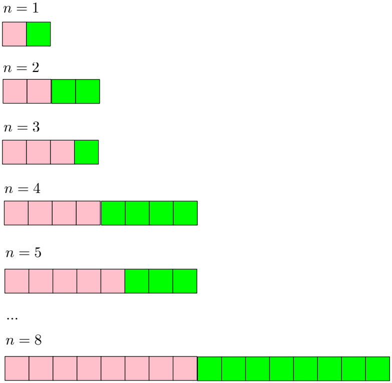
Further Discussions on Dynamic Arrays

Yufei Tao's Teaching Team

Recall: Dynamic Arrays



Double the array size when it is full.

> Why doubling? What if we triple the array size instead?

Another Version — Tripling

- Initially, the array has size 3. Define $s_1 = 3$.
- At the first expansion, the array size increases from s_1 to $s_2 = 3s_1 = 9$.
- At the second expansion, the array size increases from s_2 to $s_3 = 3s_2 = 27$.
- $\cdots$
- At the i-th expansion, the array size increases from s_i to $s_{i+1} = 3s_i = 3^{i+1}$

The cost of the i-th expansion is $O(3^{i+1})$.

> The array size is at most $3n$ where n is the number of elements inserted so far.

Another Version — Tripling

Suppose there are h expansions. Their total cost is bounded by $\sum_{i=1}^{i=h} O(3^{i+1}) = O(3^{h+1})$.

Hence, the total insertion cost is $O(n + 3^h)$.

After the h-th expansion, the array size is 3^{h+1}. As we never use more than $3n$ cells, we know: $3n > 3^{h+1}$, meaning $3^h < n$.

Therefore, $O(n + 3^h) = O(n)$.

The Best Expansion Coefficient?

Now, consider the general case where an expansion increases the array size by α times for some integer $\alpha \geq 2$. This ensures that the array size is at most αn where n is the number of elements inserted so far.

Which α is the best?

The General Algorithm

- Initially, the array size is α. Define $s_1 = \alpha$.
- At the first expansion, the array size increases from s_1 to $s_2 = \alpha s_1 = \alpha^2$.
- At the second expansion, the array size increases from s_2 to $s_3 = \alpha s_2 = \alpha^3$.
- ...
- At the i-th expansion, the array size increases from s_i to $s_{i+1} = \alpha s_i = \alpha^{i+1}$

The cost of the i-th expansion is $O(\alpha^{i+1})$.

Analysis

Consider inserting n integers into the array.

Suppose there are h expansions. Their total cost is $O(\sum_{i=1}^{i=h} \alpha^{h+1}) = O(\frac{\alpha^{h+2}}{\alpha-1})$.

The total insertion cost is $O(n + \frac{\alpha^{h+2}}{\alpha-1})$.

As we never use more than αn space, we know $\alpha n > \alpha^{h+1}$, which gives $\alpha^h < n$. Hence, the total cost is $O(n + \frac{\alpha^2}{\alpha-1} n)$, implying that the amortized cost is $O(1 + \frac{\alpha^2}{\alpha-1})$.

The term $\frac{\alpha^2}{\alpha-1}$ achieves its minimum when $\alpha = 2$.