AMORTISED TIME COMPLEXITY:

By Ryan Beales

☆ The Amortised Time Complexity for a function is the average time it takes for each run, after running the function a large number of times, which I will denote as m times.

Example 1: Suppose we have a function with best and worst case time complexity of $O(1)$. Then, the time per operation of running this function m times is

$$\frac{\overbrace{O(1)+O(1)+\ldots+O(1)}^{m \text{ times}}}{m} = \frac{mO(1)}{m} = \frac{\cancel{m}O(1)}{\cancel{m}} = O(1).$$

Example 2: Suppose we have a function with best and worst case time complexity of $O(n)$. Then, the time per operation of running this function m times is

$$\frac{\overbrace{O(n)+O(n)+\ldots+O(n)}^{m \text{ times}}}{m} = \frac{mO(n)}{m} = \frac{\cancel{m}O(n)}{\cancel{m}} = O(n).$$

So, we can see that if the best and worst case have the same time complexity, then the amortised time complexity will match that as well. Things are potentially different if those cases are different.

Example 3: Suppose we have a function with best case time complexity $O(1)$, worst case time complexity $O(n)$, and does each of these cases half the time. Then, the time per operation of running this function m times is:

$$\frac{\overbrace{O(1)+O(n)+\dots+O(1)+O(n)}^{m\text{ times}}}{m} = \frac{\frac{\cancel{m}}{2}\cdot O(1)+\frac{\cancel{m}}{2}\cdot O(n)}{\cancel{m}} = \frac{1}{2}O(1)+\frac{1}{2}O(n)=O(n).$$

Example 4: Suppose we have a function which has an initial runtime of $O(n\log(n))$, and every other run is $O(\log(n))$. Then, the time per operation of running this function m times is

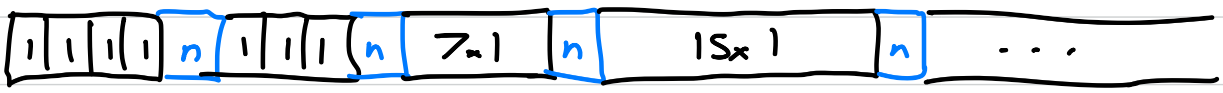
$$\frac{O(n\log(n))+\overbrace{O(\log(n))+\dots+O(\log(n))}^{(m-1)\text{ times}}}{m} = \frac{O(n\log(n))}{m}+\frac{(m-1)O(\log(n))}{m}$$

$$= \frac{O(n\log(n))^{*}}{m}+\frac{\cancel{m}O(\log(n))}{\cancel{m}}-\frac{O(\log(n))^{*}}{m}$$

$$=O(\log(n)).$$

*Note: as $m\to\infty$, $\frac{O(n\log(n))}{m}$ and $\frac{O(\log(n))}{m}\to 0$.

Example 5 — Array Queue Enqueue. Consider this function from the Week 3 Queue Analysis task. We have two cases: the normal insert, which is $O(1)$, and the case where we expand the capacity, which is $O(n)$ because of realloc. But, because the capacity is doubled each time, this happens:



Thus, when we write this out in a closed form formula, we get

$$\frac{\overbrace{1+1+1+1+n+1+1+1+n+1+\dots+1+n+1+\dots\dots+1+n+1+\dots}^{m\text{ times}}}{m}$$

$$=\frac{\log_2(m)\cdot n+(m-\log_2(m))\cdot 1}{m}$$ (Note: the $\log_2(m)$ comes from doubling)

$$=\frac{\log_2(m)\cdot O(n)^{*}}{m}+\frac{\cancel{m}\cdot O(1)}{\cancel{m}}-\frac{\log_2(m)\cdot O(1)^{*}}{m}$$

*Note from 1st year maths: $\frac{\log(x)}{x}\to 0$ as $x\to\infty$

$$=O(1).$$