Homework 3, additional problems, solution set

1. At the root of the recursion tree you have a problem of size n where the non-recursive cost is n^2. At the next level, you have two problems each of size $n/2$. Each of these problems has a non-recursive cost of $(n/2)^2$ for a total cost of $2(n/2)^2 = n^2/2$. Following this, on level i there are 2^i problems of size $n/2^i$. The total non-recursive cost of level i is $n^2/2^i$.

The leaf node consists of $T(1)$, at level $\log_2 n$, and at this level there are $2^{\log n}$ problems of size 1 for a total cost of $n \cdot T(1)$. In sum,

size of subproblems	number of subproblems	non-recursive cost
$n = 2^k$	1	n^2
2^{k-1}	2	$2 \cdot (n/2)^2 = n^2/2^1$
2^{k-2}	2^2	$2^2 \cdot (n/2^2)^2 = n^2/2^2$
	$\dots$	
2^1	2^{k-1}	$n^2/2^{k-1}$
$2^0 = 1$	2^k	$2^k \cdot T(1)$

Adding this all together, we get

$$\begin{aligned} T(n) &= \sum_{j=0}^{k-1} \frac{n^2}{2^j} + 2^k \cdot T(1) = n^2 \sum_{j=0}^{\log n - 1} \frac{1}{2^j} + n \cdot T(1) + \\ &= n^2 \cdot \left(\frac{1 - (\frac{1}{2})^{\log n}}{1 - \frac{1}{2}} \right) + n \cdot T(1) = 2n^2 \cdot \left(1 - \frac{1}{n}\right) + n \cdot T(1) \\ &= (2n^2 - 2n) + n \cdot T(1) = \begin{cases} 2n^2 - n & \text{if } T(1) = 1 \\ 2n^2 - 2n & \text{if } T(0) = 0 \end{cases} \end{aligned}$$

Check: For $T(1) = 1$, the solution gives $T(1) = 2 - 1 = 1$, which is correct, and $T(2) = 8 - 2 = 6$, which is also correct. For $T(1) = 0$, the solution gives $T(1) = 2 - 2 = 0$ and $T(2) = 8 - 4 = 4$, which is also correct.

2. Let's compare $A[1]$ and $A[\lfloor n/2 \rfloor + 1]$. If $A[1] < A[\lfloor n/2 \rfloor + 1]$ we know the maximum element must lie in $A[\lfloor n/2 \rfloor + 1 : n]$, and otherwise, the largest value lies in $A[1 : \lfloor n/2 \rfloor]$. This leads to the following algorithm, akin to binary search.

```
FindMaximum($A, start, end$)
1   if $start = end$ then return $A[start]$
2   if $end = start + 1$ then
3       if $A[end] > A[start]$ then return $A[end]$
4       else return $A[start]$
5   $mid = \lfloor (start + end)/2 \rfloor + 1$
6   if $A[mid] > A[start]$ then FindMaximum($A, mid, end$)
7   else FindMaximum($A, start, mid - 1$)
```

The problem size reduces from n to at most $\lceil n/2 \rceil$ in the recursive call. Hence there are at most $\lceil \log n \rceil$ recursive calls, and as they each perform $O(1)$ work, this yields an $O(\log n)$ runtime.

3. We observe that if there is a majority value, it will also be a majority item in at least one of the following subsets: the first $\lfloor n/2 \rfloor$ items, or the last $\lceil n/2 \rceil$ items. Accordingly, we recursively find the solution for the elements in the first half of the array, and for the elements in the second half of the array. We then check whether either of these two items is a majority item for the whole array.

The following procedure has two return values $IsMaj$, which equals "yes" if there is a majority item, and equals "no" otherwise, and *Rep*, which is the value of the majority item if there is one and equals nil otherwise.

```
MAJORITY($A$, $begin$, $end$, $IsMaj$, $Rep$)
    $mid \leftarrow \lfloor (begin + end)/2 \rfloor$; $IsMaj \leftarrow no$; $Rep \leftarrow nil$;
    MAJORITY($A$, $begin$, $mid$, $IsMajLeft$, $leftRep$)
    MAJORITY($A$, $mid + 1$, $end$, $IsMajRight$, $rightRep$)

    $countLeft \leftarrow 0$
    $countRight \leftarrow 0$
    for ($i = begin$) to $end$
        if $A[i] = leftRep$ then $countLeft++$
        if $A[i] = rightRep$ then $countRight++$

    if $countLeft \geq \lfloor (begin - end + 1)/2 \rfloor + 1$ then $IsMaj \leftarrow yes$; $Rep \leftarrow leftRep$
    if $countRight \geq \lfloor (begin - end + 1)/2 \rfloor + 1$ then $IsMaj \leftarrow yes$; $Rep \leftarrow rightRep$
    return ($IsMaj, Rep$)
```

The non-recursive computation is a simple loop which takes $O(n)$ time. There are two recursive calls are to problems of sizes $\lfloor n/2 \rfloor$ and $\lceil n/2 \rceil$, which is the same as for MERGESORT. We've already seen that this yields an $O(n \log n)$ runtime.

Claim: If $A[1, \ldots, n]$ has a majority element then it is the majority element of at least one of $A[1, \ldots, \lfloor n/2 \rfloor]$ or majority element of $A[\lfloor n/2 \rfloor + 1, \ldots, n]$.

Proof: Let x be the majority element of A. Suppose for a contradiction that it is not the majority for either $A_1 = A[1, \ldots, \lfloor n/2 \rfloor]$ or $A_2 = A[\lfloor n/2 \rfloor + 1, \ldots, n]$.

Then it occurs at most $\lfloor \lfloor n/2 \rfloor / 2 \rfloor$ times in A_1 and at most $\lfloor (n - \lfloor n/2 \rfloor)/2 \rfloor$ times in A_2, for a total of at most $\lfloor \lfloor n/2 \rfloor / 2 \rfloor + \lfloor (n - \lfloor n/2 \rfloor)/2 \rfloor$ times in A; this is a total of at most $\lfloor n/2 \rfloor$ times in A; but this is a contradiction. Therefore x must be a majority item in at least one of A_1 or A_2.