Problem set 4

J. Mestre

You should solve all problems above the line before the tutorial. Tutors will call out students to solve these problems in front of the class during the tutorials.

You should attempt to solve some of the problems below the line. Tutors will go over these problems during the tutorial.

Problem 1: Given an array A holding n objects, we want to test whether there is a *majority* element; that is, we want to know whether there is an object that appears in more than $n/2$ positions of A.

Assume we can test equality of two objects in $O(1)$ time, but we cannot use a dictionary indexed by the objects. Your task is to design an $O(n \log n)$ time algorithm for solving the majority problem.

i) Show that if x is a majority element in the array then x is a majority element in the first half of the array or the second half of the array

ii) Show how to check in $O(n)$ time if a candidate element x is indeed a majority element.

iii) Put these observation together to design a divide an conquer algorithm whose running time obeys the recurrence $T(n) = 2T(n/2) + O(n)$

iv) Solve the recurrence by unrolling it.

Problem 2: The product of two $n \times n$ matrices X and Y is a third $n \times n$ matrix $Z = XY$, where the (i, j) entry of Z is $Z_{ij} = \sum_{k=1}^{n} X_{ik} Y_{kj}$. Suppose that X and Y are divided into four $n/2 \times n/2$ blocks each:

$$X = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \text{ and } Y = \begin{bmatrix} E & F \\ G & H \end{bmatrix}.$$

Using this block notation we can express the product of X and Y as follows

$$XY = \begin{bmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{bmatrix}.$$

In this way, one multiplication of $n \times n$ matrices can be expressed in terms of 8 multiplications and 4 additions that involve $n/2 \times n/2$ matrices. Let $T(n)$ be the time complexity of multiplying two $n \times n$ matrices using this recursive algorithm.

i) Derive the recurrence for $T(n)$. (Assume adding two $k \times k$ matrices takes $O(k^2)$ time.)

ii) Solve the recurrence by unrolling it.

Problem 3: Your friend Alex is very excited because he has discovered a novel algorithm for sorting an array of n numbers. The algorithm make three recursive calls on arrays of size $\frac{2n}{3}$ and spends only $O(1)$ time per call.

Alex thinks his breakthrough sorting algorithm is can be very fast but has now idea how to analyze it. You task is to help Alex.

Algorithm 1 NEW-SORT(A)

```
1. if $|A| < 3$ then
2.     sort $A$ directly
3. else
4.     NEW-SORT($A[0 : \frac{2n}{3}]$)
5.     NEW-SORT($A[\frac{n}{3} : n]$)
6.     NEW-SORT($A[1 : \frac{2n}{3}]$)
```

i) Find the time complexity of NEW-SORT

ii) Prove that the algorithm actually sorts the input array

Problem 4: The Hadamand matrices $H_0, H_1, H_2, \ldots$ are defined as follows

1. H_0 is the 1×1 matrix $[1]$
2. For $k > 0$, H_k is the $2^k \times 2^k$ matrix

$$H_k = \begin{bmatrix} H_{k-1} & H_{k-1} \\ H_{k-1} & -H_{k-1} \end{bmatrix}$$

Show that if v is a column vector of length $n = 2^k$, then the matrix-vector product $H_k v$ can be calculated in $O(n \log n)$ time.

Problem 5: Suppose we are given an array A with n distinct numbers. We say an index i is locally optimal if $A[i] < A[i-1]$ and $A[i] < A[i+1]$ for $0 < i < n-1$, of $A[i] < A[i+1]$ for if $i = 0$, or $A[i] < A[i-1]$ for $i = n-1$.

Design an algorithm for finding a locally optimal index using divide an conquer. Your algorithm should run in $O(\log n)$ time.

Problem 6 (Advanced): Design an $O(n)$ time algorithm for the majority problem.

Problem 7 (Advanced): Given two sorted lists of size m and n. Given an $O(\log m + \log n)$ time algorithm for finding the kth smallest element in the union of the two lists.