

5

array list, growable array, resizable array, mutable array **Quick reference** Python objects are all dynamic (ie list, array, dataframe)

Dynamic Array

Data Structure

the size ahead of time.

Strengths:

Other names:

A **dynamic array** is an array with a big improvement: automatic resizing, space One limitation of arrays is that they're lookup fixed size, meaning you need to specify the append number of elements your array will hold ahead of time. insert A dynamic array expands as you add more delete elements. So you don't need to determine

time. Variable size. You can add as many items as you want, and the dynamic array will expand to hold them. **Cache-friendly**. Just like arrays, dynamic arrays place items right next to each other in memory, making efficient use of caches. Weaknesses: **Slow worst-case appends**. Usually, adding a new element at the end of the dynamic array takes O(1) time. But if the dynamic array doesn't have any room for the new item, it'll need to expand, which takes O(n) time.

Costly inserts and deletes. Just like arrays, elements are stored adjacent to each other. So adding or removing an item in the middle of the array requires "scooting over" other elements, which takes O(n) time.

Average Case

O(n)

O(1)

O(1)

O(n)

O(n)

Fast lookups. Just like arrays, retrieving the element at a given index takes O(1)

Worst Case

O(n)

O(1)

O(n)

O(n)

O(n)

Python 2.7

gas_prices.append(354)

Size vs. Capacity

10.

In Python 3.6

Here's what they look like:

gas_prices.append(346)

gas_prices.append(360)

gas_prices = []

In Python, dynamic arrays are called lists.

size: 4 а

> capacity: 10

Doubling Appends

program.

When you allocate a dynamic array, your dynamic array implementation makes an

underlying fixed-size array. The starting size depends on the implementation—let's say

this point, our dynamic array has a length of 4. But the underlying array has a length of

our implementation uses 10 indices. Now say we append 4 items to our dynamic array. At

We'd say this dynamic array's **size** is 4 and its **capacity** is 10. The dynamic array stores an

end_index to keep track of where the dynamic array ends and the extra capacity begins.

D

end_index

To make room, dynamic arrays automatically make a new, bigger underlying array. Usually twice as big. Why not just extend the existing array? Because that memory might already be taken by another

D

а

M 0 t

old array

new array

Each item has to be individually copied into the new array.

What if we try to append an item but our array's capacity is already full?

Spotify D

time.

- **Amortized cost of appending** 1. The time cost of each special O(n) "doubling append" doubles each time. 2. At the same time, the number of O(1) appends you get until the next doubling

Copying each item over costs O(n) time! So whenever appending an item to our dynamic

array forces us to make a new double-size underlying array, that append takes O(n) time.

That's the worst case. But in the best case (and the average case), appends are just O(1)

append also doubles. These two things sort of "cancel out," and we can say each append has an average cost or amortized cost of O(1).

Given this, in industry we usually wave our hands and say dynamic arrays have a time cost of O(1) for appends, even though strictly speaking that's only true for the average case or the amortized cost. in Share **f** Share Tweet course home

Next up: Merging Meeting Times → Subscribe to our weekly question email list » Programming interview questions by company: Google interview questions Facebook interview questions

 Amazon interview questions Uber interview questions Microsoft interview questions Apple interview questions Netflix interview questions Dropbox interview questions eBay interview questions LinkedIn interview questions Oracle interview questions PayPal interview questions Yahoo interview questions Programming interview questions by topic: SQL interview questions Testing and QA interview questions Bit manipulation interview questions Java interview questions Python interview questions

> Copyright © 2021 Cake Labs, Inc. All rights reserved. 228 Park Ave S #82632, New York, NY US 10003 (862) 294-2956 About | Privacy | Terms

Ruby interview questions

C++ interview questions

Swift interview questions

PHP interview questions

C# interview questions

C interview questions

JavaScript interview questions

Objective-C interview questions