# Calculating the Amortized time of my implementation of push and pop

By- Kabir Uberoi (Assignment 2 bonus)

* **Push** − suppose we do N pushes into the variable stack

  $Case\ 1: N < 1024$

  $\rightarrow$ *then each push happens in* $O(1)$*time so the amortized time will be*

  $= O\left(\frac{1+1+1 \ldots n\ times}{n}\right) = O(1)$

  $Case\ 2: N \geq 1024 (N = 1024 * 2^k)$

  $\rightarrow$ *then after each* $(1024 * 2^x)$ *pushes we will need to make a new array of the size* $(1024 * 2^x * 2)$, *copy the* $(1024 * 2^k)$ *elements and delete the old* $(1024 * 2^x)$*array. So, after every n pushes such that* $n = (1024 * 2^x)$, *we will have to do* $4n$ *operations.*
  *so, the amortized time would be* −

$$\frac{O\left(N+4096 * \sum_{k=1}^{log_2\left(\frac{n}{1024}\right)} 2^k\right)}{N} = O(1)$$

  $Case\ 3: N \geq 1024 (N \neq 1024 * 2^k)$

  $\rightarrow$ *then first push till* $n < N$ *such that n is the largest number of the form* $(1024 * 2^k)$ *which will happen in* $O(1)$*amortized time* $(as\ proven\ in\ case\ 2)$ *and then the rest of the push will happen in* $O(1)$ *amortized time as the array has been allocated and now we only have to change the value of specific indices of the array.*

Hence the Amortized time for Push operation is $O(1)$.

* **Pop** − suppose we do N pops from the variable stack of size N

  $Case\ 1: N < 1024$

  $\rightarrow$ *then each pop happens in* $O(1)$*time so the amortized time will be*

  $= O\left(\frac{1+1+1 \ldots\ n\ times}{n}\right) = O(1)$

  $Case\ 2: N \geq 1024 (N = 1024 * 2^k)$

  $\rightarrow$ *just like in push, this time also the resising of the array happens* $log_2\left(\frac{n}{1024}\right)$ *times and each time the resizing happens, if the current size is k, the number of steps* $= 2k$

$so, the\ amortized\ time\ would\ be -$

$$\frac{O\left(N+2048*\sum_{k=1}^{log_2\left(\frac{n}{1024}\right)}2^k\right)}{N} \quad = \quad O(1)$$

$Case\ 3: N \geq 1024 (N \neq 1024 * 2^k)$

$\rightarrow then\ pop\ till\ the\ size\ of\ the\ stack\ becomes \frac{1}{4} th\ of\ the\ stack\ capacity.$
$This\ will\ happen\ in\ O(1)\ amortized\ time\ as\ no\ new\ array\ is\ being\ allocated\ during\ this\ time\ and\ after\ this\ it\ is\ the\ same\ as\ case\ 2\ which\ we\ have\ proven\ to\ run\ in\ O(1)\ amortized\ time.$

Hence the amortized time for Pop operation is O(1).