Ph.D. Qualifying Exam: Analysis of Algorithms

This is a closed book exam. The total score is 100 points. Please answer all questions.

(40 points) 1. The *Hadamard matrices* $H_0, H_1, H_2, \ldots$, are defined as follows:

- H_0 is the 1×1 matrix $[1]$
- 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 using $O(n \log n)$ operations. Assume that all the numbers involved are small enough that basic arithmetic operations like addition and multiplication take unit time.

Solution: For any column vector u of length n, let u_1 denote the column vector of length $n/2$ consisting of the first $n/2$ coordinates of u. Similarly, define u_2 to be the vector of the remaining coordinates. Note then that

$$(H_k v)_1 = H_{k-1} v_1 + H_{k-1} v_2 = H_{k-1}(v_1 + v_2)$$

and

$$(H_k v)_2 = H_{k-1} v_1 - H_{k-1} v_2 = H_{k-1}(v_1 - v_2)$$

Recursion 1: This shows that we can find $H_k v$ by calculating

$$v_1 + v_2$$

and

$$v_1 - v_2$$

and recursively computing

$$H_{k-1}(v_1 + v_2)$$

and

$$H_{k-1}(v_1 - v_2)$$

Recursion 2: We need only to compute two subproblems

$$H_{k-1} v_1$$

and

$$H_{k-1} v_2$$

and combining the solutions of the two subproblems using addition (+) and subtraction (-), both taking $O(n)$ time.

The running time of this algorithm by both recursions above can be described as

$$T(n) = 2T(n/2) + O(n)$$

where the linear term is the time taken to perform the addition and the subtraction. This has solution

$$T(n) = O(n \log n)$$

by the Master theorem.

(20 points) 2. For a set of variables $x_1, \ldots, x_n$, *equality constraints* are of the form "$x_i = x_j$" and *disequality constraints* are of the form "$x_i \neq x_j$." The *constraint satisfaction problem* is to check whether all constraints can be satisfied.

For example, the constraints

$$x_1 = x_2,\ x_2 = x_3,\ x_3 = x_4,\ x_1 \neq x_4$$

cannot be satisfied.

Give an efficient algorithm that takes as input m constraints over n variables and decides whether the constraints can be satisfied. You can assume that all the equality constraints are given before the disequality constraints.

Solution: We need only compute the equivalence relation defined by the equality constraints using the algorithm for connected components. Specifically, we construct a graph $G = (V, E)$ where $V = \{1, ..., n\}$. We have

$$(i, j) \in E$$

if $x_i = x_j$ is a equality constraint. Since equality is an equivalence relation, all the variables in each connected component of G must have the same value. The decomposition into connected components can be done in $O(m+n)$ time.

Once the equivalence relation is computed, we then consider the inequality constraints. A disequality constraint $x_i \neq x_j$ is satisfiable if and only if i and j are not in the same connected component in G. This can be checked in $O(1)$ time.

Overall, the algorithm takes linear time, $O(m+n)$ to run.

(40 points) 3. Given two strings $x = x_1 x_2 \cdots x_n$ and $y = y_1 y_2 \cdots y_m$, a common substring of length k is defined as

$$x[i..i+k-1] = y[j..j+k-1]$$

(*Note: a substring is different from a subsequence.*)

Let $k_{\max}$ be the length of a *longest common substring* of x and y. Design an algorithm that takes $O(mn)$ to find $k_{\max}$.

Solution: Let $K[i,j]$ be the length of a longest common substring ending at $x[i]$ and $y[j]$. If $x[i] \neq y[j]$, $K[i,j]$ is always 0. Initialize $K[i,0] = K[0,j] = 0$. We can calculate $K[i,j]$ recursively by

$$K[i,j] = \begin{cases} K[i-1,j-1]+1 & \text{if } x_i = y_j \\ 0 & \text{otherwise} \end{cases}$$

$K[i,j]$ computed as above indicates the length of the longest common substring ending at $x[i]$ and $y[j]$. Therefore, the length of the longest common substring must be an entry with the maximum value $K[i,j]$. Then $k_{\max}$ can be found by

$$k_{\max} = \max_{1 \leq i \leq n; 1 \leq j \leq m} K[i,j]$$

which implies a dynamic programming algorithm of $O(mn)$ running time.