Fall 2019 B503 Homework 3

Due date: 11:59 pm, Oct. 20, 2019 Lecturer: Qin Zhang

Your Name:_____ Your University ID:_____

Instruction: Please submit your homework solution **before due date, via Canvas**. Homework solution must be typesetted in PDF format via LaTeX (preferred) or Word. Please add references to ALL the resources you have used for completing the assignment. You are allowed to discuss the assignment with other students, and if you do so, please list their names in the submission.

Due: TBD

Total points: 70

Late Policy: No extensions or late homeworks will be granted, unless a request is made to the course instructor before due date and written documents are provided to support the reason for late submission.

Problem 1 (10 points). Solve the following recursions using "Unrolling" and "Guess+Verify".

(a) $T(n) = 2T(\frac{n}{2}) + c$, with boundary condition $T(n) = 1$ for all $n \leq 1$.

(b) $T(n) = 2T(\frac{n}{2}) + c\sqrt{n}$, with boundary condition $T(n) = 1$ for all $n \leq 1$.

(c) $T(n) = 2T(\frac{n}{2}) + cn^2$, with boundary condition $T(n) = 1$ for all $n \leq 1$.

Problem 2 (10 points). An array A of length N contains all the integers from 0 to N except one (in some random order). In this problem, we cannot access an entire integer in A with a single operation. The elements of A are represented in binary, and the only operation we can use to access them is "fetch the jth bit of $A[i]$". Using only this operation to access A, give an algorithm that determines the missing integer by looking at only $O(N)$ bits. (Note that there are $O(NlogN)$ bits total in A, so we can't even look at all the bits). Assume the numbers are in bit representation with leading 0s.

Problem 3 (10 points). Suppose now that you're given an $n \times n$ grid graph G. (An $n \times n$ grid graph is just the adjacency graph of an $n \times n$ chess board. To be completely precise, it is a graph whose node set is the set of all ordered pairs of natural numbers (i, j), where $1 \leq i \leq n$ and $1 \leq j \leq n$; the nodes (i, j) and (k, ℓ) are joined by an edge if and only if $|i - k| + |j - \ell| = 1$.)

Each node v is labeled by a real number x_v; you may assume that all these labels are distinct. A node v of G is a *local minimum* if the label x_v is less than the label x_w for all nodes w that are joined to v by an edge.

You are given such a graph G, but the labeling is only specified in the following *implicit* way: for each node v, you can determine the value x_v by *probing* the node v.

Show how to find a local minimum of G using only $O(n)$ probes to the nodes of G. (Note that G has n^2 nodes.)

Problem 4 (10 points). An array $A[1, \ldots, n]$ is said to have a majority element if more than half of its entries are the same. Given an array, the task is to design an efficient algorithm to tell whether the array has a majority element, and, if so, to find that element.

The elements of the array are not necessarily from some ordered domain like the integers, and so there can be no comparisons of the form "is $A[i] > A[j]$?". (Think of the array elements as GIF files, say.) However you can answer questions of the form: "is $A[i] = A[j]$?" in constant time.

(a) Show how to solve this problem in $O(n \log n)$ time.

(b) Can you give a linear-time algorithm?

Problem 5 (10 points). The *Hadamard 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 using $O(nlogn)$ operations. Assume that all the numbers involved are small enough that basic arithmetic operations like addition and multiplication take constant time.

Problem 6 (10 points). You've been working with some physicists who need to study, as part of their experimental design, the interactions among large numbers of very small charged particles. Basically, their setup works as follows. They have an inert lattice structure, and they use this for placing charged particles at regular spacing along a straight line. Thus we can model their structure as consisting of the points $\{1, 2, 3, \ldots, n\}$ on the real line; and at each of these points j, they have a particle with charge q_j. (Each charge can be either positive or negative.)

They want to study the total force on each particle, by measuring it and then comparing it to a computational prediction. This computational part is where they need your help. The total net force on particle j, by Coulomb's Law, is equal to

$$F_j = \sum_{i<j} \frac{Cq_iq_j}{(j-i)^2} - \sum_{i>j} \frac{Cq_iq_j}{(j-i)^2}.$$

They've written the following simple program to compute F_j for all j:

---

for $j = 1, 2, \ldots, n$ **do**
  Initialize F_j to 0
  for $j = 1, 2, \ldots, n$ **do**
    if $i < j$ **then**
      Add $\frac{Cq_iq_j}{(j-i)^2}$ to F_j
    else if $i > j$ **then**
      Add $-\frac{Cq_iq_j}{(j-i)^2}$ to F_j
    end if
  end for
  Output F_j
end for

It's not hard to analyze the running time of this program: each invocation of the inner loop, over i, takes $O(n)$ time, and this inner loop is invoked $O(n)$ times total, so the overall running time is $O(n^2)$.

The trouble is, for the large values of n they're working with, the program takes several minutes to run. On the other hand, their experimental setup is optimized so that they can throw down n particles, perform the measurements, and be ready to handle n more particles within a few seconds. So they'd really like it if there were a way to compute all the forces F_j much more quickly, so as to keep up with the rate of the experiment.

Help them out by designing an algorithm that computes all the forces F_j in $O(n \log n)$ time.

Problem 7 (10 points). The problem of greatest common divisor (gcd) of two positive integers is defined as following: the largest integer which divides them both. Euclid's algorithm is a famous one to solve this problem, but we will look at an alternative algorithm based on divide-and-conquer.

(a) Prove that the following rule is true.

$$gcd(a,b) = \begin{cases} 2gcd(a/2, b/2) & \text{if } a, b \text{ are even} \\ gcd(a, b/2) & \text{if } a \text{ is odd, } b \text{ is even} \\ gcd((a-b)/2, b) & \text{if } a, b \text{ are odd and } a \geq b \end{cases} \quad (1)$$

(b) Give an efficient divide-and-conquer algorithm for greatest common divisor. Analyze the running time of your algorithm if a and b are n-bit integers? (In particular, since n might be large you cannot assume that basic arithmetic operations like addition take constant time.)