SEARCH ALGORITHMS

WITH RASPBERRY PI 4 MODEL B

CONTENT

RASPBERRY PI 4 MODEL B

- **ARM v8:** high-performance 64-bit quad-core Cortex-A72 processor
- **RAM:** 1GB / **2GB** / 4GB / 8GB
- 2.4 GHz and 5.0 GHz **wireless LAN**, **Bluetooth**, **Gigabit Ethernet**
- Micro SD card slot for loading **operating system** and data storage
- Power supply: 5V DC via USB-C connector
- Sufficient **documentation**

*“**Your tiny, dual-display, desktop computer**…and robot brains, smart home hub, media centre, networked AI core, factory controller, and much more.”* [1]

[1] https://www.raspberrypi.com/products/raspberry-pi-4-model-b/

```cpp
template <typename T>
void customBinarySearch(const vector<T>& Data, const T& searchValue, timeData& dataStruct){
    vector<double>          execTimes;
    CustomBinaryTree<T>     binaryTree;

    /* Display header */
    cout << setw(20) << right << "custom binary tree |" << setw(9) << right << "binary |";
    /* Initialize array */
    auto start = chrono::high_resolution_clock::now();
    for (const auto &each : Data) {
        binaryTree.insert(each);
    }
    auto end = chrono::high_resolution_clock::now();
    auto duration = chrono::duration_cast<chrono::nanoseconds>(end - start);
    dataStruct.insertionTime = static_cast<double>(duration.count());
    /* Measure execution times */
    for (int i = 0; i < measureN; i++) {
        start = chrono::high_resolution_clock::now();
        bool retval = binaryTree.find(searchValue);
        end = chrono::high_resolution_clock::now();
        duration = chrono::duration_cast<chrono::nanoseconds>(end - start);
        if (retval) {
            execTimes.push_back(static_cast<double>(duration.count()));
        }
    }

    /* Sort and discard largest 1/3 */
    sort(execTimes.begin(), execTimes.end());
    execTimes.erase(execTimes.end() - static_cast<int>(measureN / 3), execTimes.end());

    /* Calculate mean, set overall time and display measured times */
    dataStruct.mean = accumulate(execTimes.begin(), execTimes.end(), 0.0) / (double)execTimes.size();
    dataStruct.overall = dataStruct.insertionTime + dataStruct.mean;
    displayMeasuredTimes(dataStruct);
}
```

SEARCH ALGORITHMS

Algorithm: Set of instructions to solve a specific problem.

Search Algorithms: Designed to search for or retrieve an element from any data structure.

Categories:

(1) **Sequential Search** (linear search)
(2) **Interval Search** (binary search tree)

Data Structures:

- Array (1) (2)
- Circular array (1) (2)
- Singly linked list (1) (2)
- Doubly linked list (1) (2)
- Custom binary tree (2)
- STL binary tree (2)

IMPLEMENTATION

Backend:

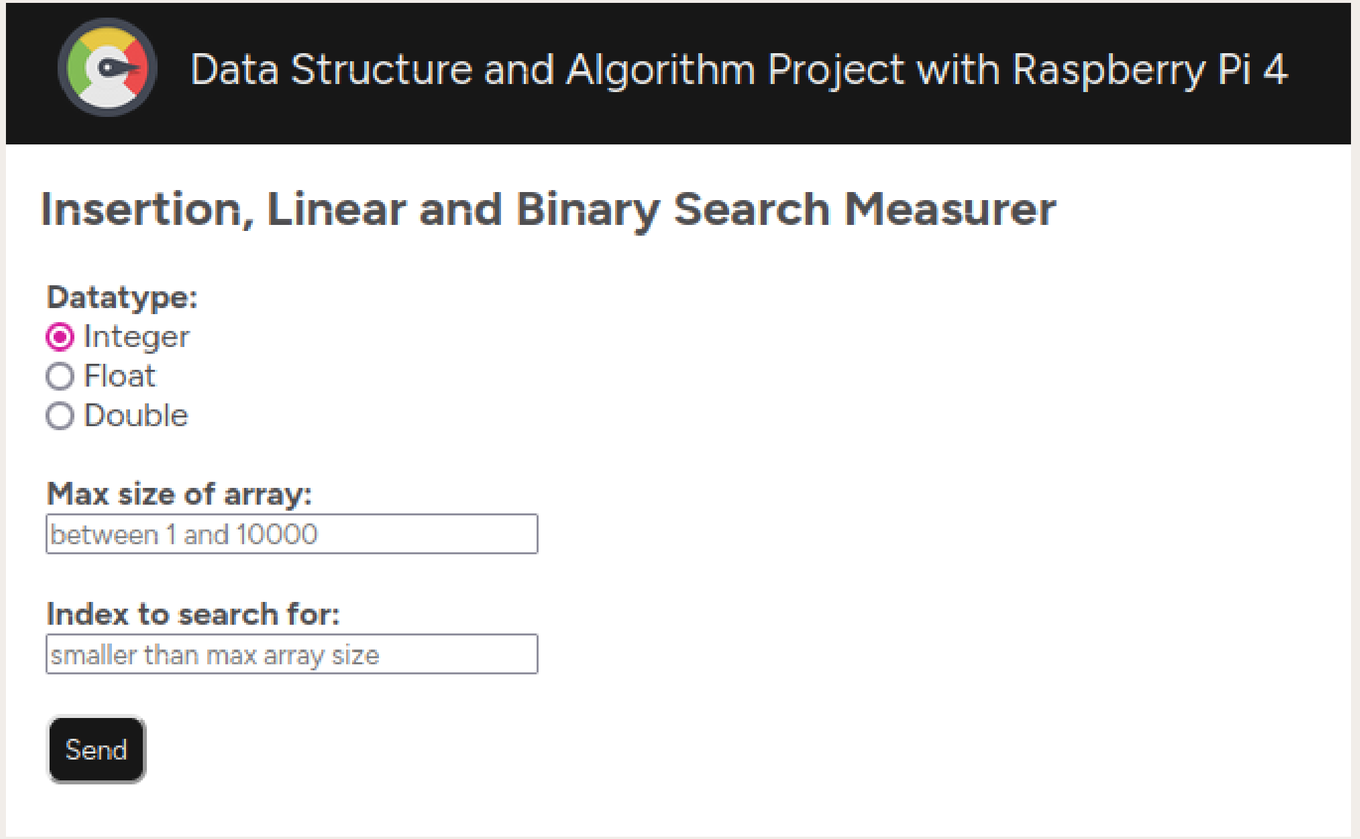
- **C++**
- **class** (templated)
- **STL**
- chrono (**high_resolution_clock**)
- Json (lohmann::json)

In between:

- HTTP requests
- **Python**
- Lighttpd (web server)
- CGI (Common Gateway Interface)

FrontEnd:

- HTML, CSS, **JavaScript**



RESULT

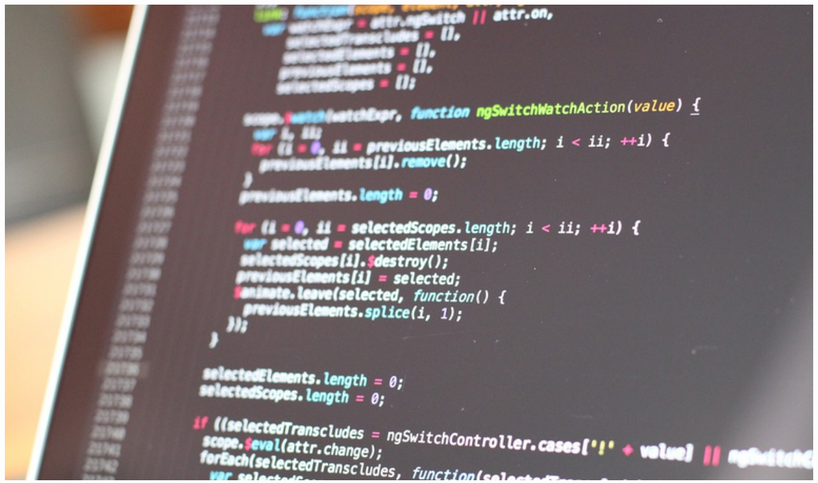
Data Structure	Search	Insertion Time	Search Time	Overall Time
array	linear	306717.0 ns	6318.4 ns	313035.5 ns
array	binary	251642976.0 ns	184.2 ns	251643160.2 ns
circular array	linear	294643.0 ns	6314.8 ns	300957.8 ns
circular array	binary	228103978.0 ns	222.3 ns	228104200.3 ns
singly linked list	linear	1093555.0 ns	7666.6 ns	1101221.6 ns
singly linked list	binary	436382301.0 ns	234194.0 ns	436616495.1 ns
doubly linked list	linear	834338.0 ns	21649.5 ns	855987.5 ns
doubly linked list	binary	505024350.0 ns	232310.0 ns	505256659.9 ns
custom binary tree	binary	5226965.0 ns	149.9 ns	5227114.9 ns
STL binary tree	binary	12601559.0 ns	628.8 ns	12602187.8 ns

ARM v8 Cortex-A72

Data Structure	Search	Insertion Time	Search Time	Overall Time
array	linear	84100.0 ns	2600.0 ns	86700.0 ns
array	binary	59142000.0 ns	85.0 ns	59142085.0 ns
circular array	linear	76800.0 ns	2400.0 ns	79200.0 ns
circular array	binary	62081400.0 ns	70.0 ns	62081470.0 ns
singly linked list	linear	627000.0 ns	2225.0 ns	629225.0 ns
singly linked list	binary	74972100.0 ns	349400.0 ns	75321500.0 ns
doubly linked list	linear	590000.0 ns	2440.0 ns	592440.0 ns
doubly linked list	binary	75120300.0 ns	350235.0 ns	75470535.0 ns
custom binary tree	binary	1776900.0 ns	120.0 ns	1777020.0 ns
STL binary tree	binary	3573000.0 ns	165.0 ns	3573165.0 ns

AMD Ryzen 7 5700U with Radeon Graphics

CHALLENGES



01 **Programming challenges:**

- bugs, bugs, bugs
- creating uniformity
- making different ends communicate

02 **Application on Raspberry Pi 4 :**

- compatibility (JSON)
- Unix

Thank you for your attention!

“Algorithms are called the aunties. They’re self-organising and so nobody fully understands them.”
— William Gibson, The Peripheral