Median and Order Statistics

Input: An array $A[1..n]$ of n distinct elements, an integer $1 \leq i \leq n$.

Output: The i-th largest element in the array A

Random-Select(S, i) $\quad\quad (i \leq |S|)$.

1. If $|S| = 1$ then return S.

2. Choose a random element y uniformly from S

3. Compare all elements of S to y. Let

$$S_1 = \{x \in S \mid x \leq y\}, \qquad S_2 = \{x \in S \mid x > y\}.$$

4. If $|S_1| = n$ then

   4.1 If $i = n$ return $\{y\}$, else $S_1 = S_1 - \{y\}$

5. If $|S_1| \geq i$ then return Random-Select(S_1, i) else return Random-Select$(S_2, i - |S_1|)$;

Correctness

Theorem 1. *The algorithm returns a singleton with the correct value.*

Proof.

By induction on the depth of the recursion, in each call to Random-Select(S', i'), $i' \leq |S'|$ and the i' largest element in S' is the i largest element in S.

When $|S'| = 1$, it includes the i largest element in S. $\square$

Run-time

Theorem 2. *The worst-case run-time of the algorithm is* $O(n^2)$.

Proof. In the worst case the size of the set that includes the i-th largest element decreases by one in each iteration. $\square$

Expected run-time

Theorem 3. *The expected run-time of the algorithm is $O(n)$.*

Proof.

Without loss of generality we can assume that in each iteration the i-th largest element is in the larger of the two sets S_1 and S_2.

$T(n)$ = the expected run-time on a set of n elements.

$$\begin{aligned} T(n) &\leq \frac{1}{n}\sum_{k=1}^{n-1} T(Max[k, n-k]) + \alpha n \\ &\leq \frac{2}{n}\sum_{k=\lceil n/2\rceil}^{n-1} T(k) + \alpha n \end{aligned}$$

We show that $T(n) \leq cn$ for some constant $c > 0$.

$$
\begin{aligned}
T(n) &\leq \frac{2}{n} \sum_{k=\lceil n/2 \rceil}^{n-1} ck + \alpha n \\
&\leq \left(\frac{2c}{n}\right)\left(\frac{1}{2}\right)\left(\frac{3n}{2}\right)\left(\frac{n}{2}\right) + \alpha n \\
&\leq \frac{3}{4} cn + \alpha n \\
&\leq cn
\end{aligned}
$$

$\square$

Linear Time Deterministic Selection Algorithm

Theorem 4. *There is a deterministic algorithm that finds the i-th largest element in an unsorted array of n elements in $O(n)$ time.*

Select (S, i) - Selects the i-th largest element in the set S.

1. $n = |S|$.
2. Partition S into $\lfloor \frac{n}{5} \rfloor$ groups of 5 elements each, and a leftover group of up to 4 elements.
3. Find the median of each of the groups, let R be the set of these $\lceil \frac{n}{5} \rceil$ values.
4. $y =$ Select$(R, \lfloor \frac{|R|}{2} \rfloor)$;
5. Compare all elements of S to y. Let

$$S_1 = \{x \in S \mid x \leq y\}, \qquad S_2 = \{x \in S \mid x > y\}.$$

6. If $|S_1| \geq i$ then return Select(S_1, i) else return Select$(S_2, i - |S_1|)$;

Correctness

Theorem 5. *The algorithms returns the correct value.*

Proof. By inductions on the calls to select() in step 6. $\square$

Run-time

Theorem 6. *The run-time of the algorithm is $O(n)$.*

Proof.

How many elements in S are larger than y, the "median of medians" value computed in step 4 of the algorithm?

Excluding the leftover group, and the group that includes y, in at least half of the remaining groups, there are at least three elements that are $> y$. Thus, at least

$$3(\frac{1}{2}\lceil\frac{n}{5}\rceil - 2) \geq \frac{3n}{10} - 6$$

in S are greater than y.

Similarly, at least $\frac{3n}{10} - 6$ elements in S are $\leq y$.

Thus, select is called in step 6 with at most $\frac{7n}{10} + 6$ elements.

$T(n) =$ run-time on sets of size n.

$$T(n) \leq T(\lceil \frac{n}{5} \rceil) + T(\frac{7n}{10} + 6) + \alpha n.$$

We show that $T(n) \leq cn$ for some constant $c > 0$.

$$\begin{aligned} T(n) &\leq c(n/5 + 1) + c(7n/10 + 6) + \alpha n \\ &\leq 9cn/10 + 7c + \alpha n \\ &\leq cn \end{aligned}$$

for $n > 70$ and sufficiently large c. $\square$