Abstract

1 Introduction

2 Definitions

3 Predefined Functions

4 Fields and Properties

In this section, I define **fields** and **properties** on dynamic arrays and resize lists. These will be used later to implement operations on them in pseudocode.

5 Common Resize List Operations

In this section, I implement common operations for dynamic arrays, then resize lists, in pseudocode. Then, I analyze and compare the time complexities of the implementations. If the operation allocates memory, I also compare their space complexities.

The following mathematical definitions will be used while analyzing time and space complexity:

$P(f(n))$ This is an alternative to big-O notation for time complexity; I will call it **big-P notation**.

The main purpose of big-P notation is to preserve the constant coefficient of the fastest-growing term in $f(n)$, while still maintaining many properties of big-O. For example, $O(2n) = O(n)$, but $P(2n) \neq P(n)$. This makes it possible to compare time complexities whose ratio converges to a constant as n becomes larger.

More formally, $P(f(n)) = P(g(n))$ iff

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = 1$$

.

5.1 Adding

Adding is the most common operation done on a dynamic array[1]. Resize lists improve the performance of adding in two ways: by allocating less memory, and reducing the amount of copying.

Time complexity

$$W(n) = P(2n)$$

Space complexity

$$S(n) = P(2^{\lceil \log_2 n \rceil + 1}) = O(n)$$

Time complexity

$$W'(n) = P(n)$$

Space complexity

$$S'(n) = P(2^{\lceil \log_2 n \rceil}) = O(n)$$

Time complexity comparison

Space complexity comparison

$$r_S = \frac{P(2^{\lceil \log_2 n \rceil})}{P(2^{\lceil \log_2 n \rceil + 1})} = \frac{1}{2}$$

[1]

5.2 Indexing

Time complexity

$T(n) = O(1)$

Time complexity

$T'(n) = O(1)$

Time complexity comparison

5.3 Iterating

Time complexity

$T(n) = O(n)$

Time complexity

$T'(n) = O(n)$

Time complexity comparison