FullStack.Cafe - Kill Your Tech Interview

Q1: Explain what is an Array? ☆

Topics: Arrays

Answer:

An **array** is a collection of **homogeneous** (same type) data items stored in **contiguous memory** locations. For example if an array is of type “int”, it can only store integer elements and cannot allow the elements of other types such as double, float, char etc. The elements of an array are accessed by using an **index**.

- $O(1)$
- $O(log\ n)$
- $O(n)$
- $O(n\ log\ n)$
- $O(n^2)$
- $O(2^n)$
- $O(n!)$

Q2: Name some characteristics of Array Data Structure ☆

Topics: Arrays Data Structures

Answer:

Arrays are:

- **Finite (fixed-size)** - An array is finite because it contains only limited number of elements.
- **Order** -All the elements are stored one by one , in contiguous location of computer memory in a linear order and fashion
- **Homogenous** - All the elements of an array are of same data types only and hence it is termed as collection of homogenous

Q3: What is time complexity of basic Array operations? ☆☆

Topics: Arrays

Answer:

Array uses continuous memory locations (**space** complexity `O(n)`) to store the element so **retrieving** of any element will take `O(1)` time complexity (constant time by using index of the retrieved element). `O(1)` describes **inserting** at the end of the array. However, if you're inserting into the middle of an array, you have to shift all the elements after that element, so the complexity for insertion in that case is `O(n)` for arrays. End appending also discounts the case where you'd have to resize an array if it's full.

Operation	Average Case	Worst Case
Read	`O(1)`	`O(1)`
Append	`O(1)`	`O(1)`

Insert	`O(n)`	`O(n)`
Delete	`O(n)`	`O(n)`
Search	`O(n)`	`O(n)`

Q4: What are Dynamic Arrays? ☆☆

Topics: Arrays Data Structures

Answer:

A **dynamic array** is an array with a big improvement: *automatic resizing*.

One limitation of arrays is that they're *fixed* size, meaning you need to specify the number of elements your array will hold ahead of time. A dynamic array expands as you add more elements. So you don't need to determine the size ahead of time.

Q5: Name some advantages and disadvantages of Arrays ☆☆

Topics: Arrays

Answer:

Pros:

- **Fast lookups**. Retrieving the element at a given index takes `O(1)` time, regardless of the length of the array.
- **Fast appends**. Adding a new element at the end of the array takes `O(1)` time.

Cons:

- **Fixed size**. You need to specify how many elements you're going to store in your array ahead of time.
- **Costly inserts and deletes**. You have to shift the other elements to fill in or close gaps, which takes worst-case `O(n)` time.

Q6: How do Dynamic Arrays work? ☆☆

Topics: Arrays

Answer:

A simple dynamic array can be constructed by *allocating an array of fixed-size*, typically *larger* than the number of elements immediately required. The elements of the dynamic array are stored contiguously at the start of the underlying array, and the remaining positions towards the end of the underlying array are reserved, or unused. Elements can be added at the end of a dynamic array in constant time by using the reserved space until this space is completely consumed.

When all space is consumed, and an additional element is to be added, the underlying fixed-sized array needs to be increased in size. Typically resizing is expensive because you have to allocate a bigger array and copy over all of the elements from the array you have overgrow before we can finally append our item.

Dynamic arrays memory allocation is language specific. For example in C++ arrays are created on the stack, and have automatic storage duration -- you don't need to manually manage memory, but they get destroyed when

the function they're in ends. They necessarily have a fixed size:

```
int foo[10];
```

Arrays created with operator `new[]` have *dynamic* storage duration and are stored on the heap (technically the "free store"). They can have any size, but you need to allocate and free them yourself since they're not part of the stack frame:

```
int* foo = new int[10];
delete[] foo;
```

Q7: What is a main difference between an Array and a Dictionary?

☆☆

Topics: Arrays

Answer:

Arrays and dictionaries both store collections of data, but differ by *how they are accessed*. Arrays provide *random access* of a sequential set of data. Dictionaries (or associative arrays) provide a *map* from a *set of keys* to a *set of values*.

- Arrays store a set of objects (that can be accessed randomly)
- Dictionaries store pairs of objects
- Items in an array are accessed by position (*index*) (often a number) and hence have an order.
- Items in a dictionary are accessed by *key* and are unordered.

This makes array/lists more suitable when you have a group of objects in a set (prime numbers, colors, students, etc.). Dictionaries are better suited for showing relationships between a pair of objects.