# CS 161 Fall 2017: Section 2 Solutions

## Randomized Algorithms: Majority Element

Suppose we are given an array, $A$, of length $n$, with the promise that there exists some number, $x$, that occurs at least $n/2+1$ times in the array. Additionally, we are only allowed to check whether two elements are equal (no comparisons).

Last week, we developed an algorithm to find this in $O(n \log n)$ using divide and conquer. This week, we will develop a randomized algorithm that is expected to run in linear time.

(a) Complete the following pseudo-code for a randomized algorithm that returns the majority element of $A$.

```
MajorityElement(Input: array A of length n)
  While True
      i = ChooseRandomNumber(1,...,n)   //uniformly random number between 1 and n


      Let count = 0
      Foreach x in A
          if x = A[i]
              count++
      if count > n/2 return A[i]
```

(b) Is there a bound on the worst-case runtime of the above algorithm?

No; the algorithm may run indefinitely, if it keeps sampling $i$ such that $A[i]$ is not the majority.

(c) Prove that the *expected* number of equality checks is at most $2n$.

Each run of the inner for loop always takes $n$ equality checks. Count the expected number of times we choose a random number before selecting $i$ corresponding to a majority element. Given that there is a majority element, at least $n/2$ of the elements are equal to the majority element, which means we have $\geq 1/2$ chance of selecting the majority element each time. This leads to the following geometric series:

$$E[checks] \leq \sum_{i=0}^{\infty} \frac{n}{2^i} = 2n$$

## Select with Different Group Sizes

The algorithm Select finds the $k^{th}$ smallest element in an array of $n$ elements in $O(n)$ time. In class, we chose to split the $n$ numbers into groups of 5 elements, and used the "median of the medians" as a pivot. In this problem, we consider what would have happened if we had split the elements into groups of 3, rather than groups of 5. To simplify your arguments, feel free to assume that all array sizes are divisible by 3 at all recursive stages of the algorithm, and ignore rounding issues. The pseudocode for this new Select algorithm is given below—the only difference between this and the algorithm we saw in class is that the number “3” in ChoosePivot was replaced by “5”.

**Algorithm 1:** SELECT$(A, n, k)$

```
if $n == 1$ then
    return $A[1]$;
$p \leftarrow$ CHOOSEPIVOT$(A, n)$;
$A_< \leftarrow \{A(i) \mid A(i) < p\}$;
$A_> \leftarrow \{[A(i) \mid A(i) > p\}$;
if $|A_<| = k - 1$ then
    return $p$;
else if $|A_<| > k - 1$ then
    return SELECT$(A_<, |A_<|, k)$;
else if $|A_<| < k - 1$ then
    return SELECT$(A_>, |A_>|, k - |A +_<| - 1)$;
```

**Algorithm 2:** CHOOSEPIVOT$(A, n)$

```
Split $A$ into $n/3$ groups $s_1, \dots, s_{n/3}$ of 3 elements each;
for $i = 1$ to $n/3$ do
    $m_i = median(s_i)$; (this takes a constant number of operations, since $|s_i| = 3$
$M = \{m_1, \dots, m_{n/3}\}$;
return SELECT$(M, n/3, n/6)$; (this returns the median of $M$)
```

(a) Suppose we divide a list $A$ of $n$ distinct numbers into $n/3$ groups of 3, and let $p$ be the median of the medians of the sets. (This is the behavior of ChoosePivot, as described above.) Derive an upper bound on the number of elements of $A$ that are greater than $p$. (This same bound also applies to the number of elements that are less than $p$, right?)

Let $g = \lceil \frac{n}{3} \rceil$ denote the number of groups. We know that $\lceil g/2 \rceil - 1$ will have a median smaller than $p$. If $p$ is larger than a group's median, then $p$ is larger than at least 2 elements in the group. This applies to all of the groups. Since there may be a remainder group that does not have a group of 3, accounting for this last group, $p$ has to be greater than at least $2 * (\lceil \frac{g}{2} \rceil - 2)$ elements.

Therefore, the number of elements greater than $p$, $|A_\geq|$, is at most the number of elements not including the pivot, $n - 1$ minus the value above.

$$\begin{aligned} |A_\geq| &\leq (n-1) - 2 * (\left\lceil \frac{g}{2} \right\rceil - 2) \\ &\leq (n-1) - (\frac{n}{3} - 4) \\ &= \frac{2n}{3} + 3 \end{aligned}$$

It is fine if it is shown $|A_\geq|$ is shown to be $\frac{2n}{3} + C$ for some reasonable constant $C$ since it is an upper bound.

(b) (2 points) Derive a recurrence for the worst-case run time $T(n)$ for this variant of Select.

For each group of 3 elements, it takes constant time to sort and therefore, takes $O(n)$ to sort all groups. The recursive call in ChoosePivot to find the median of medians takes $T(g) = T(\lceil \frac{n}{3} \rceil) \leq T(\frac{n}{3} + 1)$ time.

By symmetry, $|A_<|$, the number of elements less than $p$ is also at most $\frac{2n}{3} + 3$. Hence, the runtime of the recursive SELECT is bounded by $T(\frac{2n}{3} + 3)$. The full recurrence can be written as

$$T(n) \leq T(\frac{n}{3} + 1) + T(\frac{2n}{3} + 3) + O(n)$$

for $n > 3$ and $T(n) \leq 1$ for $n \leq 3$.

(c) (2 points) Using the substitution/"guess and check" method, solve the recurrence from part b) to compute the runtime of this version of the Select algorithm. (*Hint:* The run time may *not* be $O(n)$. )

Let's test whether the runtime is $O(n)$. Using the substitution method, we need to show $T(n) \leq cn$ for all $n \geq n_o$ for some constants $c$ and $n_o$. Substituting,

$$\begin{aligned} T(n) &\leq c(\frac{n}{3} + 1) + c(\frac{2n}{3} + 3) + an \\ &= cn + 4c + an \end{aligned}$$

However, $cn + 4c + an \leq cn$ cannot be true for $n \geq n_o$ for any value $n_o$ and $c$. The runtime is not linear!

Let's make another guess: $T(n) = O(n \log n)$. Similarly, we must show $T(n) \leq cn \log n$ for all $n \geq n_o$ for some constants $c$ and $n_o$.

For the base case, $T(1) \leq c * 1 * \log 1 = 0$.

For the inductive hypothesis, we claim $T(k) \leq ck \log k$ for all $k < n$.

Now for the inductive step, we want to prove $T(n) \leq cn \log n$. For simplicity, we simplify our recurrence to $T(n) \leq T(\frac{n}{3}) + T(\frac{2n}{3}) + O(n)$. Substituting,

$$\begin{aligned} T(n) &\leq c(\frac{n}{3}) \log(\frac{n}{3}) + c(\frac{2n}{3}) \log(\frac{2n}{3}) + an \\ &= c\frac{n}{3}(\log n - \log 3) + c\frac{2n}{3}(\log n - log(\frac{2}{3})) + an \\ &= cn \log n + (a - \frac{c}{3} \log 3 - \frac{2c}{3} \log(\frac{3}{2})n \\ &= cn \log n + (a - c \log 3 + \frac{2c}{3} \log 2)n \\ &\leq cn \log n \end{aligned}$$

Therefore, we choose $c$ and some $n_o$ such that.

$$\begin{aligned} cn_o \log n_o + (a - c \log 3 + \frac{2c}{3} \log 2)n_o &\leq cn \log n \\ (a - c \log 3 + \frac{2c}{3} \log 2)n_o &\leq 0 \\ \frac{a}{\log 3 - \frac{2}{3} \log 2} &\leq c \end{aligned}$$

We can simply set $n_o$ to 1.