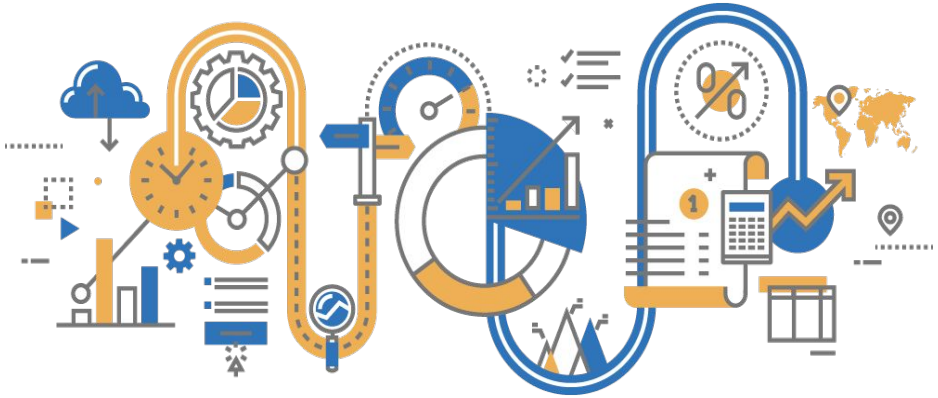
Unravelling data structures, and their applications for the greater good



Abdul Wasay,
25th April, 2020

What are data structures?

“Bad programmers worry about the code.

Good programmers worry about
data structures and their relationships.”

- Linus Torvalds

What we will be covering today?

1. Arrays and Linked lists
2. Stacks
3. Queues
4. Hash maps / Hash tables
5. Trees
6. Graphs

Arrays

- O(1) access time due to indexing
- Memory blocks allocated are contiguous
- Random access data structures
- Static structures, cannot be easily extended or reduced to fit the dataset
- Expensive to maintain new insertions and deletions
- You should know the size of your array beforehand

Linked lists

- No indexing :(
- No need for contiguous memory blocks
- Sequential access data structures
- Dynamic data structure, can grow and shrink without much overheads
- Types:
  - Singly linked lists
  - Doubly linked lists
  - Circular linked lists

Linked lists: Applications

1. Large number calculations
2. Graph representations in Adjacency lists
3. Chaining in hash tables to resolve hash collisions

Stacks

- Last-in first-out principle
- Basic operations:
  - push(X)
  - pop() -> X
- Applications:
  - Activation records (hence recursion support too)
  - Backtracking (or undo mechanism in text editors)
  - Shunting yard algorithm (First step in solving arithmetic expressions)

Queues

- First-in first-out principle
- Operations:
  - enqueue(X)
  - Dequeue -> X
- Applications:
  - BookmyShow used it for Global Citizen tickets
  - Scheduling

HashMaps

- O(1) retrieval
- Hashing functions
- Pretty fast and efficient
- Hash collisions happen, so be careful of those

Trees

- Non linear data structure
- Extends the concept of a linked list
- Terminology:
  - Root
  - Leaves
  - Parent-> Child
  - Height and level
- Types:
  - Binary tree
  - Binary search tree
  - AVL tree
  - Red black trees
  - N-ary tree

Trees: Applications

1. Java 8 uses balanced trees for implementing HashMaps with large number of collisions
2. Abstract syntax trees in compilers
3. Heaps

Graphs

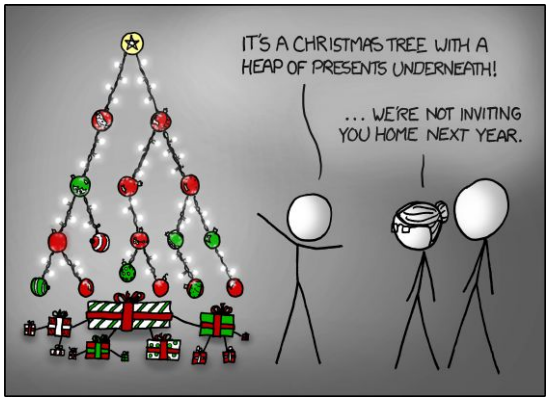
- Defined by Vertices and Edges
- Terminology:
  - Vertex
  - Edge
  - Degree of a vertex
- Representation:
  - Adjacency matrix (Terrible space complexity)
  - Adjacency lists (Better space complexity)
- Applications:
  - Social graphs (people, places and things one interacts with in the social media world)
  - Knowledge graphs
  - Of course the Google Maps platform, GPS navigations and efficient flight route calculations

A few more data structures you should explore

1. Heaps
2. Tries

Post credits - Before we wrap up

- All languages have standard libraries that either provide direct implementations of basic DSs or allow for their efficient construction. Use them!
  - Containers - C++ Reference
  - Collections Framework Overview
- Data structures form the core of all efficient programs that run out in the real world



Data dominates. If you've chosen the right data structures and organized things well, the algorithms will almost always be self-evident. Data structures, not algorithms, are central to programming.

Rob Pike

References/Sources

- https://www.cs.cmu.edu/~adamchik/15-121/lectures/Stacks%20and%20Queues/Stacks%20and%20Queues.html
- https://www.cs.cmu.edu/~clo/www/CMU/DataStructures/Lessons/lesson4_1.htm
- https://www.cs.cmu.edu/~adamchik/15-121/lectures/Linked%20Lists/linked%20lists.html
- Title credits: Chaithra
- https://www.geeksforgeeks.org/
- YK Sugi (CSDojo on YouTube)
- MIT OpenCourseware

Get in touch with me:
***abdulwasay50@gmail.com* or**
connect with me on LinkedIn

https://github.com/KnightTuring/talks-ppt