Satz 74

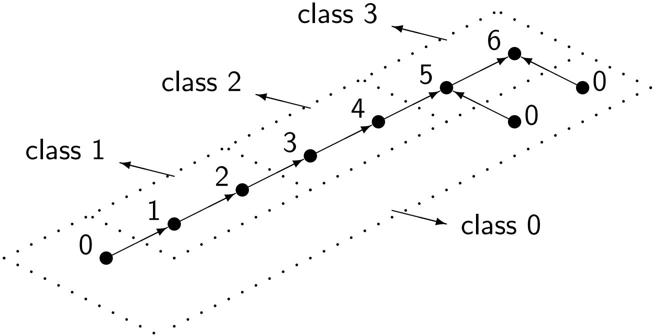
Bei der obigen Implementierung ergibt sich eine amortisierte Komplexität von $\mathcal{O}(\log^* n)$ pro Operation.

Beweis:

Sei T' der (endgültige) In-Baum, der durch die Folge der *Union*'s, ohne die *Find*'s, entstehen würde (also keine Pfad-Kompression). Ordne jedem Element x drei Werte zu:

- rank(x):= Höhe des Unterbaums in T' mit Wurzel x
- class(x):= $\begin{cases} i \geq 1 & \text{falls } a_{i-1} < \text{ rank}(x) \leq a_i \text{ ist } (i \geq 1) \\ 0 & \text{falls rank}(x) = 0 \end{cases}$

  Dabei gilt: $a_0 = 0, a_i = 2^{2^{\cdot^{\cdot^{2}}}} \Big\}\, i\ 2'\text{en}$ für $i \geq 1$.
  Setze zusätzlich $a_{-1} := -1$.


class 3
class 2
class 1
class 0
6
5
4
3
2
1
0
0
0

Beweis (Forts.):

- dist(x) ist die Distanz von x zu einem Vorfahr y im momentanen Union/Find-Baum (mit Pfad-Kompression), so dass class(y) $>$ class(x) bzw. y die Wurzel des Baumes ist.

Definiere die Potenzialfunktion

$$\text{Potenzial} := c \sum_x \text{dist}(x), \quad c \text{ eine geeignete Konstante } > 0$$

Beweis (Forts.):

Beobachtungen:

i) Sei T ein Baum in der aktuellen Union/Find-Struktur (mit Pfad-Kompression), seien x, y Knoten in T, y Vater von x. Dann ist class(x) $\leq$ class(y).

ii) Aufeinander folgende *Find(x)* durchlaufen (bis auf eine) verschiedene Kanten. Diese Kanten sind (im wesentlichen) eine Teilfolge der Kanten in T' auf dem Pfad von x zur Wurzel.

Beweis (Forts.):

Amortisierte Kosten Find(x):

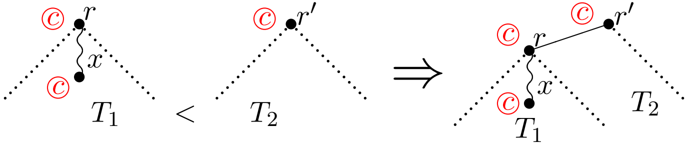
Sei $x_0 \to x_1 \to x_2 \ldots x_k = r$ der Pfad von x_0 zur Wurzel. Es gibt höchstens $\log^* n$-Kanten (x_{i-1}, x_i) mit class(x_{i-1}) $<$ class(x_i). Ist class(x_{i-1}) $=$ class(x_i) und $i < k$ (also $x_i \neq r$), dann ist dist(x_{i-1}) vor der *Find(x)*-Operation ≥ 2, nachher gleich 1.

Damit können die Kosten für alle Kanten (x_{i-1}, x_i) mit class(x_{i-1}) $=$ class(x_i) aus der Potenzialverringerung bezahlt werden. Es ergeben sich damit amortisierte Kosten

$$\boxed{\mathcal{O}(\log^* n)}$$

Beweis (Forts.):

Amortisierte Gesamtkosten aller $(n-1)$-Union's:



Die gesamte Potenzialerhöhung durch alle *Union*'s ist nach oben durch das Potenzial von T' beschränkt (Beobachtung ii).

Beweis (Forts.):

$$\begin{aligned}
\text{Potenzial}(T') &\le c \cdot \sum_{i=0}^{\log^* n} \sum_{\text{rank}(x)=j=a_{i-1}+1}^{a_i} \text{dist}(x) \\
&\le c \cdot \sum_{i=0}^{\log^* n} \sum_{\text{rank}(x)=j=a_{i-1}+1}^{a_i} \frac{n}{2^j} a_i \\
&\le c \cdot n \sum_{i=0}^{\log^* n} a_i \frac{1}{2^{a_{i-1}}} = c \cdot n \sum_{i=0}^{\log^* n} 1 \\
&= \mathcal{O}(n \log^* n)\,.
\end{aligned}$$

Die zweite Ungleichung ergibt sich, da alle Unterbäume, deren Wurzel x $\text{rank}(x) = j$ hat, disjunkt sind und jeweils $\ge 2^j$ Knoten enthalten. □

8.2.4 Erweiterungen

1) Bessere obere Schranke $\alpha(k,n), k \geq n$. Betrachte die (Variante der) Ackermannfunktion $A(m,n)$ mit:

$$
\begin{aligned}
A(0,n) &= 2n; & n \geq 0\\
A(m,0) &= 2; & m \geq 1\\
A(m+1,n+1) &= A(m, A(m+1,n))
\end{aligned}
$$

$m \downarrow$ / $n \rightarrow$					
	0	2	4	6	8
	2	4	8	16	32
	2	8	2^9		
	2				
	2				
	$\vdots$				

Die Ackermannfunktion $A(\cdot,\cdot)$ steigt asymptotisch schneller als jede primitiv-rekursive Funktion.

Definition 75

Die Klasse der primitiv-rekursiven Funktionen (auf den natürlichen Zahlen) ist induktiv wie folgt definiert:

1. Alle konstanten Funktionen sind primitiv-rekursiv.
2. Alle Projektionen sind primitiv-rekursiv.
3. Die Nachfolgerfunktion auf den natürlichen Zahlen ist primitiv-rekursiv.
4. Jede Funktion, die durch Komposition von primitiv-rekursiven Funktionen entsteht, ist primitiv-rekursiv.
5. Jede Funktion, die durch sog. primitive Rekursion aus primitiv-rekursiven Funktionen entsteht, ist primitiv-rekursiv. Primitive Rekursion bedeutet folgendes Schema für die Definition von f:

$$f(0, \cdots) = g(\cdots)$$
$$f(n+1, \cdots) = h(f(n, \cdots), \cdots)$$

wobei g, h bereits primitiv-rekursive Funktionen sind.

Weiter wird gesetzt:

$$\alpha(k,n) := \min\{i \geq 1; A(i, \left\lfloor \frac{k}{n} \right\rfloor) > \log n\}$$

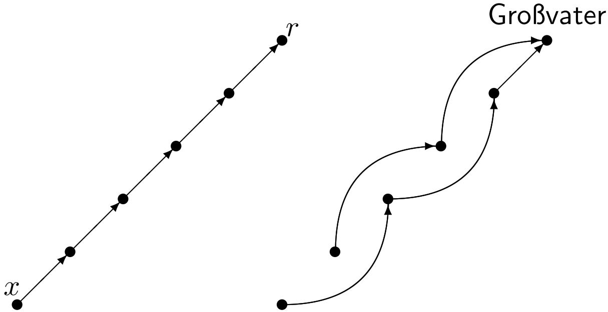
Dann gilt: Der Zeitbedarf für eine Folge von k *Find*- und *Union*-Operationen auf einer Menge mit n Elementen, darunter $n-1$ *Union*, ist

$$\mathcal{O}(k\alpha(k,n))\,.$$

Es gilt auch eine entsprechende untere Schranke.

Robert E. Tarjan:
Data Structures and Network Algorithms
SIAM CBMS-NSF Regional Conference Series in Applied Mathematics Bd. 44 (1983)

2) Variante der Pfadkompression:



Diese Variante der Pfadhalbierung erfüllt ebenfalls die $\mathcal{O}(k\alpha(k,n))$ Schranke.

Kapitel III Selektieren und Sortieren

1. Einleitung

Gegeben: Menge S von n Elementen aus einem total geordneten Universum U, $i \in \mathbb{N}$, $1 \leq i \leq n$.
Gesucht: i-kleinstes Element in S.

Die Fälle $i = 1$ bzw. $i = n$ entsprechen der Suche nach dem Minimum bzw. Maximum.
Der Standardalgorithmus dafür benötigt $n - 1$ Vergleiche.

Satz 76

Die Bestimmung des Minimums/Maximums von n Elementen benötigt mindestens $n-1$ Vergleiche.

Beweis:

Interpretiere Algorithmus als Turnier. Ein Spiel wird jeweils vom kleineren Element gewonnen. Wir beobachten: Jedes Element außer dem Gesamtsieger muss mindestens ein Spiel verloren haben $\Rightarrow n-1$ Vergleiche notwendig. $\square$

Bestimmung des Vize-Meisters bzw. des zweitkleinsten Elements

Satz 77

Das zweitkleinste von n Elementen kann mit

$$n + \lceil \log_2 n \rceil - 2$$

Vergleichen bestimmt werden.

Beweis:

Wir betrachten wiederum ein KO-Turnier: $(n-1)$ Vergleiche genügen zur Bestimmung des Siegers (Minimum).
Das zweitkleinste Element ist unter den „Verlierern“ gegen das Minimum zu suchen. Deren Anzahl ist $\leq \lceil \log_2 n \rceil$. Man bestimme nun unter diesen Elementen wiederum das Minimum und erhält damit das zweitkleinste Element in $\leq \lceil \log_2 n \rceil - 1$ weiteren Vergleichen. $\square$

Lewis Carroll:
Lawn Tennis Tournaments
St. Jones Gazette (Aug. 1, 1883), pp. 5–6
Reprinted in The Complete Work of Lewis Carroll. Modern Library, New York (1947)

Vaughan R. Pratt, Frances F. Yao:
On lower bounds for computing the i-th largest element
Proc. 14th Ann. IEEE SWAT, pp. 70–81 (1973)

Donald E. Knuth:
The art of computer programming. Vol. 3: Sorting and searching,
3. Auflage, Addison-Wesley Publishing Company: Reading (MA), 1997

2. Der Blum-Floyd-Pratt-Rivest-Tarjan Selektions-Algorithmus

Definition 78

Sei $n \in \mathbb{N}$. Der Median (das „mittlere" Element) einer total geordneten Menge von n Elementen ist deren i-kleinstes Element, wobei

$$i = \left\lceil \frac{n}{2} \right\rceil .$$

Bemerkung: Für gerade n wird manchmal auch $i = \left\lfloor \frac{n}{2} \right\rfloor + 1$ benutzt.

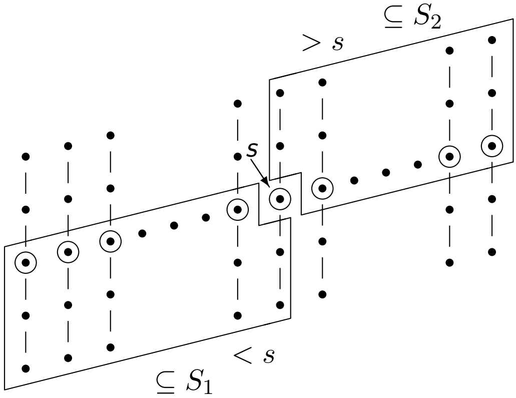
Sei m eine kleine ungerade Zahl (etwa $5 \leq m \leq 21$). Sei $S := \{a_1, \ldots, a_n\}$ eine Menge von n paarweise verschiedenen Elementen. Zur Bestimmung des i-kleinsten Element in S betrachten wir folgenden Algorithmus BFPRT.

Der BFPRT-Selektions-Algorithmus (1/3)

1. Teile S in $\lceil \frac{n}{m} \rceil$ Blöcke auf, $\lfloor \frac{n}{m} \rfloor$ davon mit je m Elementen
2. Sortiere jeden dieser Blöcke
3. Sei S' die Menge der $\lceil \frac{n}{m} \rceil$ Mediane der Blöcke. Bestimme rekursiv den Median s dieser Mediane (also das $\left\lceil \frac{|S'|}{2} \right\rceil$-kleinste Element von S').

Der BFPRT-Selektions-Algorithmus (2/3)

4. Partitioniere $S - \{s\}$ in $S_1 := \{x \in S : x < s\}$, $S_2 := \{x \in S : x > s\}$. Bemerkung: $|S_1|,\ |S_2| \geq \frac{n}{4}$, falls $n \geq 3m - 1$.



Der BFPRT-Selektions-Algorithmus (3/3)

5. Falls $i \leq |S_1|$, bestimme rekursiv das i-kleinste Element in S_1.
   Falls $i = |S_1| + 1$, gib s als Lösung zurück.
   Ansonsten bestimme rekursiv das $(i - |S_1| - 1)$-kleinste Element in S_2.

Sei $T(n)$ die worst-case Anzahl von Vergleichen für $|S| = n$ des Algorithmus BFPRT. Sei C_m die # von Vergleichen, um m Elemente zu sortieren (z.B. $C_5 = 7$, $C_{11} = 26$). Es gilt:

$$T(n) \leq \underbrace{T\left(\left\lceil \frac{n}{m} \right\rceil\right)}_{3.} + \underbrace{T\left(\left\lfloor \frac{3}{4}n \right\rfloor\right)}_{5.} + \underbrace{\left\lceil \frac{n}{m} \right\rceil}_{2.} C_m + \underbrace{\left\lfloor \frac{n}{2} \right\rfloor}_{4.}$$

Satz 79

Der Selektions-Algorithmus BFPRT bestimmt das i-kleinste Element von n Elementen mit $\mathcal{O}(n)$ Vergleichen (und Zeit).

Beweis:

Annahme: $T(n) \leq c \cdot n$, wobei $c = c(m)$ konstant ist.

Die Annahme ist ok, falls $T(n) \leq$

$$T\left(\left\lceil \frac{n}{m} \right\rceil\right) + T\left(\left\lfloor \frac{3}{4}n \right\rfloor\right) + \left\lceil \frac{n}{m} \right\rceil C_m + \left\lfloor \frac{n}{2} \right\rfloor \leq cn; \text{ dies gilt, falls}$$

$$\left\lceil \frac{n}{m} \right\rceil c + \left\lfloor \frac{3}{4}n \right\rfloor c + \left\lceil \frac{n}{m} \right\rceil C_m + \left\lfloor \frac{n}{2} \right\rfloor \leq cn \qquad \Big| \cdot \frac{1}{n} \quad \text{(IA)}$$

$$\Leftrightarrow \text{(bis auf } \lceil\rceil, \lfloor\rfloor \text{)} \; \frac{c}{m} + \frac{3}{4}c + \frac{C_m}{m} + \frac{1}{2} \leq c$$

$$\Leftrightarrow -\frac{c}{m} - \frac{3}{4}c + c \geq \frac{C_m}{m} + \frac{1}{2}$$

$$\Leftrightarrow c \geq \frac{\frac{C_m}{m} + \frac{1}{2}}{1 - \frac{3}{4} - \frac{1}{m}}$$

$\square$

Bemerkung: $m = 11 \rightsquigarrow c = c(m) \approx 20$.

Literatur:

Vaughan R. Pratt, Frances F. Yao:
On lower bounds for computing the i-th largest element
Proc. 14th Ann. IEEE SWAT, pp. 70–81 (1973)

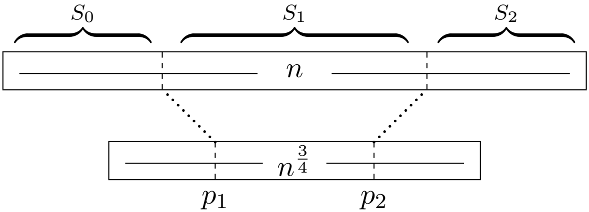
Manuel Blum, Robert W. Floyd, Vaughan R. Pratt, Ron L. Rivest, Robert E. Tarjan:
Time bounds for selection
J. Comput. Syst. Sci. **7**, pp. 448–461 (1973)

3. Randomisierter Median-Algorithmus

Problemstellung: Bestimme den Median von n Elementen

1. Wähle $n^{\frac{3}{4}}$ Elemente zufällig und gleichverteilt aus den n Elementen aus.
2. Sortiere diese $n^{\frac{3}{4}}$ Elemente mit einem (Standard-) $n \log n$-Algorithmus.
3. Setze

$$p_1 := \max\{\tfrac{n^{\frac{3}{4}}}{2} - \sqrt{n}, 1\}\text{-kleinstes Element der } n^{\frac{3}{4}} \text{ Elemente.}$$
$$p_2 := \min\{\tfrac{n^{\frac{3}{4}}}{2} + \sqrt{n}, n^{\frac{3}{4}}\}\text{-kleinstes Element der } n^{\frac{3}{4}} \text{ Elemente.}$$



4. Partitioniere die n Elemente in

   $S_0 := \{\text{Elemente} < p_1\}$
   $S_1 := \{p_1 \leq \text{Elemente} \leq p_2\}$
   $S_2 := \{p_2 < \text{Elemente}\}$

5. Falls $|S_0| \geq \lceil \frac{n}{2} \rceil$ oder $|S_2| \geq \lceil \frac{n}{2} \rceil$ oder $|S_1| \geq 4 \cdot n^{\frac{3}{4}}$, dann wiederhole den Algorithmus;
   ansonsten sortiere S_1 und liefere das $(\lceil \frac{n}{2} \rceil - |S_0|)$-kleinste Element davon ab.

Satz 80

Obiger randomisierter Algorithmus bestimmt den Median von n-Elementen mit einer erwarteten Anzahl von $\frac{3}{2}n + o(n)$ Vergleichen.

Beweis:

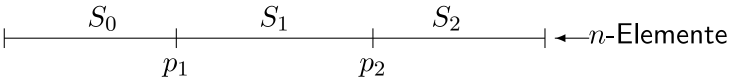
i) Korrektheit: klar.

Beweis (Forts.):

ii) Anzahl der Vergleiche in einer Iteration:

$$\mathcal{O}(n^{\frac{3}{4}} \log n^{\frac{3}{4}}) + \text{Kosten der Partitionierung}$$

Für die Partitionierung ist der naive Ansatz zu ungünstig, stattdessen:



Wähle zuerst jeweils mit Wahrscheinlichkeit $\frac{1}{2}$ aus, ob Element x mit p_1 oder p_2 verglichen wird, mache zweiten Vergleich nur, falls nötig.

Beweis (Forts.):

Die erwartete Anzahl von Vergleichen ist dann

$$
\begin{aligned}
&= \frac{n}{2}\left(\frac{|S_0|}{n}\cdot 1 + \frac{|S_1|+|S_2|}{n}\cdot 2\right) + \frac{n}{2}\left(\frac{|S_2|}{n}\cdot 1 + \frac{|S_0|+|S_1|}{n}\cdot 2\right)\\
&= \frac{n}{2}\left(\frac{|S_0|+|S_2|}{n} + 2\frac{\overbrace{|S_0|+|S_1|+|S_2|}^{n}+|S_1|}{n}\right)\\
&= \frac{n}{2}\left(3+\frac{|S_1|}{n}\right) = \frac{3}{2}n + o(n)
\end{aligned}
$$

Wir zeigen nun, dass der Algorithmus mit Wahrscheinlichkeit $\geq 1 - \mathcal{O}(n^{-\frac{1}{4}})$ nur eine Iteration benötigt (daraus folgt dann, dass insgesamt die Anzahl der Vergleiche $\leq \frac{3}{2}n + o(n)$ ist).

Beweis (Forts.):

Dafür verwenden wir Hilfsmittel aus der Wahrscheinlichkeitstheorie/Stochastik:

- **Bernoulli-Zufallsvariable** (ZV): X, Werte $\in \{0,1\}$ mit

$$X = \begin{cases} 1 \text{ mit WS } p \\ 0 \text{ mit WS } q = 1-p \end{cases}$$

- **Erwartungswert** einer ZV:
  $\mathbb{E}[X] = \sum_{x \in \text{Wertebereich}} x \cdot \Pr[X = x]$
  (X ist diskret, d.h. der Wertebereich von X ist endlich)
- **Markov-Ungleichung**: $\Pr[X \geq t] \leq \frac{\mathbb{E}[X]}{t}$ für X nicht negativ
- **Chebyshev-Ungleichung**: $\Pr[|X - \mathbb{E}[X]| \geq t] \leq \frac{\text{Var}(X)}{t^2}$

Beweis (Forts.):

- **Binomialverteilung**: Seien $X_1, \ldots, X_n$ unabhängige, identisch verteilte Bernoulli-Zufallsvariablen mit $\Pr[X_i = 1] = p$.

$$X := \sum_{i=1}^{n} X_i \,.$$

X ist binomial verteilt, mit Wertebereich $\{0, 1, \ldots, n\}$.

$$\begin{aligned}
\Pr[X = k] &= \binom{n}{k} p^k (1-p)^{n-k} \\
\mathbb{E}[X] &= n \cdot p \\
\mathrm{Var}[X] &= n \cdot p \cdot (1-p) = n \cdot p \cdot q \,.
\end{aligned}$$

In Zeichen: $X \sim \mathrm{B}(n, p)$

Beweis (Forts.):

Die Auswahl der $n^{\frac{3}{4}}$ Elemente wird wiederholt, falls $|S_0| \geq \frac{n}{2}$. Dies passiert gdw wir höchstens $\frac{1}{2}n^{\frac{3}{4}} - \sqrt{n}$ Elemente aus der Hälfte aller Elemente $\leq$ dem Median auswählen.

Wir bestimmen die Wahrscheinlichkeit dafür, dass keine neue Auswahl der $n^{\frac{3}{4}}$ Elemente stattfinden muss.
Setze Bernoulli-Zufallsvariable $X_1, \ldots, X_n$ mit:

$$X_i = \begin{cases} 1, \text{ falls Element } i < \text{Median ausgewählt wird} \\ 0, \text{ sonst} \end{cases}$$

$X := \sum X_i$ ist binomialverteilt mit Parametern n und $\frac{1}{2}n^{-\frac{1}{4}}$, und $\mathbb{E}[X] = \frac{1}{2}n^{\frac{3}{4}}$, $\mathrm{Var}[X] = n \cdot \frac{1}{2}n^{-\frac{1}{4}}(1 - \frac{1}{2}n^{-\frac{1}{4}}) = \frac{1}{2}n^{\frac{3}{4}}(1 - o(1))$.

Beweis (Forts.):

Die Wahrscheinlichkeit hierbei ist

$$\Pr[|S_0| \geq \frac{n}{2}] = \Pr[X \leq \frac{1}{2}n^{\frac{3}{4}} - \sqrt{n}] \leq \Pr[|X - \mathbb{E}[X]| \geq \sqrt{n}]$$
$$\leq \frac{\frac{1}{2}n^{\frac{3}{4}}(1 - o(1))}{n} \leq \frac{1}{2}n^{-\frac{1}{4}}(1 - o(1))$$

Die anderen beiden Wahrscheinlichkeitsbedingungen ($\Pr[|S_2| \geq \lceil \frac{n}{2} \rceil]$ und $\Pr[|S_1| \geq 4 \cdot n^{\frac{3}{4}}]$) ergeben analoge Abschätzungen.

Damit: Wiederholung mit WS $\leq \mathcal{O}(n^{-\frac{1}{4}})$. $\square$