Quiz 1 covers everything up until and including the lecture on May 3. To study for this quiz, you should review your assignment(s) and the lecture notes. For further pratice, we're providing some extra problems below. We're also giving you model solutions at the end of this handout. You should attempt these problems prior to looking at the solutions.

1 Understanding Big O

Mark all that applies (one point per question; must get all correct):

1. The function $f(n) = 99,999,999$ is big-O of

   [a] $O(2^n)$ [b] $O(n^3)$ [c] $O(n \log n)$ [d] $O(n)$ [e] $O(\log n)$ [f] $O(1)$.

2. The function $f(n) = 42 + 100n + \log n$ is big-Θ ("theta") of

   [a] $\Theta(2^n)$ [b] $\Theta(n^3)$ [c] $\Theta(n \log n)$ [d] $\Theta(n)$ [e] $\Theta(\log n)$ [f] $\Theta(1)$.

3. The function $f(n) = n \log n + 10n^3 + 58n$ is big-O of

   [a] $O(2^n)$ [b] $O(n^3)$ [c] $O(n \log n)$ [d] $O(n)$ [e] $O(\log n)$ [f] $O(1)$.

2 Going Down By 2

Consider the following function:

```java
int fooBar(int[] x) {
    while (x.length > 1) {
        int[] y = new int[x.length/2];
        for (int i=0;i<x.length/2;i++) {
            y[i] = x[2*i];
            if (2*i+1 < x.length) y[i] = y[i]+x[2*i+1];
        }
        x = y;
    }
    return x[0];
}
```

Remember the cost of `new int[t]` takes $\Theta(t)$ time. Analyze the cost of `fooBar`.

3 Theta Bounds By Squeezing

An alternate defition of $\Theta(\cdot)$ is the following:

Say $f(n) = \Theta(g(n))$ if $f(n) = O(g(n))$ and $g(n) = O(f(n))$.

Use this definition to prove the following:

- $n^2 + n = \Theta(n^2)$
- $1 + 2 + 3 + \cdots + n = \Theta(n^2)$, without using the summation formula.

4 Model Solutions

1. *Understanding Big O:* (1) It's Big-O of everything. You should have marked all the choices. (2) It's only big theta of n. (3) It's big O of n^3 and 2^n.
2. *Going down by 2:* Like before, we'll analyze the cost of each iteration as a function of the length of x. Importantly, notice that the length of x changes in every iteration.
   In the iteration where the length of x is m, the cost of new-ing an array `y` is $O(m)$ and the for-loop costs $O(m)$. The other steps cost $O(1)$. Therefore, it takes $O(m)$ time—or $a \cdot m + b$ time—for an iteration whose starting length of x is m.
   Observe now that the length of x is halved in every iteration, until the length becomes 1. Therefore, the total cost is
   $$T(n) = (a \cdot n + b) + (a \cdot \tfrac{n}{2} + b) + (a \cdot \tfrac{n}{4} + b) + \cdots + (a \cdot 1 + b)$$
   Since we assumed that the starting length n is a power of two, there is a $k \in \mathbb{Z}_+ \cup \{0\}$ such that $n = 2^k$, and the equation above becomes
   $$\begin{aligned} T(n) &= (a \cdot 2^k + b) + (a \cdot 2^{k-1} + b) + (a \cdot 2^{k-2} + b) + \cdots + (a \cdot 2^0 + b) \\ &= a(1 + 2 + 2^2 + \cdots + 2^k) + b \cdot k \\ &= a(2^{k+1} - 1) + b \cdot k \end{aligned}$$
   But as $n = 2^k$, we have $k = \log_2 n$ and $T(n) = a(2n-1) + b\log_2 n = O(n)$.
3. *Theta bounds by squeezing:*
   - $n^2 + n = \Theta(n^2)$. Let $f(n) = n^2 + n$ and $g(n) = n^2$. Showing that $f(n) = O(g(n))$ is just a matter of taking limits. To show that $g(n) = O(f(n))$, we show that
     $$\lim_{n \to \infty} \frac{n^2}{n^2 + n} = 1$$
     Since both $f(n) = O(g(n))$ and $g(n) = O(f(n))$, we conclude that $f(n) = \Theta(g(n))$.
   - $1 + 2 + 3 + \cdots + n = \Theta(n^2)$. Let $f(n) = 1 + 2 + 3 + \cdots + n$ and $g(n) = n^2$. First, we show that $f(n) = O(g(n))$.
     $$f(n) = 1 + 2 + \cdots + n \leqslant \underbrace{n + n + \cdots + n}_{n \text{ terms}} = n \times n = n^2 = O(n^2).$$
     We now show that $g(n) = O(f(n))$. For this, we'll show that
     $$f(n) = 1 + 2 + \cdots + n \geqslant \tfrac{n}{2} + \ldots n \geqslant \underbrace{\tfrac{n}{2} + \tfrac{n}{2} + \cdots + \tfrac{n}{2}}_{n/2 \text{ terms}} = \tfrac{n}{2} \times \tfrac{n}{2} = \tfrac{1}{4} \cdot g(n)$$
     Hence, $g(n) \leqslant 4f(n)$ and $g(n) = O(f(n))$. In conclusion, because both $f(n) = O(g(n))$ and $g(n) = O(f(n))$, we know $f(n) = \Theta(g(n))$.