# Assignment - 1

**Name: Dhairya Arora**

**Enrolment Number: 01616401522**

**B.Tech(IT) 4th Semester**

**2022-26**

Code:

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <unordered_map>
#include <ctime>
#include <cstdlib>
#include <chrono>
using namespace std;
using namespace std::chrono;

pair<int, double> linearSearch(const vector<int> &arr, int target)
{
    auto start_time = high_resolution_clock::now();
    for (int i = 0; i < arr.size(); ++i)
    {
        if (arr[i] == target)
        {
            auto end_time = high_resolution_clock::now();
            auto duration = duration_cast<microseconds>(end_time -
                start_time);
            return make_pair(target, duration.count());
        }
    }
    return make_pair(-1, 0.0);
}

pair<int, double> binarySearch(const vector<int> &arr, int target)
{
    auto start_time = high_resolution_clock::now();
```

```
        else if (arr[mid] < target)
        {
            low = mid + 1;
        }
        else
        {
            high = mid - 1;
        }
    }
    return make_pair(-1, 0.0);
}

pair<int, double> directSearch(const vector<int> &arr, int target)
{
    auto start_time = high_resolution_clock::now();
    unordered_map<int, int> hash_table;
    for (int num : arr)
    {
        hash_table[num] = num;
    }
    auto end_time = high_resolution_clock::now();
    if (hash_table.find(target) != hash_table.end())
    {
        auto duration = duration_cast<milliseconds>(end_time -
            start_time);
        return make_pair(target, duration.count());
    }
    else
    {
```

```
    else
    {
        return make_pair(-1, 0.0);
    }
}

vector<int> generateRandomArray(int size)
{
    vector<int> arr;
    for (int i = 1; i <= 2 * size; ++i)
    {
        arr.push_back(i);
    }
    random_shuffle(arr.begin(), arr.end());
    return arr;
}

int main()
{
    int arraySize = 100000;
    vector<int> searchValues = generateRandomArray(arraySize);
    vector<pair<int, double>> linearSearchResults;
    vector<pair<int, double>> binarySearchResults;
    vector<pair<int, double>> directSearchResults;

    for (int i = 0; i < 10; ++i)
    {
        int target = searchValues[rand() % arraySize];
```

```
    for (int i = 0; i < 10; ++i)
    {
        int target = searchValues[rand() % arraySize];

        // Linear Search
        auto linearResult = linearSearch(searchValues, target);
        linearSearchResults.push_back(linearResult);

        // Binary Search
        vector<int> sortedArray = searchValues;
        sort(sortedArray.begin(), sortedArray.end());
        auto binaryResult = binarySearch(sortedArray, target);
        binarySearchResults.push_back(binaryResult);

        // Direct Search (Hashing)
        auto directResult = directSearch(searchValues, target);
        directSearchResults.push_back(directResult);
    }

    // Printing results
    cout << "Linear Search Results:" << endl;
    cout << "(Target, Time in microseconds)" << endl;
    for (auto result : linearSearchResults)
    {
        cout << "(" << result.first << ", " << result.second << "), ";
    }
    cout << endl << "
        ----------------------------------------------------" << endl;
```

```
    cout << "Linear Search Results:" << endl;
    cout << "(Target, Time in microseconds)" << endl;
    for (auto result : linearSearchResults)
    {
        cout << "(" << result.first << ", " << result.second << "), ";
    }
    cout << endl << "
        ----------------------------------------------------" << endl;

    cout << "\nBinary Search Results:" << endl;
    cout << "(Target, Time in microseconds)" << endl;
    for (auto result : binarySearchResults)
    {
        cout << "(" << result.first << ", " << result.second << "), ";
    }
    cout << endl << "
        ----------------------------------------------------" << endl;

    cout << "\nDirect Search Results:" << endl;
    cout << "(Target, Time in milliseconds)" << endl;
    for (auto result : directSearchResults)
    {
        cout << "(" << result.first << ", " << result.second << "), ";
    }
    cout << endl;

    return 0;
}
```

## Output:

```
Linear Search Results:
(Target, Time in microseconds)
(127108, 323), (146145, 64), (104164, 11), (192168, 228), (165416, 121), (65374, 91
    ), (149001, 236), (187103, 142), (50640, 115), (106510, 372),
-----------------------------------------------------

Binary Search Results:
(Target, Time in microseconds)
(127108, 1), (146145, 1), (104164, 1), (192168, 0), (165416, 1), (65374, 1), (149001
    , 0), (187103, 0), (50640, 1), (106510, 1),
-----------------------------------------------------

Direct Search Results:
(Target, Time in milliseconds)
(127108, 131), (146145, 78), (104164, 76), (192168, 76), (165416, 101), (65374, 74),
    (149001, 78), (187103, 67), (50640, 93), (106510, 82),
```

## Graph:

- Linear Search
- Binary Search
- Direct Search