

Amortized complexity of inserting n elements in an ArrayList where the capacity is doubled each time the array is full:

i^{th} insertion	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	..
capacity	0	1	2	4	4	8	8	8	8	16	16	16	16	16	16	16	16	32	..
allocate	0	1	2	4	0	8	0	0	0	16	0	0	0	0	0	0	0	32	..
copy	0	0	1	2	0	4	0	0	0	8	0	0	0	0	0	0	0	16	..
write	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	..
free	0	0	1	2	0	4	0	0	0	8	0	0	0	0	0	0	0	16	..

Assumptions:

- Writes " $a[i] = v$ " and copies " $a[i] = b[i]$ " take at most " $\bar{w}$ " and " $\bar{c}$ " steps (both are constant time).
- Array allocation " $T^* a = \text{new } T[2 \cdot i]$ " and deletion " $\text{delete } a$ " take at most " $\bar{a}$ " and " $\bar{d}$ " steps (also constant time). Actually, they depend on OS, machine, states of other processes...

We want to bound the total number of steps

Observe

- $\text{write}(i) = 0$ for $i=0$ and 1 otherwise.

- $\text{copy}(i) = 2^p$ if $(i-1) = 2^p$ for some p and 0 otherwise —

- $\text{allocate}(i) = 1$ if $i = 2^p$ for some p and 0 otherwise —

- $\text{delete}(i) = \text{allocate}(i)$ —

$$\begin{aligned} \sum_{i=0}^n \text{write}(i) &= n \\ \sum_{i=0}^n \text{copy}(i) &= \sum_{k=0}^{\lfloor \log_2(n-1) \rfloor} 2^k = 2^{\lfloor \log_2(n-1) \rfloor + 1} - 1 \\ \sum_{i=0}^n \text{allocate}(i) &= \sum_{i=0}^n \text{delete}(i) = \sum_{k=0}^{\lfloor \log_2(n) \rfloor} 1 = \lfloor \log_2 n \rfloor \end{aligned}$$

The total number of steps " $t(n)$ " of inserting n elements:

$$\begin{aligned} t(n) &\leq \bar{w} \cdot n + \bar{c} \cdot \left(2^{\lfloor \log_2(n-1) \rfloor + 1} - 1 \right) + (\bar{a} + \bar{d}) \cdot \lfloor \log_2 n \rfloor \\ &\leq \bar{w} \cdot n + \bar{c} \cdot (2 \cdot (n-1) - 1) + (\bar{a} + \bar{d}) \cdot n \\ &\leq (\bar{w} + 2 \cdot \bar{c} + \bar{a} + \bar{d}) \cdot n \end{aligned}$$

amortized complexity

Amortized complexity $t(n)/n$ is bounded by $(\bar{w} + 2 \cdot \bar{c} + \bar{a} + \bar{d})$ and is therefore in $O(1)$