Algorithms and Data Structures

Introduction to Data Structures

Part I: Data Structures

Agenda

- What is a Data Structure?
- Why do we need to learn Data Structures?
- How do we learn Data Structures?

What is a Data Structure

- A Data Structure (DS) is a way of organizing data so that it can be used efficiently.
  - Dictionary - alphabet order.
  - Yellow-book - similar alphabet order with categories.
  - Map - geometry information.
  - Textbook:
    - List of chapters and their pages - content table.
    - List of terms / index and their pages - index table.
  - Moving - using moving boxes to put small / similar things together.

Why Data Structures

- DS help to manage and organize data.
- DS play important role in creating fast and powerful algorithms.
- DS help to make your programs easier to be understood.
- How Best programmers differentiate them with beginners?
  - Always use the right DS on the right scenario.
- In a career path, SE (Software engineer) will be advanced to Software Architect
  - SE: code on existing DS.
  - SA: choose the DS for SE to code on.

Abstract Data Type

- An Abstract Data Type (ADT) is an abstraction of a data structure that provides only the interface to which a data structure must adhere to.
- The interface only defines data and operations but does not give any specific details about:
  - how something should be implemented;
  - in what programming language;
- How to get to Washington.D.C?
  - By plane -> which company? -> which flight?
  - By car -> what type of car? -> which route?
  - By bicycle -> which type of bike? -> which route?
  - By walk -> emmm...

What are we going to learn?

- ADTs:
  - Arrays / LinkedList / Stack / Queue
  - Tree / Set / Dictionary / Map
  - Graph
- For each ADT, we are going to examine:
  - Logic view
  - Operations
  - Cost of operations
  - Implementation in Python

Part II: Array

What is an Array?

- Most of time, when we talk about Arrays, we talk about Static Arrays.
- A Static Array is a fixed length container containing n elements indexable from the range [0, to n - 1].
- Indexable means each element/slot/index in the array can be referenced with a number.
- Python has no build-in Array data structure.
- Python's build-in Array-like data structure called Lists (and actually Dynamic Arrays)

When is an Array used?

- Storing and accessing sequential data;
- Temporarily storing objects;
- Used by IO routine such as buffers;
- Used to return multiple values by a function;
- Used in Dynamic Programming to cache the answers to subproblems.

Example

- Let's initiate an integer array:
  - arr.length = 10;
  - max capability for arr is 10;
  - index from 0 to 9
  - arr[5] = 56
  - arr[9] = 9
  - arr[-1]? arr[10]? -> index out of bounds

Complexity of Static Arrays

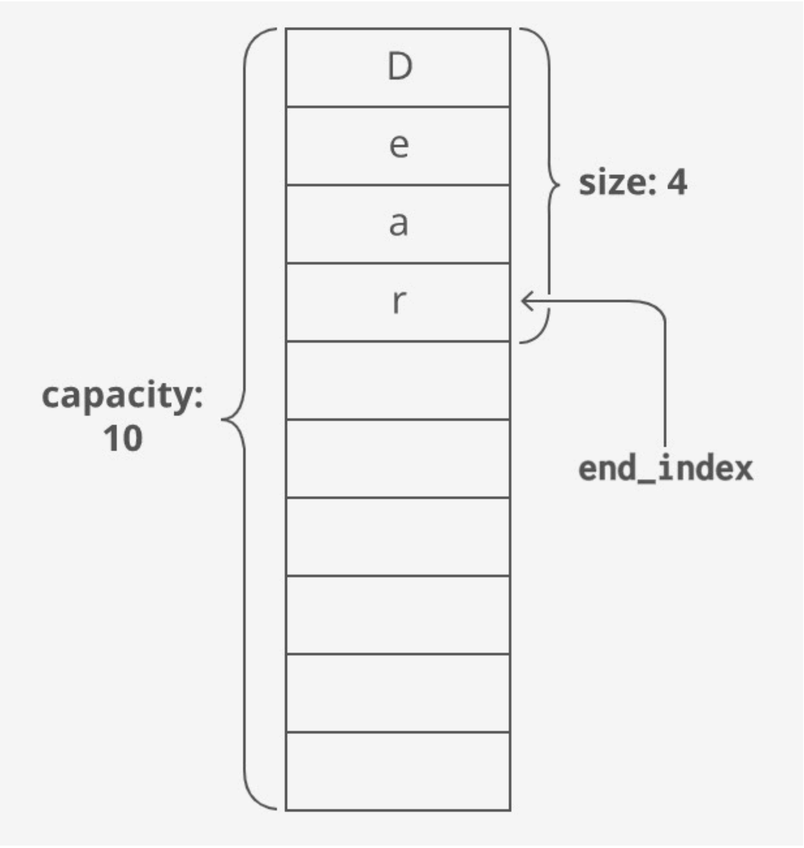
- Access
- Search
- Insertion (random, append, prepend)? Can we?
- Deletion (random)? Can we?

Static VS Dynamic

- Static Arrays:
  - Fixed size array,
  - no operations such as insertion, deletion, etc
- Dynamic Arrays:
  - size varies based on data usage
  - allow insertion and deletion
  - efficient in space complexity

How?

- Dynamic Arrays use Static Arrays as underneath data structure.
- When a Dynamic Array is initiated (without size), a static array with a base-size is created.
  - the base-size is decided by the prediction.
  - a reasonable number that represents average usage.
  - two properties: capability (the size of the static array) and size (the actually used slots)

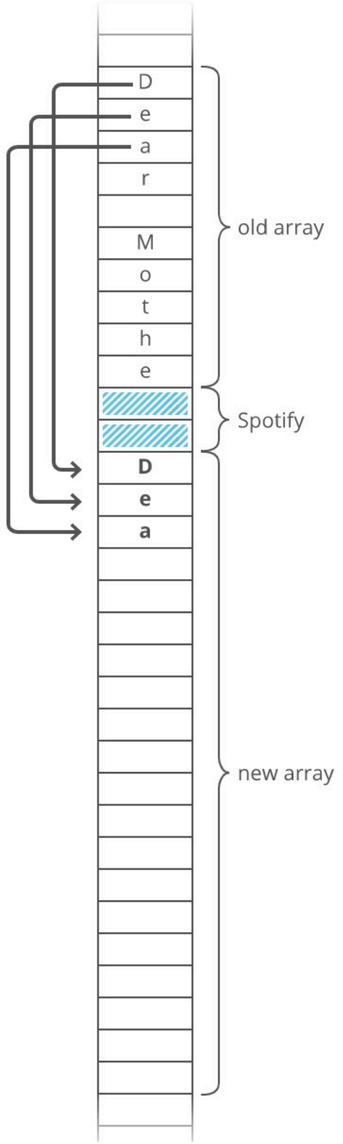


Operations

- Access
- Search
- Insertion (think carefully if the underlying static array is full already, say, size == capability)

Double Append Situation

- Append
- Prepend
- Random Insertion
- Deletion
  - Half usage situation



Complexity Summary

- Append
- Prepend
- Random Insertion
- Deletion
  - Half usage situation

Operation	Static Arrays	Dynamic Arrays
Access	O(1)	O(1)
Search	O(n)	O(n)
Append	N/A	O(1)
Prepend	N/A	O(n)
Insertion	N/A	O(n)
Deletion	N/A	O(n)

Exercise

- Describe the following operations (if applicable) for a static array and a dynamic array:
  - create a size 10 array “num”
  - fill num with some integers.
  - access index 5, 9
  - search for number 4
  - fill num until it is full (hint: static array can only handle until this)
  - append “1” into num
  - append “1” into num
  - insert “0” after the 5th element
  - prepend “2” into num
  - delete first element

Challenges

- How to find the missing number in integer array?
  - Given an integer array which contains n-1 different numbers from 1 to n (one number is missing), you need to write a program to find that missing number in an array.
  - For example:
    - Input: [3, 4, 5, 1]
    - Output: 2

Challenges

- How to find duplicate number on Integer array?
  - Given an integer array which contains n numbers ranging from 1 to n-1. There is exactly one number is repeated in the array.

Challenges

- How to find all pairs on integer array whose sum is equal to given number?
  - Given an integer array and a number, you need to write a program to find all elements in the array whose sum is equal to the given number. Remember, the array may contain both positive and negative numbers, so your solution should consider that.

Challenges

- Write a program to find k^{th} smallest element in unsorted array.
  - Given an unsorted array of numbers and k, you need to find the k^{th} smallest number in the array.
  - For example if given array is {1, 2, 3, 9, 4} and k=2 then you need to find the 2^{nd} smallest number in the array, which is 2.
  - One way to solve this problem is to sort the array in ascending order then pick the $(k-1)^{th}$ element, that would be your k^{th} smallest number in array because array index starts at zero, but can you do better?

Challenges

- Write a program to find common elements in three sorted array?
  - Given three arrays sorted in non-decreasing order, print all common elements in these arrays.
  - Examples:
    - input1 = {1, 5, 10, 20, 40, 80}
    - input2 = {6, 7, 20, 80, 100}
    - input3 = {3, 4, 15, 20, 30, 70, 80, 120}
    - Output: 20, 80

Challenges

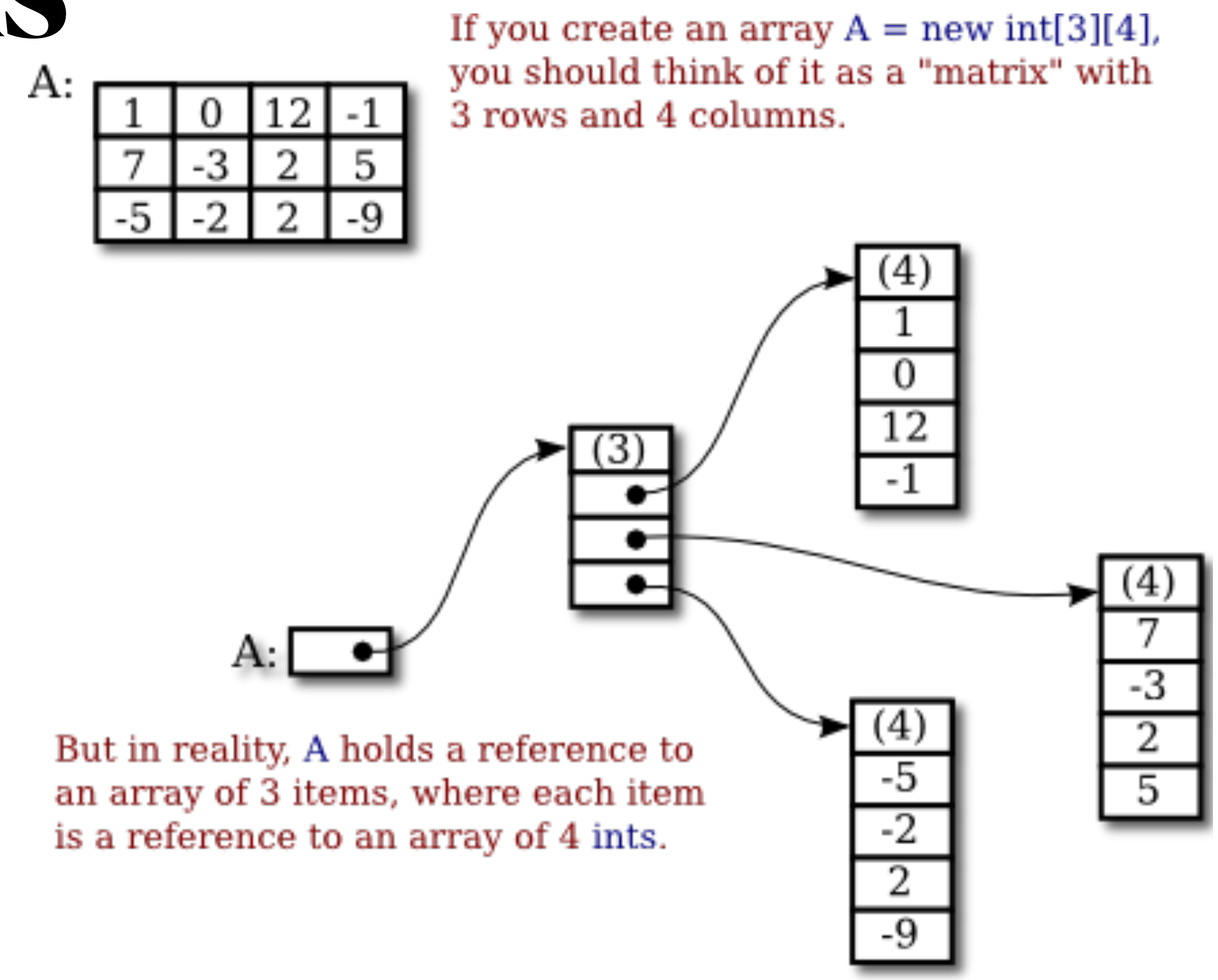
- Write a program to find intersection of two sorted arrays.
  - Your task is to write a method return the intersection of two sorted arrays. For example, if the two sorted arrays as input are {21, 22, 34, 35, 41} and {61, 45, 34, 21, 11}, it should return an intersection array with numbers {34, 21}, For the sake of this problem you can assume that numbers in each integer array are unique.

What is a Matrix?

- A Matrix is a special two dimension array:
  - The first dimension corresponds to n, or r; and we call them rows.
  - The second dimension corresponds to m or c; and we call them columns.
  - Elements on first dimension are actually references to the second dimension.
  - Double for loop is natural for matrix operations.

How Matrix Works

- A Matrix is a special two dimension array:
  - The first dimension corresponds to n;
  - The second dimension corresponds to m;
  - Elements on first dimension are references to the second dimension

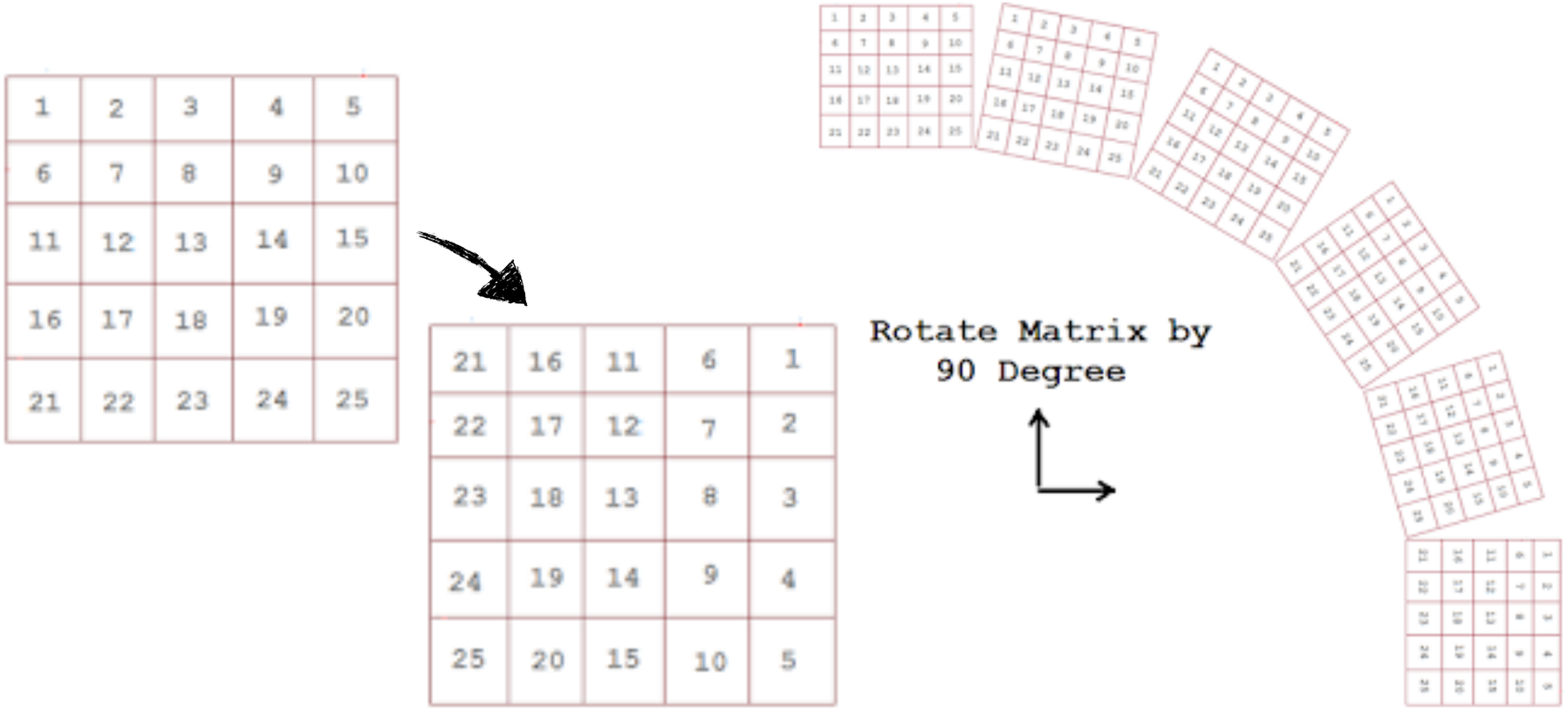


Matrix Manipulation

- Transpose
  - change a matrix A from n*m to m*n
- Calculate sum of a row r
  - summarize A[r][0,1,...,m-1]
- Calculate sum of a column c
  - summarize A[0,1,...,n-1][c]
- Calculate sum of diagonals (n == m should be true)
  - summarize A[i][i] and A[i][n-i-1], where i = 0, 1, ...n-1
- Addition, Subtraction, Multiplication

Challenges

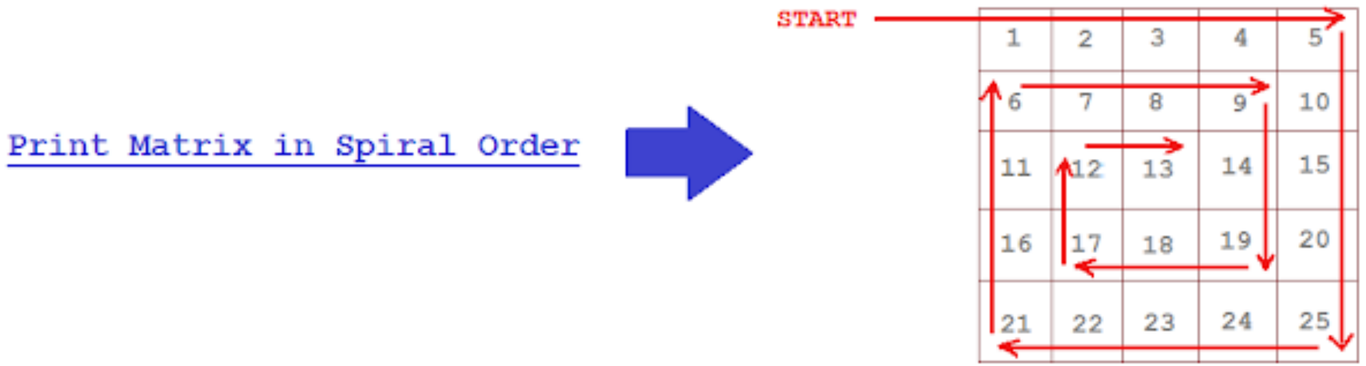
- Rotate a matrix by 90 degree. (Given N*M Matrix, Rotate it by 90 degrees



Challenges

- Print a matrix in Spiral form. (Given a Matrix(2D array), print it in spiral form.)

Print Matrix in Spiral Order



1, 2, 3, 4, 5, 10, 15, 20, 25, 24, 23, 22, 21, 16, 11, 6, 7, 8, 9, 14, 19, 18, 17, 12, 13

Challenges

- Search in a row wise and column wise sorted matrix

Input: Search 10

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Output: True(Element found)

Input: Search 18

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Output: False(Element not found)

Thank you!