Competitive Programming

From Problem 2 Solution in O(1)

Data Structures: What and Why

Mostafa Saad Ibrahim
PhD Student @ Simon Fraser University

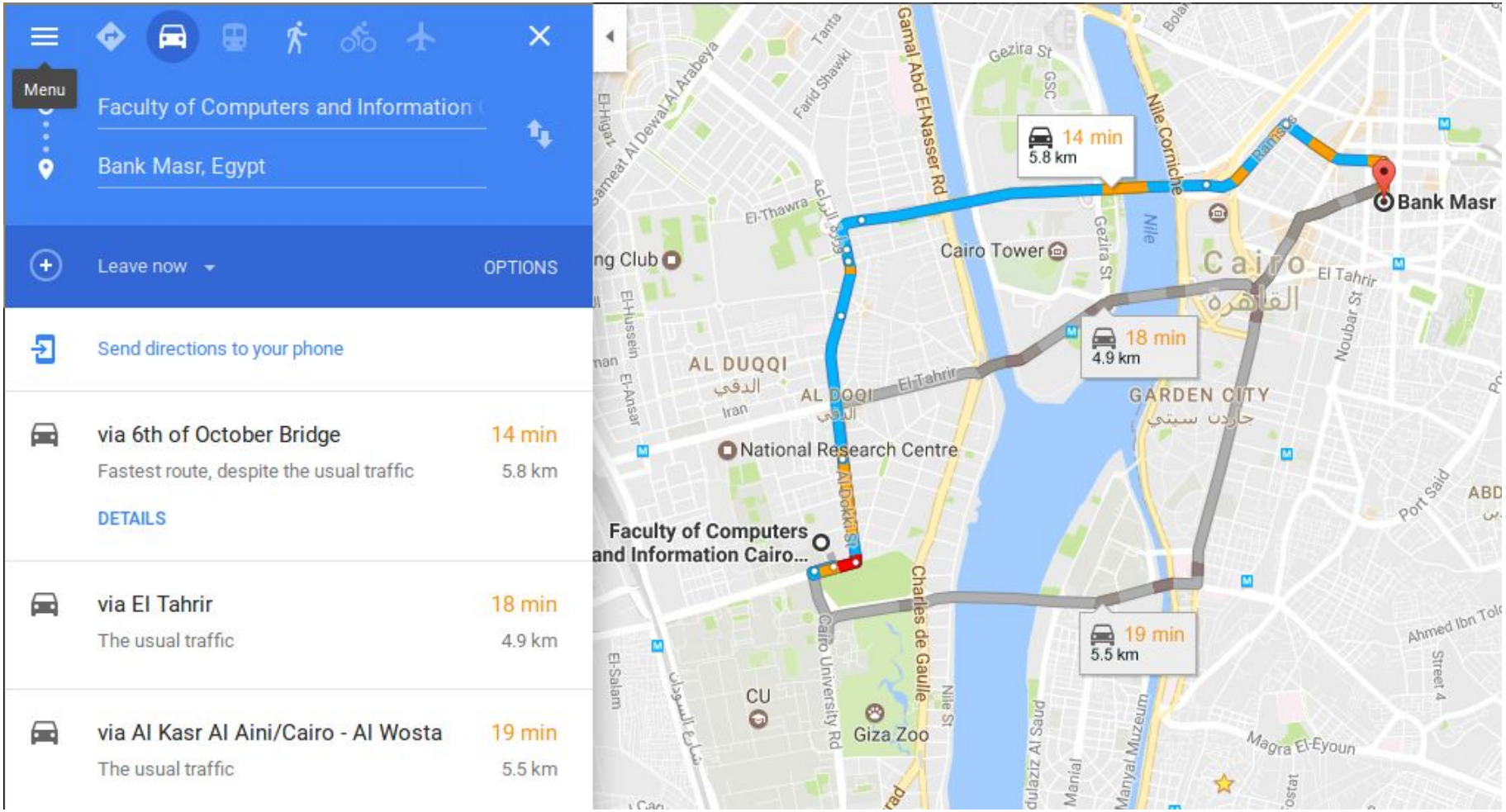
What is a data structure?

- Any computer software deals with **data**
  - **Organize** …. **Store** …. **Process** …. **Use** it!
  - Facebook data: Your account details (name, email, password), your friends list, your images and videos, your posts (replies, likes), Ads, groups, attached files … etc
  - Dictionary: There are many thousands of words? How to store the words? Sorted? In an array? What about common prefix between words (**auto**pilot, **auto**biography, **auto**mobile)? Search for a word that starts with "univ"?
  - Google maps: I want to go From School to Home? From Cairo to London? How to store these locations/path ? How to efficiently find the path between 2 points?

What is a data structure?

- A data structure is a specialized **format** for **organizing** and **storing** data
  - We don't want only to store data, we need to process it **efficiently**!
  - Suggest K friends to Aly that are of 2 friends apart. But we have millions of users?
  - As **algorithms** field is all about **efficiency**, Data Structures and Algorithms are highly related!
  - Please watch: Algorithms - What and Why

Why data structures?



Why data structures?

- Think about the **google maps** example?
  - We have lots of details about the locations and paths
  - Variety of transportation options
  - Transportations availability vs date/time
  - Roads that get closed temporarily
  - New added roads. Changed roads. Removed roads
  - Lots of data
  - Lots of requested operations over them!
  - We need something to wrap all of that!

Why data structures?

- Organize the data for specific purposes
  - From a purpose to another, we may arrange data in different ways
  - That is, it is NOT one way to implement a data structure
  - When we have much data (Millions of users on facebook), things become much more complex and critical!
  - E.g. Search engines for scientific or social purposes
  - E.g. If a storm hits specific list of locations, how many homes will face a power outage?
  - E.g. We are in a war, and want to destroy the minimum number of bridges in a city to disconnect 2 points of our enemy? Rockets are expensive!

Data inside a data structure?

- Primitive Data Types
  - Basic variables (Your age in facebook account)
  - Arrays (Array of your **posts**)
- Other Data Structures (DS)
  - So a data structure may contains other DSes
  - Facebook Account Data Structure has also
    - Post Data Structure
    - Album Data Structure

Data inside a data structure?

- What is inside the Facebook account DS?
  - Your Facebook ID (An **integer**)
  - Your Name (**Array of chars**)
  - Your Email / Address / Password (Each is Array of chars)
  - Your Age (integer)
  - Your Albums (Array of **Albums**)
    - What is an album! Array of photos!
    - What is photo? Date, Location and Image
    - What is an image! 2D Array of pixels!
    - A pixel? 3 integers (red, green, blue) = **primitives**
  - Your Posts (Array of **Posts**)
    - What is a post!

Data inside a data structure?

- What is inside the post DS?
  - Post text (Array of chars)
  - Replies on post (Array of replies)
    - What is a reply? Author Facebook Account ID + Array of chars for his text reply?
    - But this is sequential array of replies! Very **simple**.
    - What if we want to allow **reply on reply on reply**?
    - More **complex way** of organizing your data needed!
- Two lessons
  - Complex functionalities = Complex Data Structures
  - In the end of a data structure setup = must be **primitives**

The common data structures

- From a project to another!
  - People noticed some data structures that **repeat** much
  - So, instead of each one keep **re**implementing them
  - They are **already** implemented for you to use
  - Hence, in real life projects, you build **more complex** data structures **based** on the common data structures
  - In a data structure **course**, we study these common data structures
  - Through this study, we learn how to **build** a data structure
  - Hence, we learn how to **use** them the commons properly
  - **Why learn them if already implemented**? Read above again slowely.

The built-in data structures

- Programming languages
  - As mentioned, the common ones are already coded
  - In C++, **STL** library provides its data structures
  - Java and C# name them: **Collections**
- Use first and learn later approach
  - One approach to make studying DS easily, is to learn using them initially. See STL Vids
  - But then, some students decide to escape learning them well, claiming no need for that. This is a fatal mistake
  - You may not be able to build professional data structures or even use the built-in one properly without proper study

Basic Data structures

- Basic ones
  - Basic Data structures are usually based on easy logic
  - Hence, overall data and operations are easy
  - These are very good from teaching perspectives
  - We also use them much in reality
  - Examples: **Lists, Stack, Queues and Trees**
  - Queue Data Structure: You in restaurant and people come to ask for food orders. People have a fair concept **First come First Out (FIFO).** A queue data structure just implement that concepts. It supports 2 basic operations: Enqueue (In end of list) and Dequeue (out from list head). In programming, operating system **jobs** need a queue

Advanced Data structures

- Advanced ones
  - These were more critical or complex scenarios are needed
  - Popular ones in courses are **Heaps, Hash tables and Red Black trees**
  - Ones that typically are not explained include *Segment Tree, Binary Indexed Trees, Trie, Suffix Arrays, Treaps*
  - Hash Table: Arrays uses key as integer. What if want the index be another data structure? or a string? Hashtable does this magic. One way is by converting a complex input to a single integer index and use it to return the target object.

Data Structure Efficiency

- Operations (methods) in your data structure
  - Assume we have N (10000) employees
  - Is a loop over these N items like 3 nested loops?
  - No, it seems like 10000 operations vs 10^12 operations!
  - Seems the first is efficient, but the 2nd is not!
  - So how to measure the efficiency of a function?
  - The **complexity (asymptotic)** analysis in the **algorithms** field answers that.
  - Can one data organizing forces us to write efficient code, while other data organizing allows a faster one?!

Data Structures tradeoffs

- Tradeoffs
  - As mentioned, one might implement **several** data structures over the **same** data
  - However, one data structure might make some operations so fast, while others are so slow
  - And another one might reverse that
  - One must understand what is requested to achieve a proper tradeoffs
  - Your deep understanding for the requirements + your design skills are the **key**
  - Don’t be a dummy engineer who implements whatever comes to mind!

Data Structures tradeoffs

- Tradeoffs
  - Sometimes, there are only specific available **resources**
  - An efficient solution, might respect these resources constraints
  - Specifically, we have 2 measures: **Time and Memory**
  - A **real time** application usually emphasize the time concern (Think in games)
  - **Mobiles** forces **memory** constraints relative to servers. So, one might design memory efficient solution, but slow processing
  - *Your feelings when the browser consume your machine available ram? or slow when processing large data?*

Logical and Physical Views

- What is an email service?
  - An online address (user@service.provider.com)
  - You create the account (the address)
  - Then, you can send/receive emails
  - This is a logical view (thought / abstract / interface).
- What are email service providers?
  - They providers **implement** such service in their own ways
  - Examples; Gmail, Yahoo, Hotmail
  - Many people might be fan of the implementation efficiency of gmail (mostafa.saad.fci@gmail.com)
  - This is a physical view (reality).

Logical and Physical Views

- What is a queue?
  - A data structure to enqueue and dequeue items respecting FIFO principle
  - This is a logical view (Abstract Data Type)
- **How** a queue is **implemented**?
  - Using an array internally of items
  - Index to know where we are
  - Add element put in the array, index++
  - Remove element, do index--
  - Can we implement in different ways? SURE
  - *List ADT might be implemented using linked list or array-based list*

Abstract data type (ADT)

- **Data abstraction** is a programming (and design) technique that relies on the separation of interface (what) and implementation(How).
- ADT Interface
  - Basic Data elements (E.g. Queue item)
  - Operations list (name, input, output)
- Data Structure implements ADT
  - Its own variables to implement the functionality
  - Methods details (specific time/memory efficiency)
  - **Submthods** that main methods need it

Books

- Books
  - Most of the books will be ok
  - Some books are pure data structures
  - While others will mix it with algorithms.
  - Following are some **free** available books (not neccessirily the best)
  - Open Data Structures (C++)
  - Data Structures and Algorithm Analysis (Java)
  - Algorithms and Data Structures The Basic Toolbox
  - Notes on Data Structures and Programming
  - See this answer

تم بحمد الله

علمكم الله ما ينفعكم

ونفعكم بما تعلمتم

وزادكم علماً