DSAL SoSe 2022

Alex-Daniel Tibelius 418656
Marc Gehring 358302
Ryskulbekov Malik 384898

Übungsblatt 6 27. Mai 2022

Aufgabe 4

Wir benutzen das Waagenprinzip. Wir teilen den Array in 2er-Gruppen (2 Nummern pro Gruppe), dann vergleichen wir jedes Element in der Gruppe miteinander. Das Element mit kleinerem "Gewicht"(Wert) gewinnt. In jeder Runde werden die Elemente weniger (n/2, n/4, usw).

Die maximale Anzahl an Runden ist $\lceil \log_2 n \rceil$, was deutlich wird, wenn man ungerade n einsetzt oder n, die keine ganzzahlige Potenz von 2 sind. Mit $r = 1/2 = 2^{-1}$, mit der Formel $\sum_{k=0}^{n} ar^k = a(\frac{1-r^{n+1}}{1-r})$ und weil $\lceil x \rceil < x + 1$ ist die Anzahl der Vergleiche deswegen maximal:

$\sum_{k=1}^{\lceil \log_2 n \rceil} nr^k = \sum_{k=0}^{\lceil \log_2 n \rceil} nr^k - n < \sum_{k=0}^{\log_2(n)+1} nr^k - n = n(\frac{1-r^{\log_2(n)+2}}{1-r}) - n = n\frac{1-(2^{-1})^{\log_2(n)+2}}{1-\frac{1}{2}} - n = n\frac{1-((2^{-1})^{\log_2 n} * (2^{-1})^2)}{\frac{1}{2}} - n = 2n(1-((2^{\log_2 n})^{-1} * \frac{1}{4})) - n = 2n(1-(\frac{1}{n} * \frac{1}{4})) - n = 2n(1-\frac{1}{4n}) - n = 2n(\frac{4n-1}{4n}) - n = \frac{4n-1}{2} - n = 2n - \frac{1}{2} - n = n - \frac{1}{2} < n$

Damit haben wir das kleinste Element gefunden mit höchstens n Vergleichen.

Um das zweitkleinste Element zu finden, müssen wir das kleinste Element mit allen Elementen, mit denen es schon verglichen wurde, wieder vergleichen. Im worst case wurde das zweitkleinste Element schon im ersten Vergleich mit dem kleinsten Element ëleminiert". Daher braucht man im worst case genau $log_2 n$ Vergleiche. Damit finden wir das zweitkleinste Element. Also braucht man insgesamt höchstens $n + log_2 n$ Vergleiche.

Aufgabe 5

Ja, Algo mit 7er Gruppen arbeitet in O(n) Zeit.
Für 5er Gruppen Zeitkomplexität:
$T(n) \leq T(n/5) + T(n * 7/10) + cn$

$T(n/5)$ - Median of median in 5er Gruppen (Quickselect) zu finden.

cn liegt in O(n) - Partition Teil

$T(7n/10)$ - Quickselect Rekursion worst case, k Element im grosseren Teil. Laut Vorlesung Pivot ist grosser als $3n/10$ Elements und kleiner als $3n/10$ anderen Elements.

Also induktiv die Gleichung ergibt $\approx 10cn \in O(n)$

Jetzt betrachten das gleiche für 7er Gruppe.

$T(n/7)$ - Median of median in 7er Gruppen zu finden (Quickselect), da jetzt wir alles in 7er Teilen.

cn liegt in O(n) - Partition Teil, gleich wie in 5er.

$T(\frac{5n}{7}+8)$ - Quickselect Rekursion worst case, k Element im grosseren Teil. Gleicher Lösungsweg wie mit 5er: in einer halbte des Sublists gibt es mindestens 4 Elements die grosser als Median of median sind. Da wir 2 Sublists nicht betrachten (1 mit Median of median und 2 (letzte), wo Anzahl von Elements bis 7), haben wir folgende Gelichung: $4 \cdot (\lceil \frac{1}{2} \lceil \frac{n}{7} \rceil \rceil - 2) \geq \frac{2n}{7} - 8$

Entsprechend Anzahl von Elements, die kleiner als median of median is auch $\frac{2n}{7} - 8$.Also für worst case, wo median of median stets in grossten Teil ist, gibt folgende Formel: $n - (\frac{2n}{7} - 8) = \frac{5n}{7} + 8$. Insgesamt für Variant mit 7er Gruppe haben wit folgende:

$$T(n) \leq T(\frac{n}{7}) + T(\frac{5n}{7}+8) + O(n)$$

Um Gleichung zu verifizieren, lösen wir Rekursion auf

$$T(n) \leq T(\frac{n}{7}) + T(\frac{5n}{7}+8) + an$$

$$\leq c(\frac{n}{7}) + c(\frac{5n}{7}+8) + a \cdot n$$

$$\leq \frac{cn}{7} + c(\frac{5n}{7}+8) + a \cdot n$$

Benutzen Definition von gross O.

$$\leq \frac{cn}{7} + c(\frac{5n}{7}+8) + a \cdot n \leq cn$$

$$\frac{6cn}{7} + 8c + an \leq cn$$

$$8c + an \leq \frac{cn}{7}$$

$$an \leq c(\frac{n}{7} - 8)$$

$$c \geq \frac{an}{(\frac{n}{7} - 8)} = \frac{7an}{n - 56}$$

Nach Definition für n>56, existiert c, sodass $c > \frac{7a}{1 - 56/a}$. Damit ist condition für O(n) erfüllt und Algo für 7er Gruppe laüft in linearer Zeit.

Aufgabe 6

a)

Hashing kann genutzt werden, um Keys einer langen Länge in Keys mit einer deutlichen kürzeren Länge zu konvertieren. Die vorige längere Länge man aus verschiedenen Gründen entstanden sein, aber gibt es insgesamt gar nicht so viele Keys wie der Wert des höchsten Keys vermuten ließe. Durch ein Hash-Funktion wird Input einer beliebigen Länge auf eine bestimmte, kleine Länge reduziert. Dadurch sinkt der Wertebereich, in dem man einen bestimmten key suchen muss. Das beschleunigt den Suchprozess, denn die herkömmlichen

Suchalgorithmen funktionieren nur gut in kleineren Wertebereichen. Das wird auch deutlich, wenn man sieht, dass diese Suchalgorithmen linear im Wertebereich komplex sind. Wichtig hierfür ist auch, dass die Hash-Funktion die Werte so gleichmäßig wie möglich auf den kleineren Wertebereich mappen sollte, um Kollisionen zu vermeiden. Der Nachteil von Hashtables ist jedoch der erhöhte Speicherverbrauch.

b)

c)

Man könnte denken, dass diese Lösung speicherplatztechnisch sinnvoll ist. Jedoch muss man beachten, dass das Kopieren und Einfügen auch aufwendig ist, besonders, wenn der Hashwert neu berechnet werden muss. Also lohnt sich der Algorithmus nur, wenn der Hashtable selten vergrößert werden muss.