CSCI 104L Lecture 19: Amortized Runtime

Recall the List ADT:

1. insert (int position, T value)
2. remove(int position)
3. set(int position, T value)
4. T get (int position)

Analyze the runtime analysis for each of these operations when using an array.

1. What would be the runtime of Get?

2. What would be the runtime of Remove?

3. What would be the runtime of Insert?

Amortized analysis takes the big picture and says: the first x operations will take no more than $\Theta(y)$ time, for an average of $\Theta(\frac{y}{x})$ per operation. Amortized runtime is the **worst-case average-case**.

- What would be the amortized runtime of Inserting at the end of an ArrayList?

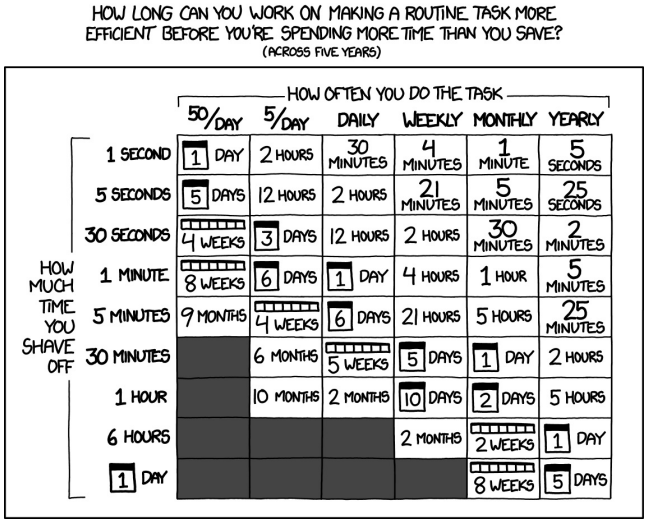

Figure 1: XKCD # 1205

Let's say we have a boolean array as a "counter". Each index starts at 0 (false), and then the counter starts counting up in binary.

Flipping an index from 0 to 1, or from 1 to 0, costs an operation. Counting from 0000 to 0001 only takes 1 operation, but counting from 0011 to 0100 takes 3 operations.

Our increment function should correctly increase the binary number by 1, flipping all necessary bits.

- What is the worst-case runtime of our increment function?
- What is the amortized runtime of our increment function?

Increment	Time	Total Time	Average Time
1	1	1	1
2	2	3	1.5
3	1	4	1.33
4	3	7	1.75
5	1	8	1.6
6	2	10	1.67
7	1	11	1.57
8	4	15	1.88
16	5	31	1.94
32	6	63	1.97
64	7	127	1.98