COMP 3804 — Tutorial January 17, 2025

Problem 1: Define O, Ω, Θ, and o.

Problem 2: I am sure you all remember the trick that Gauss used, when he was a little boy, to prove that for any integer $n \geq 1$,

$$1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2}. \tag{1}$$

Use the definitions in Problem 1 to prove the following. For each case, give two proofs, one that uses (1) and one that does not use it.

- $1 + 2 + 3 + \cdots + n = O(n^2)$.
- $1 + 2 + 3 + \cdots + n = \Omega(n^2)$.
- $1 + 2 + 3 + \cdots + n = \Theta(n^2)$.

Problem 3: Let $n \geq 1$ be an integer and let $x \neq 1$ be a real number. Prove, without using induction, that

$$1 + x + x^2 + x^3 + \cdots + x^{n-1} = \frac{x^n - 1}{x - 1}.$$

Problem 4: Justin Bieber claims that

$$2^n = O(1).$$

Here is Justin's proof:

The proof is by induction. For the base case, if $n = 0$, then $2^n = 1$, which is a constant and, thus, $O(1)$.

For the induction step, let $n \geq 1$ and assume that $2^{n-1} = O(1)$. Then

$$2^n = 2 \cdot 2^{n-1} = 2 \cdot O(1),$$

thus, 2^n is at most 2 times a constant, which is a constant, i.e., it is $O(1)$.

Should Justin get an A+ for COMP 3804?

Problem 5: The Fibonacci numbers are recursively defined as follows: $F_0 = 0$, $F_1 = 1$, for each integer $n \geq 2$, $F_n = F_{n-1} + F_{n-2}$.

Prove that $F_n \geq 2^{n/2}$ for every integer $n \geq 6$.

Problem 6: Solve the following recurrence using the *unfolding method* that we have seen in class. Give the final answer using Big-O notation.

$$T(n) = \begin{cases} 1 & \text{if } n = 1, \\ n + 5 \cdot T(n/7) & \text{if } n \geq 7. \end{cases}$$

You may assume that n is a power of 7.

Problem 7: The function $T(n)$ is recursively defined as follows:

$$T(n) = \begin{cases} 1 & \text{if } 1 \leq n \leq 2, \\ n + T(n/3) + T(2n/3) & \text{if } n \geq 3. \end{cases}$$

Use the *recursion tree method* that we have seen in class to prove that $T(n) = \Theta(n \log n)$.

Problem 8: The Hadamard matrices $H_0, H_1, H_2, \ldots$ are recursively defined as follows:

$$H_0 = (1)$$

and for $k \geq 1$,

$$H_k = \left(\begin{array}{c|c} H_{k-1} & H_{k-1} \\ \hline H_{k-1} & -H_{k-1} \end{array} \right).$$

Thus, H_0 is a 1×1 matrix whose only entry is 1,

$$H_1 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix},$$

and

$$H_2 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}.$$

(8.1) Let $k \geq 0$ be an integer and let $n = 2^k$. How many entries does the matrix H_k have? Express your answer in terms of n.

(8.2) Describe a recursive algorithm BUILD that has the following specification:

> **Algorithm** BUILD(k):
> **Input:** An integer $k \geq 0$.
> **Output:** The matrix H_k.

For any positive integer n that is a power of 2, say $n = 2^k$, let $T(n)$ be the running time of your algorithm BUILD(k). Derive a recurrence for $T(n)$. Use the Master Theorem to give the solution to your recurrence.

(8.3) If x is a column vector of length 2^k, then $H_k x$ is the column vector of length 2^k obtained by multiplying the matrix H_k with the vector x.

Describe a recursive algorithm MULTIPLY that has the following specification:

Algorithm MULTIPLY(k, x)**:**
Input: An integer $k \geq 0$ and a column vector x of length $n = 2^k$.
Output: The column vector $H_k x$ (having length n).
Running time: must be $O(n \log n)$.

Explain why the running time of your algorithm is $O(n \log n)$. You are allowed to use the Master Theorem.
Hint: The input only consists of k and x. The matrix H_k is not given as part of the input.