CSE 373 Data Structures SP16 HW2

Problem 1 (5 pts)

$2/N$

37

$\sqrt{N}$

N

$N \log \log N$

$N \log N, N \log(N^2)$

$N \log^2 N$

$N^{1.5}$

N^2

$N^2 \log N$

N^3

$2^{N/2}$

2^N

Problem 2 (12 pts)

<table>
<tr><th></th><th>Big-Oh</th><th>n=20</th><th>n=200</th><th>n=2000</th></tr>
<tr><td>1</td><td>$O(n)$</td><td colspan="3" rowspan="4">machine dependent</td></tr>
<tr><td>2</td><td>$O(n^2)$</td></tr>
<tr><td>3</td><td>$O(n^3)$</td></tr>
<tr><td>4</td><td>$O(n^2)$</td></tr>
<tr><td>5</td><td>$O(n^5)$</td><td rowspan="2"></td><td colspan="2" rowspan="2">too long</td></tr>
<tr><td>6</td><td>$O(n^4)$</td></tr>
</table>

The discussion should make reference to how the amount of time needed to run the fragment grows roughly at the rate predicted by the Big-Oh analysis.

Problem 3 (3 pts)

$f(n) = O(g(n))$ iff there exist positive constants C and n_0 such that $f(n) \leq Cg(n)$ for all $n \geq n_0$. Any combination of C, n_0, and $g(n)$ that matches this definition suffices, however we aim to minimize $g(n)$.

$$\begin{aligned} f(n) &= 6n^3 + 30n + 403 \\ &\leq 6n^3 + 30n^3 + 403n^3 \\ &= 439n^3 \end{aligned}$$

Hence, $f(x) \leq Cn^3$ for $C = 439$ and $n \geq 1$ (here we have $n_0 = 1$), and so $f(x)$ is $O(n^3)$

Problem 4 (5 pts)

Basis: If `size = 0`, then `v` is empty and hence cannot contain `val`. The program correctly outputs 0.

Inductive Hypothesis: Assume the program provides the correct result for all arrays of `size` $\leq k$.

Inductive Step: We must show that the program will provide the correct result for all arrays of `size` $= k + 1$. Consider any array `v` of `size` $= k + 1$. The program checks if `val` is the $k + 1$ element of `v`. If it is, the program will correctly return 1. If it is not, the program will return the result of a search of the array composed of the first k values of `v`. Since we did not find our value in the $k + 1$ position of `v`, if it is to be found anywhere it will be found among the first k elements. Since we have assumed by the inductive hypothesis that the program will return the correct result for all arrays of `size` $\leq k$, we can assume that whatever is returned by this search of the first k elements of `v` will be correct. Thus the program provides the correct result for `v`.

Since `v` was any array of `size` $= k + 1$, we can assume the program provides the correct output for *all* arrays of this size. Since k is similarly arbitrary, we can assume that our program provides the correct output for all inputs.