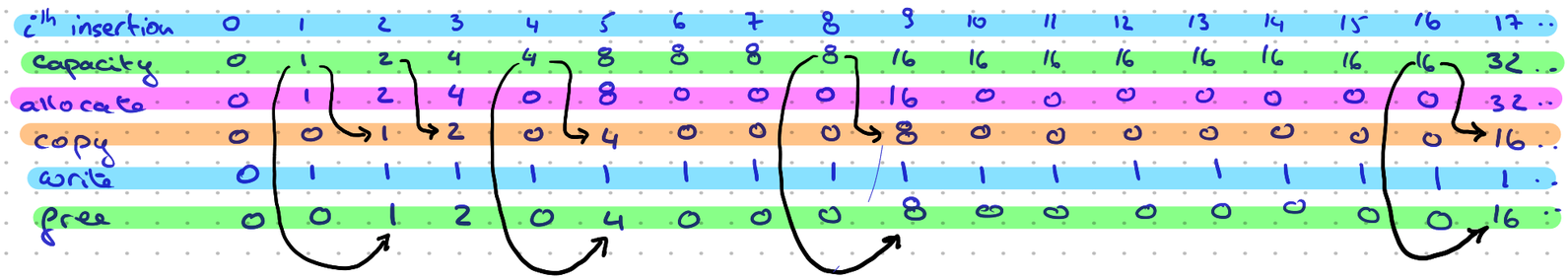
Amortized Complexity of inserting n elements in an ArrayList where the capacity is doubled each time the array is full:



Assumptions:

- Writes "$a[i] = v$" and copies "$a[j] = b[j]$" 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, status of other processes ...

We want to bound the total number of steps

Observe

- $\text{write}(i) = 0$ for $i = 0$ and 1 otherwise. $\longrightarrow \sum_{i=0}^{n} \text{write}(i) = n$
- $\text{copy}(i) = 2^j$ if $(i-1) = 2^j$ for some j and 0 otherwise $\longrightarrow \sum_{i=0}^{n} \text{copy}(i) = \sum_{k=0}^{\lfloor \log_2(n-1) \rfloor} 2^k = 2^{\lfloor \log_2(n-1) \rfloor + 1} - 1$
- $\text{allocate}(i) = 1$ if $i = 2^j$ for some j and 0 otherwise $\longrightarrow \sum_{i=0}^{n} \text{allocate}(i) = \sum_{i=0}^{n} \text{delete}(i) = \sum_{k=0}^{\lfloor \log_2(n) \rfloor} 1 = \lfloor \log_2 n \rfloor$

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

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

amortized complexity

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