[TAP:RKTNZ] Complexity

- What is the running time of the following method:

```
public int log(int n) {
    int pow = 0;
    while (n > 1) {
        n = n / 2;
        pow++;
    }
    return pow;
}
```

A. O(log n)
B. O(n)
C. O(n log n)
D. $O(n^2)$
E. Whatever

Problem Solving Day!

Spring 2018

Profs Bill & Jon

Complexity Practice

- True or false:
  - $n^2-10n+100$ is $O(n^2)$ → T
  - n^2 is $O(n^2-10n+100)$ → T
  - $\log_2(n)$ is $O(n)$ → T
  - x is $O(\log_2(n))$ → F
  - $\sin(n)$ is $O(1)$ → T
  - n is $O(n\log_2(n))$ → T
  - $n\log_2(n)$ is $O(n)$ → F

Induction Practice

- Prove that merge sort time complexity is `O(n log(n))`
  - Prove for `n = 2^k`, $k \geq 0$ (it is true for other `n`, but harder to prove)
  - Let `T(2^k)` be the time it takes to merge sort 2^k elements.
  - That is, show that `T(2^k)` $\leq 2^k * k$

Base case: $T(2^0) \leq 0 * 2^0$

Assume: $T(2^{k-1}) \leq (k-1) * 2^{k-1}$
Show: $T(2^k) \leq (k) * 2^k$

$T(2^k) = 2*T(2^k/2) + 2^k$

$= 2*T(2^{k-1}) + 2^k$

$\leq 2*((k-1) * 2^{k-1}) + 2^k$ by IH

$=(k-1) * 2^k+2^k$

$=2^k (k-1+1)$

$=2^k * k$

Recursion Practice

- Write a recursive method that prints the digits of a number in reverse order.

```
public static void reverseDigits(int num) { … }
```

- Write a recursive method that multiplies two numbers, a and b (non-negative), using only addition:

```
public static int multiply(int a, int b) { … }
```

- Write a recursive method that replaces all instances of value a with value b in a Vector:

```
public static <E> void replace(Vector<E> data, E a, E b) { … }
```

Recursion Practice

- Write a recursive method that prints the digits of a number in reverse order.

```
public static void reverseDigits(int num) {
  if (num/10 == 0){
    System.out.print(num);
  }else{
    System.out.print(num%10);
    reverseDigits(num/10);
  }
}
```

Recursion Practice

- Write a recursive method that multiplies two numbers, a and b (non-negative), using only addition:

```
public static int multiply(int a, int b){
   if (b == 0)
     return 0;

   return a + multiply(a,b-1);
 }
```

Recursion Practice

- Write a recursive method that multiplies two numbers, a and b, using only addition:

```
public static int multiply(int a, int b){
   if (b == 0)
     return 0;

   if (b > 0)
     return a + multiply(a,b-1);
   else
     return -a + multiply(a,b+1);
}
```

Recursion Practice

- Write a recursive method that replaces all instances of value a with value b in a Vector:

```
public static <E> void replace(Vector<E>  data, E a, E b) {
  replaceHelper(data,a,b,0);
}

public static <E> void replaceHelper(Vector<E>  data, E a, E b, int idx){
  if (idx == data.size())
    return;

  if (data.get(idx).equals(a))
    data.set(idx, b);

  replaceHelper(data,a,b,idx+1);
}
```