil \log \log r \rceil$ wiederholen, und wir erhalten

$$\text{cost}(C_-) - \ell(C_-) \leq \text{cost}(C_{\leq s'}) - \ell(C_{\leq s'}) + 2n,$$

also

$$\text{cost}(C) - \ell(C) \leq \text{cost}(C_{\leq s'}) - \ell(C_{\leq s'}) + 4n.$$

Das geht nun weiter bis wir bei einem Wald F_0 angekommen sind, in dem jeder Rang höchstens 1 ist. Dies ist nach $\log^* r$ Schritten der Fall. In einem solchen Wald gilt für jede Folge C_0 die Ungleichung $\text{cost}(C_0) - \ell(C_0) \leq 0$, weil dort jede Kompression Kosten 0 hat. Bei jedem Schritt kommt ein Term $2n$ dazu, und somit gilt

$$\text{cost}(C) - \ell(C) \leq 2n \log^* r. \tag{1}$$

4.3 Der allgemeine Schritt

Dies ist aber noch lange nicht alles. Mit der neuen Schranke können wir jetzt wieder von vorne anfangen, und erhalten eine noch bessere Schranke. Dazu definieren wir rekursiv eine Funktion $L_j(r)$ wie folgt: $L_0(r) = \lceil \log_2 r \rceil$, für alle $r \geq 1$. Angenommen, wir haben L_j definiert. Dann ist

$$L_{j+1}(r) = \begin{cases} 0, & \text{falls } r \leq 1 \text{ ist,} \\ 1 + L_{j+1}(L_j(r)), & \text{sonst.} \end{cases}$$

In Worten:

> $L_{j+1}(r)$ gibt an, wie oft man die Funktion L_j auf r anwenden muss, bis man eine Zahl kleiner oder gleich 1 erhält.

Für $j = 1$ haben wir definitionsgemäß $L_1(r) = \log^* r$.

Lemma 4. *Für jedes $j \geq 1$, für jeden Wald mit Rang höchstens r und für jede Kompressionsfolge C gilt*

$$\text{cost}(C) \leq j\ell(C) + 2nL_j(r). \tag{2}$$

Dieses Lemma lässt sich so interpretieren: Wenn wir für jede Kompression in C einen Kostenrahmen von j vorsehen, dann bleibt noch ein nicht abgedeckter Überschuss von höchstens $2nL_j(r)$.

Beweis. Wir wollen das Lemma durch vollständige Induktion nach j zeigen. Nach (1) gilt die Aussage (2) für $j = 1$. Für den Induktionsschritt nehmen wir an, dass die Induktionsannahme (2) für ein $j \geq 1$ gilt, und wir wollen zeigen, dass auch gilt:

$$\text{cost}(C) \leq (j+1)\ell(C) + 2nL_{j+1}(r). \tag{3}$$

Sei F ein Wald mit Rang höchstens r und C eine Kompressionsfolge. Setze $s = \lceil \log L_j(r) \rceil$, und betrachte die Zerlegung von F in die Knoten F_- mit Rang höchstens s und die Knoten F_+ mit Rang mindestens $s+1$. Lemma 3(ii) ergibt

$$\text{cost}(C) \le \text{cost}(C_-) + \text{cost}(C_+) + |F_-| + \ell(C_+).$$

Nach Induktionsannahme ist

$$\text{cost}(C_+) \le j\ell(C_+)+2\cdot|F_+|\cdot L_j(r) \le j\ell(C_+)+2\cdot\frac{n}{2^{s+1}}\cdot L_j(r) \le j\ell(C_+)+2\cdot\frac{n}{2L_j(r)}\cdot L_j(r) \le j\ell(C_+)+n.$$

Weiter ist $|F_-| \le n$. Also folgt

$$\text{cost}(C) \le \text{cost}(C_-) + (j+1)\ell(C_+) + 2n.$$

Nun benutzen wir noch Lemma 3(i) und erhalten

$$\text{cost}(C) - (j+1)\ell(C) \le \text{cost}(C_-) - (j+1)\ell(C_-) + 2n.$$

Nun wiederholen wir das ganze mit F_-, bis wir schließlich wieder bei Rang höchstens 1 angelangt sind. Da wir jedesmal die Funktion $r \mapsto s = \lceil \log_2 L_j(r) \rceil \le L_j(r)$ auf den Rang anwenden, ist dies nach höchstens $L_{j+1}(r)$ Schritten der Fall. Jedesmal kommt ein Term $2n$ hinzu, woraus (3) folgt, und das Lemma ist bewiesen. □

4.4 Wie oft soll man iterieren?

Nach Lemma 2 gilt für jeden Wald F mit n Knoten, jede Kompressionsfolge C der Länge m, und jedes $j \ge 1$:

$$\text{cost}(C) \le jm + 2nL_j(\lfloor \log_2 n \rfloor).$$

Hierbei haben wir $\ell(C) = m$ eingesetzt und $r \le \log_2 n$ abgeschätzt. Wie sollen wir nun j wählen? Am besten sollen die beiden Summanden ausgeglichen werden. Der erste Summand wächst mit j, der zweite Summand fällt extrem stark mit j. Der bester Parameter j wird also sehr klein sein, und wir machen keinen großen Fehler, wenn wir den Faktor j im ersten Summanden ignorieren.

$$m \approx 2nL_j(\lfloor \log_2 n \rfloor) \Leftrightarrow L_j(\lfloor \log_2 n \rfloor) \approx m/n.$$

Wir definieren also:

$$\alpha(m,n) := \min\{\, j \ge 1 \mid L_j(\lfloor \log_2 n \rfloor) \le 3 + m/n \,\}.$$

Der additive Term 3 wurde gewählt, weil man $L_j(x)$ für $x \ge 5$ durch Erhöhen von j nicht unter 3 drücken kann. Die Funktion $\alpha(n,m)$ heißt die *inverse Ackermannfunktion.* Wenn wir $j = \alpha(m,n)$ wählen, ergibt sich

$$\text{cost}(C) \le m(\alpha(m,n)+2) + 6n.$$

Wenn wir alles zusammensetzen, erhalten wir:

Satz 1. *Gegeben sei eine Folge von UNION-FIND-Operationen, die auf die Ausgangspartition* $\{\{1\},\{2\},\ldots,\{n\}\}$ *angewendet wird und m FIND-Operationen enthält. Wenn wir Vereinigung nach Rang mit Pfadkompression verwenden, dann ist die Laufzeit* $O(n + m\alpha(m,n))$.

4.5 Wachstum von L_j und die Umkehrfunktionen B_j

$$L_j(r) \le k \iff r \le B_j(k)$$

$B_0(k) = 2^k$; $B_{j+1}(0) = 1$; $B_{j+1}(k) = B_j(B_{j+1}(k-1)) = \underbrace{B_j(B_j(B_j(\cdots(B_j}_{k\text{-mal}}(1))\cdots)))$, für $k > 0$

	$k=0$	1	2	3	4	5	…	k
$B_0(k)$	1	2	4	8	16	32		2^k
$B_1(k)$	1	2	4	$16 = 2^{2^2} = 2\uparrow 3$	$2^{16} = 2^{2^{2^2}} = 2\uparrow 4$	$2\uparrow 5$		$2\uparrow k$
$B_2(k)$	1	2	4	$2^{16} = 2\uparrow 4$	$2\uparrow(2^{16})$			
$B_3(k)$	1	2	4	$2\uparrow(2^{16})$				
$B_4(k)$	1	2	4					