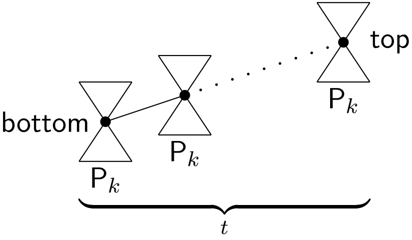
Kette von P_k's:



Gesamtzahl der Elemente:

n

$t(2k+1)$ in den P_k's

$r = n - t(2k+1)$ Rest

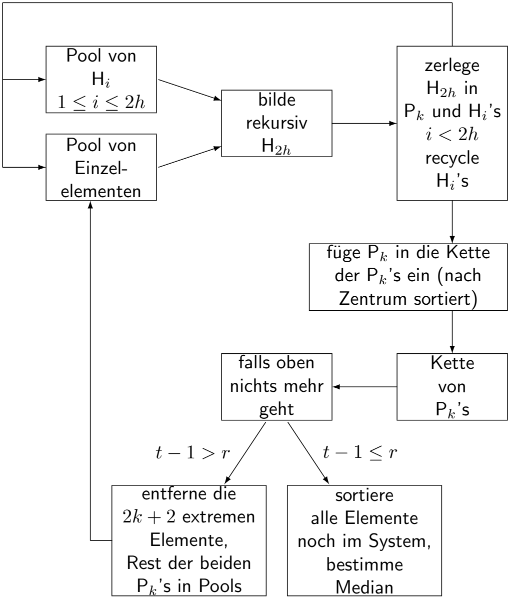
Wenn $r < t - 1$, dann wissen wir, dass top größer ist als

$$k + (t-1)(k+1) > k + (k+1)\left(\frac{n+1}{2k+2} - 1\right) = \frac{n-1}{2}$$

$\Rightarrow$ top $>$ Median (Entsprechendes gilt für das Element bottom)

Setze

$$k := \left\lfloor n^{\frac{1}{4}} \right\rfloor$$
$$h \text{ sdg. } 2^{h-1} \leq k < 2^h$$



Pool von H_i $1 \le i \le 2h$
Pool von Einzel-elementen
bilde rekursiv H_{2h}
zerlege H_{2h} in P_k und H_i's $i < 2h$ recycle H_i's
füge P_k in die Kette der P_k's ein (nach Zentrum sortiert)
Kette von P_k's
falls oben nichts mehr geht
$t - 1 > r$
$t - 1 \le r$
entferne die $2k + 2$ extremen Elemente, Rest der beiden P_k's in Pools
sortiere alle Elemente noch im System, bestimme Median

Definiere $r :=$ Anzahl der noch im H_{2h}-Produktionsprozess steckenden Elemente (für jedes $i < 2h$ höchstens ein H_i, daher $r \leq \sum_{i=0}^{2h-1} 2^i = 2^{2h} - 1$).

$R :=$ Anzahl der im letzten Schritt zu sortierenden Elemente. Es gilt: $t \leq r + 1$, und damit

$$R = t(2k+1) + r \leq 2^{2h}(2k+1) + 2^{2h} - 1\,.$$

$m :=$ Gesamtzahl der im Algorithmus produzierten P_k's.

$$m = t + 2\frac{n-R}{2(k+1)} = t + \frac{n-R}{k+1}\,.$$

Also: r und t sind $\mathcal{O}(n^{\frac{1}{2}})$, R ist $\mathcal{O}(n^{\frac{3}{4}})$.

Gesamtzahl der vom Algorithmus durchgeführten Vergleiche $=$

1. Anzahl der Kanten in allen P_k's
2. $+$ Anzahl der Kanten, die gelöscht werden, um die P_k's zu formen
3. $+$ Anzahl der Kanten, die zum Schluss in übriggebliebenen H_i's, $i < 2h$, stecken
4. $+$ Anzahl der Vergleiche, um jedes Zentrum der P_k's in die (sortierte) Kette einzufügen
5. $+$ Anzahl der Vergleiche, um die zum Schluss übriggebliebenenen R Elemente zu sortieren

$$\leq \left(\frac{n-R}{k+1}+t\right)\left[\underbrace{2k}_{1}+\underbrace{3k+2h}_{2}+\underbrace{\log\frac{n}{2k+1}}_{4}\right]+\underbrace{R\log R}_{5}+\underbrace{r}_{3}.$$

Mit $k = \left\lfloor n^{\frac{1}{4}} \right\rfloor$, h so, dass $2^{h-1} \leq k < 2^h$, ergibt sich damit

$$\begin{aligned} r &= \mathcal{O}(k^2) \\ t &= \mathcal{O}(k^2) \text{ zum Schluss} \\ R &= \mathcal{O}(k^3)\text{, und damit die} \\ \text{Anzahl der Vergleiche } &= T(n) \leq 5n + o(n)\,. \end{aligned}$$

Verbesserte Version (besseres Zurechtschneiden, bessere Verwertung der Reste):

$$T(n) = 3n + o(n)$$

Bester bekannter Algorithmus (von Dor/Zwick):

$$2,95n + o(n)$$

Literatur:

Arnold Schönhage, Michael Paterson, Nicholas Pippenger:
Finding the median
J. Comput. Syst. Sci. **13**, pp. 184–199 (1976)

Dorit Dor, Uri Zwick:
Selecting the median
SIAM J. Comput. **28**(5), pp. 1722–1758 (1999)

5. Eine untere Schranke für die Medianbestimmung

Satz 89

Jeder (vergleichsbasierte) Medianalgorithmus benötigt im worst-case mindestens $\lceil \frac{3n}{2} \rceil - 2$ *Vergleiche.*

Beweis:

Gegenspielerargument (adversary argument)

n Elemente, o.B.d.A. n ungerade, alle Elemente paarweise verschieden. Die Menge aller Elemente wird in drei Teilmengen partitioniert: U enthält die Kandidaten für den Median, G enthält Elemente, die sicher größer als der Median sind, und L enthält Elemente, die sicher kleiner als der Median sind. Anfangs sind alle Elemente in U.

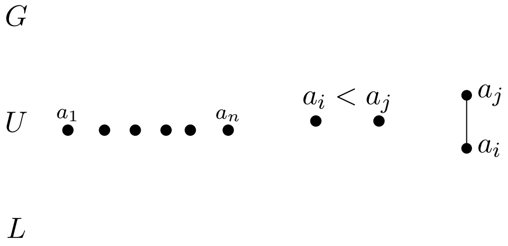
Beweis (Forts.):

Der Algorithmus stellt nun Fragen der Form $a_i < a_j$. Der Gegenspieler gibt konsistente Antworten, die den Algorithmus jedoch dazu zwingen, möglichst viele Fragen stellen zu müssen, bevor die Antwort feststehen kann (d.h. U soll möglichst ungeordnet bleiben).

Durch die (konsistenten!) Antworten des Gegenspielers auf die "$\stackrel{?}{<}$"-Queries des Algorithmus entsteht ein DAG. Der Gegenspieler hält diesen DAG "einfach", z.B. angenommen $y > z$ und $y > x$, dann soll y „sehr groß" sein $\Rightarrow y \to G$.

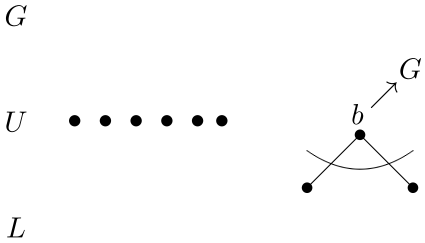
Strategie des Gegenspielers

Beweis (Forts.):



Strategie des Gegenspielers

Beweis (Forts.):

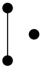


Strategie des Gegenspielers

Beweis (Forts.):

$G \quad \leq \frac{n+1}{2} - 1$

$U \quad \bullet \; \bullet \; \bullet \; \bullet \; \bullet \; \bullet$



Solange ein Element unverglichen ist, ist nicht klar, welches der Median ist.

$L \quad \leq \frac{n+1}{2} - 1$

Beweis (Forts.):

Solange $|L|, |G| < \frac{n+1}{2}$, kann der Algorithmus annehmen, dass der Median in U ist. Solange U mindestens zwei unverglichene Elemente und keine Zusammenhangskomponente mit > 2 Elementen enthält, kann der Algorithmus den Median nicht bestimmen.

Strategie des Gegenspielers

Beweis (Forts.):

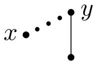
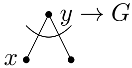
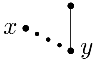
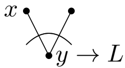
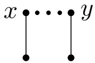
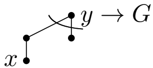
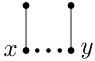
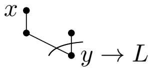
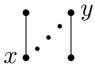
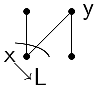
Die Strategie des Gegenspielers hat zwei Phasen.

Erste Phase: Query sei $x \overset{?}{<} y$:

i) $x, y \in G$ bzw. $x, y \in L$: irgendeine konsistente Antwort

ii) $x \in G \wedge y \in L \cup U$ (bzw. $x \in L \wedge y \in G \cup U$):
Antwort: $y < x$ (bzw. $x < y$).

iii) Sei $x, y \in U$.

	Query	Antwort des Gegenspielers	Anzahl der Paare in U	$\lvert U\rvert$	$\lvert L\rvert$	$\lvert G\rvert$
1.			$+1$	—	—	—
2.			-1	-1	0	$+1$
3.			-1	-1	$+1$	0
4.			-1	-1	0	$+1$
5.			-1	-1	$+1$	0
6.			-1	-1	$+1$	0

Die erste Phase endet, wenn $|L| = \frac{n-1}{2}$ oder $|G| = \frac{n-1}{2}$. Während der Phase 1 enthält U mindestens zwei (in U) maximale und mindestens zwei (in U) minimale Elemente (bzgl. des DAGs). $\Rightarrow$ Während Phase 1 kann der Algorithmus den Median mit Sicherheit nicht bestimmen.

Der Gegenspieler beginnt mit Phase 2, sobald

$$|L| \text{ wird } \frac{n-1}{2} \text{ oder } |G| \text{ wird } \frac{n-1}{2}.$$

O.B.d.A.:

$$|L| = \frac{n-1}{2}$$

Der Gegenspieler zwingt nun den Algorithmus, das minimale Element in U bzgl. der gesamten totalen Ordnung zu bestimmen (da dieses unter den Vorgaben der Median ist).