Tutorial 3: Finding the median

Definition 1 (Median problem)**.** An instance of the *median problem* is an array $A \in \mathbb{Z}^n$ of n integers. The valid solution to such an instance is the value $y = A'\big[n/2\big]$ when A' is obtained by sorting A.[1]

The simplest solution to the median problem sorts A and outputs the $\frac{n}{2}$-th value, an algorithm with time complexity $\Theta(n \log n)$. Your goal is to design a more efficient algorithm.

1 Median using approximate median

The approximate median problem is a relaxed version of the median problem defined as follows.

Definition 2 (Approximate median problem)**.** An instance of the *approximate median problem* is an array $A \in \mathbb{Z}^n$ of n integers. A valid solution to such an instance is any value $y \in A'\big[\frac{3n}{10}\big], \ldots, A'\big[\frac{7n}{10}\big]$ when A' is obtained by sorting A.

Assume you have an APPROXMEDIAN algorithm that solves the approximate median problem and has time complexity $\Theta(n)$. Using this algorithm and the Divide & Conquer approach, design an algorithm that solves the median problem and has time complexity $\Theta(n)$.

1.1 Hint

Consider solving a more general problem called *selection problem* instead of tackling the median problem directly.

Definition 3 (Selection problem)**.** An instance of the *selection problem* is an array $A \in \mathbb{Z}^n$ of n integers and an index $k \in \{1, \ldots, n\}$. The valid solution to such an instance is the value $A'[k]$ when A' is obtained by sorting A.

1.2 Solution

Consider the following algorithm.

Claim 1. *The* SELECTION *algorithm solves the selection problem.*

Proof. Proceed by induction on n.

Base Case: When $n = 1$, the only valid input for k is 1. The call to APPROXMEDIAN will return 1, and the for loop will add $A[1]$ to $S_=$ in its only iteration. The first **if** statement will then return $A[1]$, which is correct.

[1]Technically, the median can be either $A'\big[\lfloor n/2 \lfloor\big]$ or $A'\big[\lceil n/2 \lceil\big]$ when n is even, but we'll ignore floors and ceilings for this tutorial.

Algorithm 1: Selection($A = A[1], \ldots, A[n]$; k)

$\ell \leftarrow$ ApproxMedian(A);
$S_L \leftarrow \emptyset$; $S_= \leftarrow \emptyset$; $S_R \leftarrow \emptyset$;

for $i \leq n$ **do**
  if $A[i] < A[\ell]$, add $A[i]$ to S_L;
  if $A[i] = A[\ell]$, add $A[i]$ to $S_=$;
  if $A[i] > A[\ell]$, add $A[i]$ to S_R;

if $|S_L| < k$ and $|S_L| + |S_=| \geq k$ **return** $A[\ell]$;
if $|S_L| > k$ **return** Selection(S_L, k);
if $|S_L| + |S_=| < k$ **return** Selection(S_R, $k - (|S_L| + |S_=|)$);

Inductive Step: Let $n > 1$ and let $k \leq n$. Suppose for all $m < n$ and all $k' \leq m$ that Selection($A = A[1], \ldots, A[m]; k'$) solves the selection problem.

The call to ApproxMedian selects some pivot $1 \leq \ell \leq n$. The **for**-loop splits A into three sub-arrays: (1) S_L, containing elements less than $A[\ell]$, (2) $S_=$, containing elements equal to $A[\ell]$, and (3) S_R, containing elements greater than $A[\ell]$.

There are three cases to analyze. Case (1) is when $A[\ell]$ is the valid solution to the selection problem. Since $A[\ell]$ is the k-th smallest element in the array, we have $|S_L| < k$ and $|S_L| + |S_=| \geq k$, so the first **if** statement returns the correct solution. Case (2) is when $A[\ell]$ is greater than the valid solution. In this case, S_L contains the valid solution as its k-th smallest element. By the inductive hypothesis, the second **if** statement returns the valid solution. Case (3) is when $A[\ell]$ is less than the valid solution. In this case, S_R contains the valid solution as its $(k - (|S_L| + |S_=|))$-th smallest element. By the inductive hypothesis, the last **if** statement returns the valid solution. □

Claim 2. *The* Selection *algorithm has time complexity* $\Theta(n)$*.*

Proof. The call to ApproxMedian takes $\Theta(n)$ time. The **for**-loop takes $\Theta(n)$ time. Since ApproxMedian guarantees $A[\ell] \in A'[\frac{3n}{10}], \ldots, A'[\frac{7n}{10}]$, we know that $|S_L| \geq \frac{3n}{10}$ and $|S_R| \geq n - \frac{7n}{10} = \frac{3n}{10}$. So $T(n)$ satisfies the recurrence

$$T(n) \leq T'(n) = T'(\frac{7n}{10}) + \Theta(n)$$

By the Master Theorem, $T'(n) = \Theta(n)$, so $T(n) = O(n)$. Since the first call to Selection does $\Theta(n)$ work excluding the recursive calls, we also have $T(n) = \Omega(n)$, so $T(n) = \Theta(n)$. □

2 Approximate median using median

Assume that we have a Median algorithm that runs in time $\Theta(n)$, but can only find the median in sets of size at most $\frac{n}{5}$. Use this algorithm to solve the approximate median problem on instances of size n with and has time complexity $\Theta(n)$.

Algorithm 2: MEDIANOFMEDIANS($A = A[1], \ldots, A[n]$)

for $1 \leq i \leq n/5$ **do**
  $B[i] \leftarrow$ MEDIAN($A[5i-4], A[5i-3], A[5i-2], A[5i-1], A[5i]$);
return MEDIAN(B);

2.1 Solution

The solution is known as the *median-of-medians* algorithm.

Claim 3. *The* MEDIANOFMEDIANS *algorithm solves the approximate median problem.*

Proof. Let z be the integer returned by the MEDIANOFMEDIANS algorithm. z is the median of B, and $|B| = \frac{n}{5}$, so there are $\frac{n/5}{2} = \frac{n}{10}$ elements in B which are less than or equal to z. Furthermore, each of these $\frac{n}{10}$ elements was the median of some size-5 subarray of A. Therefore,3 of the elements in these subarrays are less than or equal to z. In total, there are $\frac{3n}{10}$ elements in A less than or equal to z.

By a symmetric argument, we can show there are at least $\frac{3n}{10}$ elements in A which are greater than or equal to z. Therefore, z is a valid solution to the Approximate Median problem. □

Claim 4. *The* MEDIANOFMEDIANS *algorithm has time complexity* $\Theta(n)$.

Proof. Within the **for**-loop, the call to MEDIAN takes $\Theta(1)$ time. The loop iterates $\frac{n}{5}$ times, so in total the loop takes $\Theta(n)$ time. The final call to MEDIAN is on an array of size $\frac{n}{5}$, so it takes time $\Theta(\frac{n}{5}) = \Theta(n)$. In total, $T(n) = \Theta(n)$. □

3 Linear-time median algorithm

Use the ideas developed above to design an algorithm that solves the median problem and has time complexity $\Theta(n)$.

3.1 Solution

Claim 5. *The* SELECTION *algorithm solves the selection problem.*

Proof. Proceed by induction on n.

Base Case: When $n \leq 5$, the algorithm always picks the pivot $\ell = 1$. The rest of the algorithm is the same as in question 1, so the inductive argument in that question can be mirrored here.

Inductive Step: Let $n > 5$ and let $k \leq n$. Suppose for all $m < n$ and all $k' \leq m$ that SELECTION($A = A[1], \ldots, A[m]; k'$) solves the selection problem.

We can combine the arguments used in questions 1 and 2. First, we show that ℓ is a solution to the Approximate Median problem. Since the calls to SELECTION which determine ℓ are on arrays of length less than n, applying the inductive hypothesis tells us that these calls do indeed return the median of their array arguments. That is, this is exactly the Median-of-Medians approach. So ℓ is set to a valid Approximate Median, which is a valid pivot index.

The remainder of the proof is exactly as written in the inductive step for question 1. □

Algorithm 3: SELECTION($A = A[1], \ldots, A[n];\ k$)

if $n \leq 5$, $\ell \leftarrow 1$;
else
  for $1 \leq i \leq n/5$ **do**
    $B[i] \leftarrow$ SELECTION($A[5i-4, \ldots, 5i]; 3$);
  $\ell \leftarrow$ SELECTION($B;\ \frac{n}{10}$);
$S_L \leftarrow \emptyset$; $S_= \leftarrow \emptyset$; $S_R \leftarrow \emptyset$;

for $i \leq n$ **do**
  if $A[i] < A[\ell]$, add $A[i]$ to S_L;
  if $A[i] = A[\ell]$, add $A[i]$ to $S_=$;
  if $A[i] > A[\ell]$, add $A[i]$ to S_R;

if $|S_L| < k$ and $|S_L| + |S_=| \geq k$ **return** $A[\ell]$;
if $|S_L| > k$ **return** SELECTION($S_L;\ k$);
if $|S_L| + |S_=| < k$ **return** SELECTION($S_R;\ k - (|S_L| + |S_=|)$);

Claim 6. *The* SELECTION *algorithm has time complexity* $\Theta(n)$.

Proof. In the definition of Θ, we can take $n_0 = 6$, so we may assume $n > 5$. Furthermore, we can say $T(n) = \Theta(1)$ for $n \leq 5$.

The **for**-loop takes $\Theta(n)$ time. The first 2 recursive calls take $T(5) = \Theta(1)$ and $T(n/5)$ time respectively. Since ℓ is a solution to the Approximate Median problem, the last recursive call will take at most $T(7n/10)$ time as argued in question 1. So $T(n)$ satisfies the recurrence

$$T(n) \leq T(n/5) + T(7n/10) + \Theta(n)$$

Drawing the recursive tree rooted at $T'(n)$, we see that the time spent on the i-th row of the tree can be summed up as $\sum_{j=0}^{i} (\frac{1}{5})^{i-j} (\frac{7}{10})^j n$. Summing over all rows and applying the binomial theorem and geometric series identity, we have

$$\begin{aligned} T(n) &= \sum_{i=0}^{\log n} \sum_{j=0}^{i} (\frac{1}{5})^{i-j} (\frac{7}{10})^j n \\ &= n \sum_{i=0}^{\log n} (\frac{1}{5} + \frac{7}{10})^i \\ &= n \sum_{i=0}^{\log n} (\frac{9}{10})^i \\ &= n \frac{1 - (\frac{9}{10})^{\log n}}{1 - 9/10} \\ &= 10n(1 - (9/10)^{\log n}) \leq 10n \end{aligned}$$

Therefore $T(n) = O(n)$. Since SELECTION also contains a $\Theta(n)$ loop, $T(n) = \Theta(n)$ as well. $\square$