Homework 4

Josh Bowden

CS 330 - Discrete Structures
Professor Sasaki

February 27, 2017

Problem 1. Let $P(n)$ be the statement that $1^3 + 2^3 + \cdots + n^3 = \left(\frac{n(n+1)}{2}\right)^2$, for positive integer n. Prove by induction that $\forall n \in \mathbb{N}\,(P(n))$.

Theorem. For every postive integer n,

$$\sum_{j=1}^{k} j^3 = \left(\frac{n(n+1)}{2}\right)^2$$

$\square$

Proof. By induction on n,

Base case: $n = 1$

$$f(x) = (x^2 + 2x)/(2x + 1)$$
$$g(x) = x^2$$

By definition, for $f(x)$ to be $O(g(x))$, $f(n) \leq c \cdot g(n)$ for positive constants c and n_0 such that $n \geq n_0$.

Then,

$$\frac{f(n)}{g(n)} < 1$$
$$\implies \frac{(n^2 + 2n)/(2n + 1)}{n^2} < 1$$
$$\implies \frac{n^2 + 2n}{n^2(2n + 1)} < 1$$
$$\implies \frac{n^2}{n^2(2n + 1)} + \frac{2n}{n^2(2n + 1)} < 1$$
$$\implies \frac{1}{2n + 1} + \frac{2}{2n^2 + n} < 1 \text{ for } n > 1$$

$\therefore (x^2 + 2x)/(2x + 1)$ is $O(x^2)$.

$\square$

Problem 2. The ternary search algorithm locates an element in a list of increasing integers by successively splitting the list into three sub-lists of equal (or as close to equal as possible) size, and restricting the search to the appropriate piece. Specify the steps of this algorithm.

(1) Let x be the integer to search for.

(2) Let L be a list of n elements.

(3) Let $\Delta n = n \backslash 3$ (where '\' denotes integer division)

(4) Let Λ_1, Λ_2, and Λ_3 denote the three sub-lists of near equal length,
where Λ_1 is the elements of L_0 to $L_{\Delta n}$,
and Λ_2 is the elements of $L_{\Delta n+1}$ to $L_{2\Delta n}$,
and Λ_3 is the elements of $L_{2\Delta n+1}$ to L_n.

(5) Compare the first last item in Λ_1 and the first item in Λ_3.

(6) Repeat the process by restarting at Step 1 where $L = \Lambda_i$ with the containing Λ list.

Problem 3. Use the definition of Big-O to show that $(x^2 + 2x)/(2x + 1)$ is $O(x^2)$.

Proof. Given,

$$f(x) = (x^2 + 2x)/(2x + 1)$$
$$g(x) = x^2$$

By definition, for $f(x)$ to be $O(g(x))$, $f(n) \leq c \cdot g(n)$ for positive constants c and n_0 such that $n \geq n_0$.

Then,

$$\begin{aligned}
&\frac{f(n)}{g(n)} < 1 \\
&\implies \frac{(n^2 + 2n)/(2n + 1)}{n^2} < 1 \\
&\implies \frac{n^2 + 2n}{n^2(2n + 1)} < 1 \\
&\implies \frac{n^2}{n^2(2n + 1)} + \frac{2n}{n^2(2n + 1)} < 1 \\
&\implies \frac{1}{2n + 1} + \frac{2}{2n^2 + n} < 1 \text{ for } n > 1
\end{aligned}$$

$\therefore (x^2 + 2x)/(2x + 1)$ is $O(x^2)$.

$\square$

Problem 4. Show that $x \log x$ is $O(x^2)$ but that x^2 is not $O(x \log x)$.

Proof. $O(x \log x) = O(x^2)$

Under a similar premise from the previous proof,

$$\begin{aligned}
&\frac{n \log n}{n^2} < 1 \\
&\implies \frac{\log n}{n} < 1 \text{ for } n > 1
\end{aligned}$$

$\therefore x \log x$ is $O(x^2)$. $\square$

Proof. $O(x^2) \neq O(x \log x)$

$$\frac{x^2}{x \log x} < 1$$
$$\implies \frac{1}{\log x} \not< 1 \text{ for any positive } n$$

$\therefore x^2$ is not $O(x \log x)$. $\square$

Problem 5. Show that each of these pairs of functions are of the same order.

(a) $\log_{10}(x^2 + 1), \log_2 x$

Proof. $O(\log_{10}(x^2 + 1)) = O(\log_2 x) = O(n \log_1 0n)$

Since, $\log_{10}(x^2 + 1) \not< 1$ for any positive n, we'll assume the order of $O(n \log n)$.

Then,

$$\frac{\log_{10}(n^2 + 1)}{n \log_{10} n} < 1 \text{ for a positive } n$$

$\therefore \log_{10}(x^2 + 1)$ is $O(n \log_{10} n)$.

Given that the Change of Base Formula is,

$$\log_a x = \frac{\log_b x}{\log_b a}$$

Then, $\log_2 x = \frac{\log_{10} x}{\log_{10} 2}$.

So,

$$\frac{\frac{\log_{10} n}{\log_{10} 2}}{n \log_{10} n}$$
$$\implies \frac{1}{n(\log_{10} 2)} < 1 \text{ for a positive } n.$$

$\therefore \log_{10}(x^2 + 1)$ is $O(n \log n)$.

$\therefore O(\log_{10}(x^2 + 1)) = O(\log_2 x) = O(n \log_{10} n)$. $\square$

(b) $\log_{10} x, \log_2 x$

Problem 6. (a) Given a real number x and a positive integer k, determine the number of multiplications used to find starting with x and successively squaring (to find x^2, x^4, x^8, and so on). (b) Is this a more efficient way to find xˆ(2ˆk) (where ˆ means exponentiation) than by multiplying x by itself the appropriate number of times?

(a) $log_2(k)$ operations of successive squaring

(b) Yes, since multiplying x by itself takes k operations

Problem 7. Determine the least number of comparisons (the best-case performance) needed

(a) To find the maximum of a sequence of n integers.
$O(n)$ assuming an unsorted list since all integers within the list must be tested to ensure the maximum integer was obtained.

(b) To locate an element in a list of n terms using a binary search.
$O(1)$ if the element is located on the first iteration.

Problem 8. Describe how the number of comparisons used in the worst case changes when the size of the list to be sorted doubles from n to $2n$, where n is a positive integer when these sorting algorithms are used.

(a) The selection sort begins by finding the least element in the list. This element is swapped to the front. Then the least element among the remaining elements is found and swapped into the second position. This procedure is repeated until the entire list has been sorted.
n^2 to $4n^2$

(b) The binary insertion sort is a variation of the insertion sort that uses a binary search technique rather than a linear search technique to identify the location to insert the i'th element in the correct place among the previously sorted elements.
n^2 to $4n^2$