CSCI-GA.1170-001/002 Fundamental Algorithms October 6, 2015

Solutions to Problem 2 of Homework 4 (26 (+14) Points)

Name: GOWTHAM GOLI (N17656180) *Due: Tuesday, October 6*

Your are given an array $A[1] \dots A[n]$ of n "objects". You have a magic unit-time procedure $Equal(A[i], A[j])$, which will tell if objects $A[i]$ and $A[j]$ are the same. Unfortunately, there is no other way to get any meaningful information about the objects: e.g., cannot ask if $A[i]$ is "greater" than $A[j]$ of if it is more "sexy", etc., just the equality test. We say that A is a *repetitor* if it contains strictly more than $n/2$ elements which are all pairwise the same. In this case any of A's (at least $n/2$) repetitive elements is called *dull*. For example, if the "object" is a string, the array $(boring, funny, cute, boring, boring)$ is a repetitor where $boring$ is dull while $funny$ is not. On the other hand, the array $(hello, hi, bonjorno, hola, whasup)$ is not a repetitor. You goal is to determine if A is a repetitor, and, if so, output its dull "object" (which is clearly unique).

(a) (8 points) Design a simple divide-and-conquer algorithm for this problem running in time $O(n \log n)$. Make sure you argue the correctness and the running time.
(**Hint**: Prove that if A is a repetitor, at least one of its "halves" is as well.)

Solution:

Theorem 1. *If A is a repetitor, at least one of its "halves" is as well*

Proof. Let p be the dull of A. If the left half of A is not a repetitor then there are less than $n/4$ occurrences of p in $A[1 \dots mid]$ which implies that the right half $A[mid + 1 \dots n]$ has to have more than $n/4$ occurrences of p for p to be a dull of A. Therefore, the right half is a repetitor. Similarly, we can argue for the left half to be a repetitor.

If neither of the halves of A is a repetitor, it means that there are less than $n/4$ occurrences of p in $A[1 \dots mid]$ and $A[mid + 1 \dots n]$. Therefore, p cannot be the dull in this case

If both the halves of A are repetitors then $A[1 \dots mid]$ has to have more than $n/4$ occurrences of p and $A[mid + 1 \dots n]$ has to have more than $n/4$ occurrences of p for p to be a dull of A.

Therefore we can conclude that If A is a repetitor, at least one of its "halves" is as well □

Using the above theorem, we can implement a $O(n \log n)$ algorithm as follows

Psuedocode

```
1  Algorithm: FindDull(A, low, high)
2  if low is high then
3  |  return A[low]
4  end
5  else
6  |  mid ← (low + high)/2
7  |  leftDull ← FindDull(A, low, mid)
8  |  rightDull ← FindDull(A, mid + 1, high)
9  |  if leftDull is rightDull then
10 |  |  Return leftDull
11 |  end
12 |  else
13 |  |  lDullCount = CountOccurrences(leftDull, A[low ... high])
14 |  |  rDullCOunt = CountOccurrences(rightDull, A[low ... high])
15 |  |  if lDullCount > (high - low)/2 then
16 |  |  |  Return leftDull
17 |  |  else if rDullCount > (high - low)/2 then
18 |  |  |  Return rightDull
19 |  |  else
20 |  |  |  Return NoDull
21 |  end
22 end
```

Algorithm 2: Alorithm to find the dull OF A in $O(n \log n)$ time

□

(b) (4 Points) Remember, if A was an integer array, the procedure PARTITION(A, p, r) (see Section 7.1) makes $x = A[r]$ the pivot element and returns the index q, where the new value of $A[q]$ contains the pivot x, the new values $A[p \dots q-1]$ contain elements less or equal to x, and the new values $A[q+1 \dots r]$ contain values greater than x. Write the pseudocode of the modified procedure NEW-PARTITION(A, p, r), which only uses the *Equal* operator and returns q such that $A[q+1 \dots r]$ contain all the elements equal to x (while $A[p \dots q]$ contain all other elements).

Solution:

Psuedocode

```
1  Algorithm: NEW-PARTITION(A, p, r)
2  x ← A[r]
3  i ← p − 1
4  for j = p to r-1 do
5  |  if A[j] NOT EQUAL x then
6  |  |  i ← i + 1
7  |  |  SWAP(A[i], A[j])
8  |  end
9  end
10 SWAP(A[i + 1], A[r])
11 Return i + 1
```

Algorithm 3: Modified procedure NEW-PARTITION(A, p, r)

□

(c) (2 points) Consider the following, more general, algorithm REPEAT(A, n, t), which tells if some element of $A[1] \dots A[n]$ is repeated at least t times. (Clearly, REPETITOR can just call REPEAT with $t = n/2 + 1$.)

REPEAT$(A,\ n,\ t)$
If $n < t$ **Return** *no*
Pick $i \in \{1 \dots n\}$ *at random.*
Swap$(A[i], A[n])$
$q \leftarrow$NEW-PARTITION$(A,\ 1,\ n)$
If $n - q \geq t$ **Then Return** $(yes, A[n])$
Return REPEAT$(A,\ q,\ t)$

Argue that the algorithm above is correct.

Solution:

It is quite obvious that $t \leq n$, Therefore if $t > n$ the algorithm returns *no*

Let the randomly picked element be *key*. NEW-PARTITION$(A, 1, n)$ returns A such that $A[1 \dots q-1] \neq key$ and $A[q \dots n] = key$. Therefore the array has $n - q$ occurrences of *key*

If $n - q \geq t$ then *key* is repeated atleast t times. Else we keep repeating the procedure with a new key each time until we find an element that is repeated atleast t times □

(d) (3 points) Argue that the algorithm above always terminates in time $O(n^2)$ (irrespective of the random choices of i).

Solution: In the worst case scenario, If the array has all distinct elements and no element can occurs more than t times ($\because t \geq 2$), q will be $n-1$ in the first recursive call, $n-2$ in the second recursive call, $n-3$ in the third recursive and so on until 1 in the first recursive call

Therefore in the worst case, the running time $T(n)$ is

$$\begin{aligned} T(n) &= T(n-1) + n \\ &= T(n-2) + n - 1 + n \\ &= T(n-3) + n - 2 + n - 1 + n \\ &\vdots \\ &= 1 + 2 + \ldots + n - 1 + n \\ &= \frac{n(n+1)}{2} \\ &= O(n^2) \end{aligned}$$

□

(e) (3 points) Give an example of an (integer) array A and a value $t \geq 2$ where the algorithm indeed takes time $\Omega(n^2)$.

Solution: Let $A = [1, 2, 3, 4, 5, 6, 7, 8, 9]$ and $t = 3$. There is no element in A that occurs more than 3 times. The algorithm will run as follows

Irrespective of the element picked as the pivot, as all the elements are distinct, q will be $n-1$ (i.e 8) in the first recursive call, $n-2$ (i.e 7) in the second recursive call and so on until 1 in the first recursive call.

$\implies T(n) = 1 + 2 + \ldots + n = n(n+1)/2 = \Omega(n^2)$ □

(f) (4 Points) Let $T(n)$ be the worst case (over all arrays $A[1 \ldots n]$ and $t > n/2$ such that A contains t identical elements) of the *expected* running time of REPEAT(A, n, t) (over the random choice of i). For concreteness, assume NEW-PARTITION takes time exactly n. Prove that

$$T(n) \leq \frac{1}{2} \cdot T(n-1) + n$$

(**Hint**: Prove that in this case no recursive sub-call will be made with probability $t/n > 1/2$.)

Solution: Given that A contains t identical elements ($t > n/2$), Therefore A is repititor and let p be it's dull.

If p is randomly picked up as the pivot, then the algorithm will terminate after the first recursive call itself. Therefore the probability that there will be no recursive sub call is the (total number of occurrences of p)/(number of elements in A) $> (n/2n) = 1/2$

In the worst case, For a recursive call to happen, the pivot chosen has to be a unique non-dull (or the element with least number of repetitions if the elements are not distinct) element of A. Probability that the non-dull element will be chosen as a pivot is $< 1/2$

Therefore in the worst case of the expected running time,
$T(n) =$ (probability that the next recursive call will happen)(Time taken by the next recursive call) + n
$\implies T(n) \leq \frac{1}{2}.T(n-1) + n$ □

(g) (2 points) Show by induction that $T(n) \leq 2n$.

Solution:

Base case - $T(1) = 1 \leq 2.1 = 2$. Base case is true

Induction Hypothesis - $T(k) \leq 2k, \ \forall k = 1, \ldots, n-1$

Induction Step

$$\begin{aligned} T(n) &\leq \frac{1}{2}.T(n-1) + n \\ &\leq \frac{1}{2}(2(n-1)) + n \\ &= n - 1 + n \\ &\leq 2n \end{aligned}$$

By Induction, we can conclude that $T(n) \leq 2n$ □

(h*) (**Extra Credit**; 6 points) Consider the following test for repetitor. For 100 times, run $\textsc{Repeat}(A, n, n/2 + 1)$ for at most $4n$ steps. If one of these 100 runs ever finishes within $4n$ steps, use that answer. If none of the 100 runs terminates within $4n$ steps, return *no*. Argue that the running time of this procedure is $O(n)$. Then argue that the probability it returns the incorrect *no* answer (when it should have returned *yes*) is at most 2^{-100}.
(**Hint**: Show that when the answer is *yes*, the probability of not finding this answer in $4n$ steps is at most $1/2$. Google for "Markov's inequality" if you want to be formal.)

Solution: The test runs for at most $4n$ steps and 100 times, therefore total number of steps is $400n$. Therefore running time is $O(n)$ ($\because 400 << n$) □

(i**) (**Extra Credit**; 8 points) Try to design $O(n)$ deterministic test for a repetitor.

Solution:

Key Idea

The dull of the array remains preserved whenever we cancel a pair of distinct elements from the array

Psuedocode

```
1  Algorithm: FIND-DULL(A)
2  count ← 0
3  for i ← 0 to n-1 do
4  |  if count is 0 then
5  |  |  x ← A[i]
6  |  |  count ← 1
7  |  else if x ≠ A[i] then
8  |  |  count ← count − 1
9  |  Else  count ← count + 1
10 |  num ← COUNT-OCCURRENCES(x, A)
11 |  if num > n/2 then
12 |  |  Return x
13 |  else
14 |  |  Return NODULL
15 end
```

Algorithm 4: Algorithm to find dull of A in $O(n)$ time

To prove the correctness of the above algorithm it is sufficient if we prove that if A is a repititor and say α is the dull then at the end of the for loop $x = \alpha$

In the above algorithm, during i^{th} iteration, we compare $A[i-1]$ with x and cancel both if they are different, and increment count otherwise.

So if $count = 0$, then all elements upto $A[i-1]$ would have been eliminated through distinct-elements pair formations. If $count > 0$, then $\{x, \dots \text{ count times } \dots, x, A[i], \dots A[n-1]\}$ elements would have still survived at the end of the i^{th} iteration.
Let S_i be $\{x, \dots \text{ count times } \dots, x, A[i], \dots A[n-1]\}$ then α is a dull of S_i.

At the end of for loop $S_n = \{x, \dots \text{ count times } \dots, x, A[n], \dots A[n-1]\}$. It means that α is a dull of $\{x, \dots \text{ count times } \dots, x\}$ and $count > 0$. Hence $\alpha = x$ □