Due: Friday, Sept. 27 *at start of your lecture section*

Problem Set 1

Instructions: We encourage you to try all of these problems, but please hand in only the ones labeled "Hand In". You will hand in your completed problem set at the beginning of class.
Honor Code for Problem Sets: You can work with other students in the course on these problems, but your written work should be your own. This means: Once you begin writing up the solutions, you may no longer confer with anyone else. Also, you may only use the resources provided by your instructors: the text, slides from class, and other handouts—and, of course, your own notes from class. You should note on your assignment those students with whom you collaborated.

Some Examples

Here are some sample $O()$ problems with solutions.

1. Show that $x^2 + 3x - 5$ is $O(x^2)$.

   Solution: Note that $x^2 + 3x - 5 \leq x^2 + 3x$, and if $x \geq 3$, then $3x \leq x^2$. Thus $x^2 + 3x - 5 \leq x^2 + x^2 = 2x^2$, and so letting $N = 3$ and $c = 2$, the definition of $O()$ is satisfied: $x^2 + 3x - 5 \leq cx^2$ for all $x \geq N$.

2. Show that n^3 is *not* $O(n^2)$.

   Solution: If n^3 were $O(n^2)$, then there would be some integer $N > 0$ and some constant $c > 0$ such that, for every $n \geq N$, $n^3 \leq cn^2$. But that would mean that $n \leq c$ for all $n \geq N$, which is impossible: For example, let $n = c + N$; then $n \geq N$ but $n \not\leq c$.

3. Show that $n^2 + 1000$ is $O(n^2)$.

   Solution: Here's another approach: Consider what happens to the ratio $(n^2 + 1000)/n^2$ as n gets very large.

   $$\lim_{n\to\infty} \frac{n^2 + 1000}{n^2} = \lim_{n\to\infty} (1 + 1000/n^2) = 1.$$

   This means that, given *any* $c > 1$ (e.g., $c = 1.1$), for all large enough values of n, $\frac{n^2+1000}{n^2} \leq c$. Thus $n^2 + 1000 \leq cn^2$ for large enough values of n.

   This technique is widely applicable. Note that we found a wide range of possible values for c (any $c > 1$), and we were able to show that for each c there exists *some* n_0 that works—even though we didn't find a precise value for n_0!

Big-O

(*7 points*) .... Hand In

Some of the statements below are true while others are not. Determine which are which and ustify your answers using arguments similar to those above.

a) $n^2 - 10n + 100$ is $O(n^2)$

b) n^2 is $O(n^2 - 10n + 100)$

c) $\log_2(x)$ is $O(x)$

d) x is $O(\log_2(x))$

e) $\sin(x)$ is $O(1)$ Note: $f(x)$ is $O(1)$ if $f(x) \leq c$ for some constant $c > 0$ and all large enough x.

f) n is $O(n \log_2(n))$

g) $n \log_2(n)$ is $O(n)$

(*3 points*) .... **Hand In**

In class I claimed that the worst-case running time (in terms of number of operations) of the `contains` method in the `Vector` class is $O(n)$. Justify my claim by (possibly over-) counting the number of operations that could ever be performed on a `Vector` of size n as a result of calling `contains`. The method `contains` is reproduced below for your convenience. You may assume that the `equals` method for type `E` has worst-case running time $O(1)$.

```
public boolean contains(E elem)
{
    int i;
    for (i = 0; i < elementCount; i++) {
        if (elem.equals(elementData[i])) return true;
    }
    return false;
}
```

Mathematical Induction

For each of the following problems, give a clear, complete proof **using mathematical induction**.

(*3 points*) .... **Hand In**

Prove that for all $n \geq 1, 1^2 + 2^2 + ... + n^2 = n(n+1)(2n+1)/6$.

Consider using summation notation in your proof: $\sum_{k=1}^{n} k^2 = \frac{n(n+1)(2n+1)}{6}$.

(*3 points*) .... **Hand In**

Let $s_1 = 1$ and $s_n = 2s_{n-1}$ for all $n > 1$. Prove that $s_n = 2^{n-1}$ for all $n \geq 1$.

(*3 points*) .... **Hand In**

Prove that the calling the recursive method $fib()$ on any $n \geq 2$ results in fewer than 2^n recursive calls to itself.

(*3 points*) .... **Practice**

Prove that for all $n \geq 10, fib(n) \geq (3/2)^n$. Hint: Base cases are $n = 10, 11$.

(*3 points*) .... **Practice**

Prove that the Towers of Hanoi algorithm for $n \geq 1$ disks described in class will find a solution that requires exactly $2^n - 1$ moves.