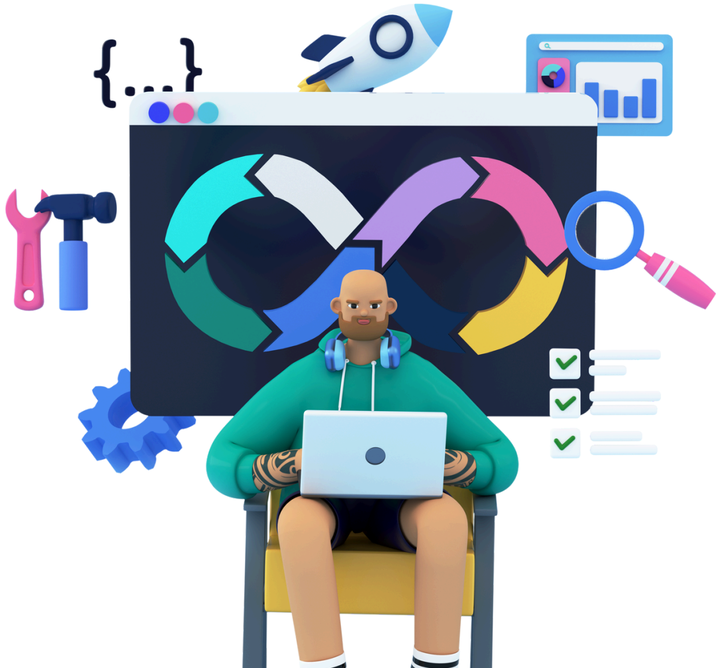
Introducing:

DS & Algo

To

DevOps Engineers

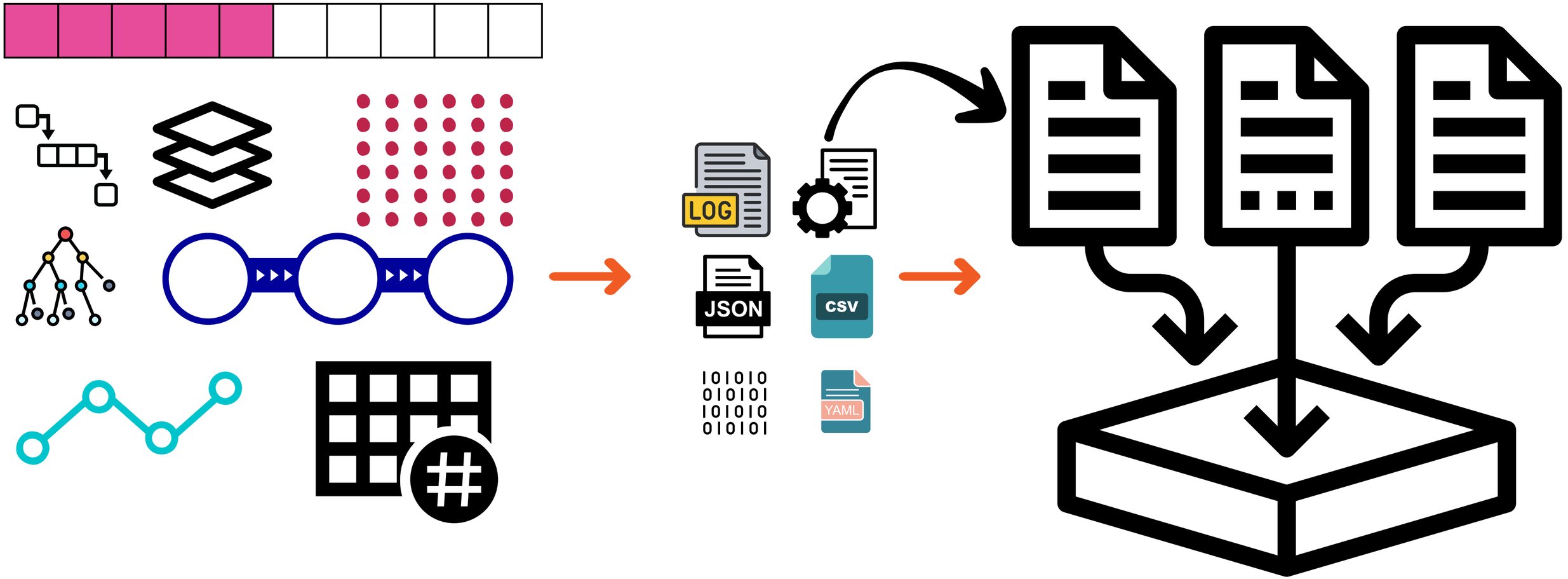
WATCH NOW



@Sandip Das

Data Structures

Data structures are ways to store and organize data in a computer so that it can be accessed and modified efficiently.




@Sandip Das

Let’s have a Look into this to understand better use-case of DS:

```
[
  {
    "timestamp": "2024-06-03T10:00:02Z",
    "level": "ERROR",
    "message": "Failed to authenticate user",
    "userId": 12345,
    "reason": "Invalid credentials",
    "ip": "192.168.1.100"
  },
  {
    "timestamp": "2024-06-03T10:00:03Z",
    "level": "FATAL",
    "message": "Critical system failure",
    "reason": "Out of memory",
    "action": "Application terminating"
  }
]
```

Let’s assume we have Multiple Log entries

This is a representation of log entries as properly structured and labeled array data, which we can loop, sort, and add/remove items easily.

Now imagine, if it were a single-line with no proper structure, e.g “ERROR: Failed to authenticate user. UserId: 12345, Reason: Invalid credentials, IP: 192.168.1.100 ”--- it would have been tough to differentiate or work with the data.

Types of Data Structures

From Development point of videw

Primitive Data Structures

Basic structures that directly operate upon machine instructions

(e.g., integers, floats, characters, pointers).

Integers
Whole numbers without a decimal point.
Example: int a = 10;

Floats
Numbers with a decimal point.
Example: float b = 10.5;

Characters
Single letters, digits, or symbols.
Example: char c = 'A';

Pointers
Variables that store the memory address of another variable.
Example: int *ptr = &a;

Non-Primitive Data Structures

These are more complex data structures built using primitive data types.

Arrays
A collection of elements of the same type stored in contiguous memory locations.
Example: int arr[5] = {1, 2, 3, 4, 5};

Linked Lists
A sequence of nodes where each node contains data and a reference to the next node.
Example: struct Node { int data; struct Node* next; };

Stacks
A collection of elements with Last In, First Out (LIFO) access.
Example: stack<int> s;

Queues
A collection of elements with First In, First Out (FIFO) access.
Example: queue<int> q;

Trees
A hierarchical structure with nodes connected by edges.
Example: struct Node { int data; struct Node* left; struct Node* right; };

Graphs
A collection of nodes connected by edges, can be directed or undirected.
Example: struct Graph { int V; vector<int> *adj; };

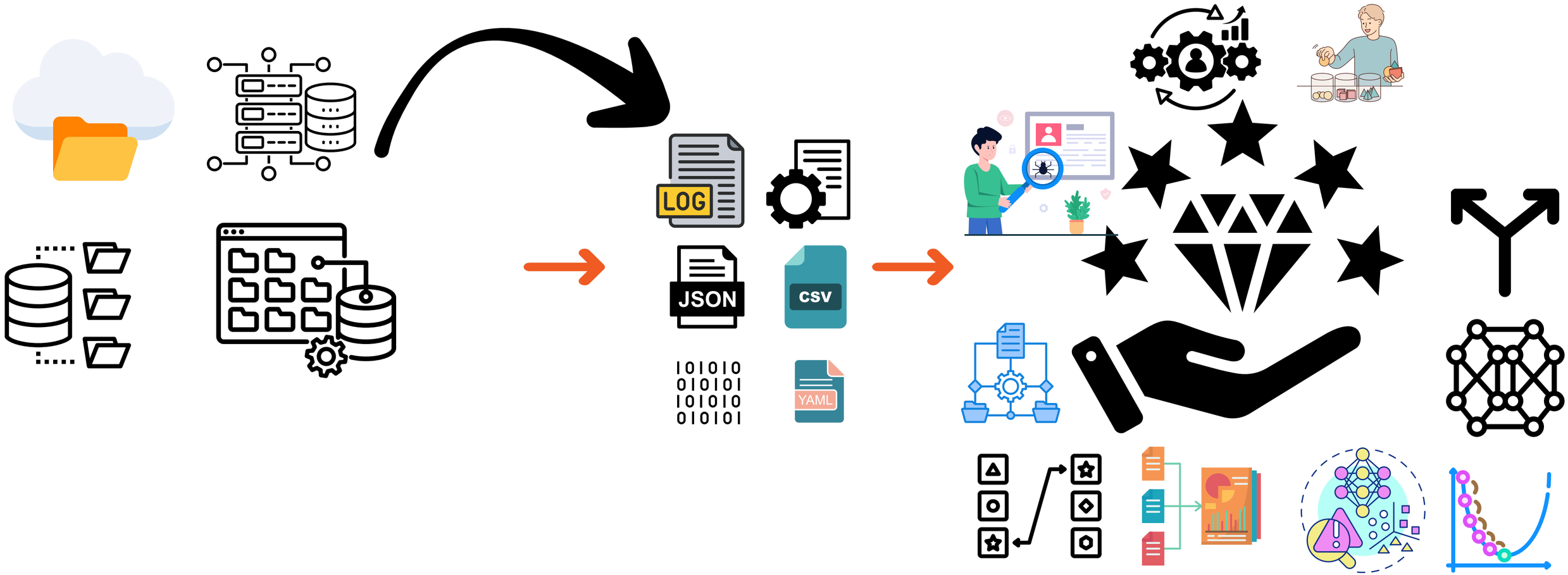
Hash Tables
A collection of key-value pairs, implemented using arrays and hash functions.
Example: unordered_map<string, int> hashTable;

Heaps
Description: A specialized tree-based structure that satisfies the heap property.
Example: priority_queue<int> maxHeap;

@Sandip Das

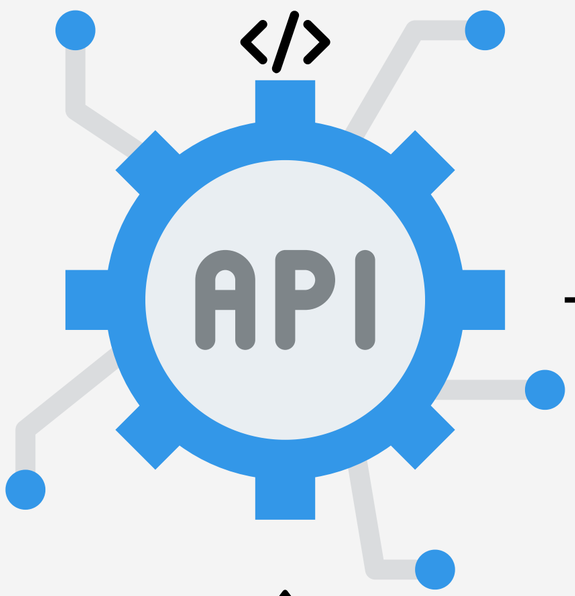
Algorithms

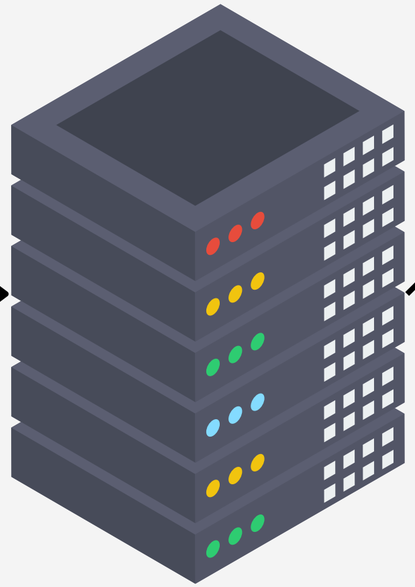
Algorithms are step-by-step procedures or formulas for solving a problem or performing a task.



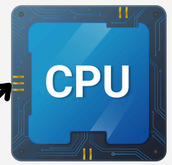
Let’s have a Look into this to understand better use-case of Algo:

@Sandip Das

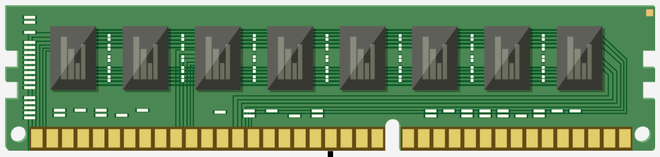




40%



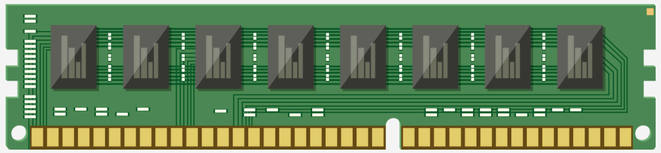
60%

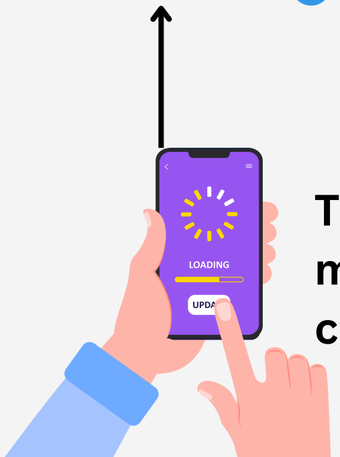


After Optimizing using Alogorithms

3%

10%





The user using the app and some how responses are slow, upon debugging found application utilizes more RAM and CPU, all ok, no memory leak, which means the code is not optimized, we can optimize codes to reduce resource use case and make it faster using suitable algorithms!

Different Types of Algorithms

From Development point of videw

@Sandip Das

1. **Sorting Algorithms**
   - **Quick Sort**: Efficient for large datasets, used in databases and file systems.
   - **Merge Sort**: Useful for sorting linked lists and large datasets, stable sorting.
   - **Heap Sort**: Used in priority queues, database query optimizations.
2. **Search Algorithms**
   - **Binary Search**: Efficient searching in sorted arrays, used in database indexing.
   - **Depth-First Search (DFS) & Breadth-First Search (BFS)**: Used in graph traversals, network analysis, and pathfinding.
3. **Graph Algorithms**
   - **Dijkstra's Algorithm**: Shortest path finding, used in routing and network optimization.
   - **Kruskal's and Prim's Algorithms**: Minimum spanning tree, used in network design and optimization.
4. **Dynamic Programming**
   - **Knapsack Problem**: Resource allocation and optimization problems.
   - **Fibonacci Sequence**: Used in various optimization and computational problems.
5. **Greedy Algorithms**
   - **Huffman Coding**: Data compression algorithms.
   - **Activity Selection**: Task scheduling and resource allocation.
6. **String Algorithms**
   - **Knuth-Morris-Pratt (KMP):** String searching, used in text editors and search engines.
   - **Rabin-Karp**: Pattern matching, used in plagiarism detection.
7. **Divide and Conquer**
   - **Matrix Multiplication**: Optimization in computational problems.
   - **Closest Pair of Points**: Used in computational geometry and graphics.

There are algo many more algorithms are there and in this series cover most of them and their practical use cases in DevOps

Thank You

@LearnTechWithSandip

SUBSCRIBE

For staying till the end

Follow on: Sandip Das