Computer Architecture Project

Youssef Abdeltawab, Suman Mallah, Tanner McGee, Farrah Omar, Quetzin Pimentel

Introduction

- In our project, we compared how fast the insertion sort algorithm runs in C++ and Python on different hardware setups.
- 8 different architectures
- Changes that were expected to be observed: how fast each machine would take and difference in efficiency between each language

C++ InsertionSort

```cpp
#include <bits/stdc++.h>
#include <chrono>
using namespace std;
using namespace chrono;

void insertionSort(vector<int>& arr) {
    int n = arr.size();
    for (int i = 1; i < n; i++) {
        int key = arr[i];
        int j = i - 1;

        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j = j - 1;
        }
        arr[j + 1] = key;
    }
}

void printArrayToFile(const vector<int>& arr, const string& filename) {
    ofstream outFile(filename);
    for (int num : arr) {
        outFile << num << endl;
    }
    outFile.close();
}

vector<int> readArrayFromFile(const string& filename) {
    ifstream inFile(filename);
    vector<int> arr;
    int temp;
    while (inFile >> temp) {
        arr.push_back(temp);
    }
    inFile.close();
    return arr;
}

int main() {
    string inputFilename = "unsorted_numbers.txt";
    string outputFilename = "sorted_numbers.txt";

    auto start = high_resolution_clock::now();

    vector<int> arr = readArrayFromFile(inputFilename);
    insertionSort(arr);
    printArrayToFile(arr, outputFilename);

    auto stop = high_resolution_clock::now();
    auto duration = duration_cast<milliseconds>(stop - start);

    cout << "Total time taken: " << duration.count() << " milliseconds\n";
    return 0;
}
```

Python InsertionSort

```python
import time

def insertion_sort(arr):
    for i in range(1, len(arr)):
        key = arr[i]
        j = i - 1
        while j >= 0 and key < arr[j]:
            arr[j + 1] = arr[j]
            j -= 1
        arr[j +1] = key

def read_array_from_file(filename):
    with open(filename, 'r') as file:
        arr = [int(line.strip()) for line in file.readlines()]
    return arr

def write_array_to_file(arr, filename):
    with open(filename, 'w') as file:
        for number in arr:
            file.write(f"{number}\n")

# Driver code
input_filename = 'unsorted_numbers.txt'
output_filename = 'sorted_numbers.txt'

start_time = time.time()

arr = read_array_from_file(input_filename)
insertion_sort(arr)
write_array_to_file(arr, output_filename)

end_time = time.time()
total_time = end_time - start_time

print(f"Total time taken: {total_time:.6f} seconds")
```

Machines Used

Quetzin (OS: macOS Sonoma 14.4.1, Processor: Apple M1 Max, 10 cores, 2.06 GHz, RAM: 16 GB, 85, 35)

Quetzin (OS: macOS Sonoma 14.4.1, Processor: Intel i7 9750H, 6 cores, 2.6 GHz, RAM: 16 GB, 92, 29)

Quetzin (OS: Windows 11, Processor: Intel i7 11700K, 8 cores, 3.6 GHz, RAM: 32 GB, 69, 10)

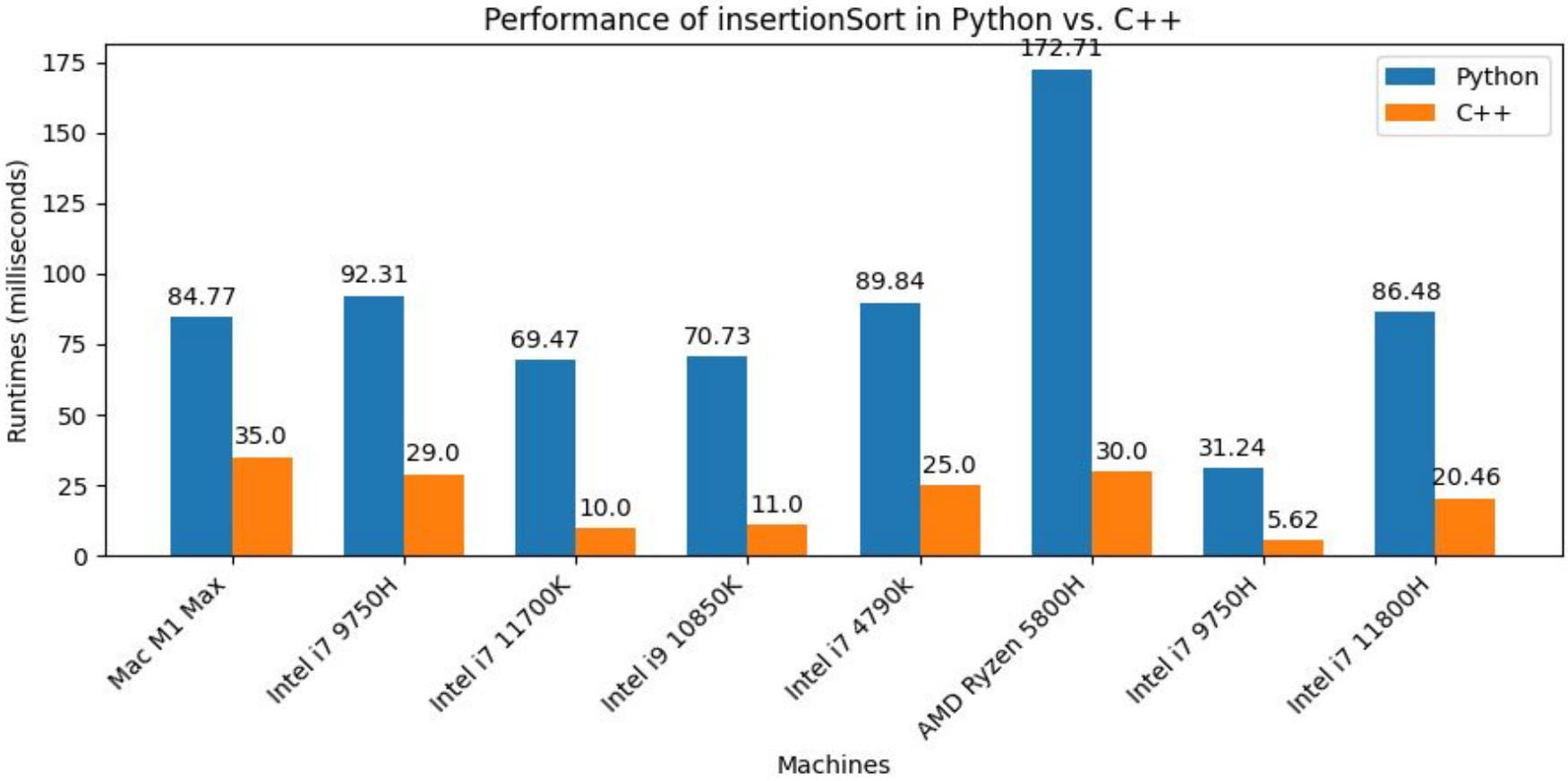
Quetzin (OS: Windows 11, Processor: Intel i9 10850K, 10 cores, 3.6 GHz, RAM: 32 GB, 70, 11)

Tanner (OS: Windows 10, Processor: Intel i7 4790k, 4 cores, 4.0 GHz, RAM: 16 GB, 89, 25)

Tanner (OS: Windows 11, Processor: AMD Ryzen 7 5800H, 8 cores, 3201 MHz, RAM: 16 GB, 172, 30)

Suman (OS: Windows 11, Processor: Intel i7 9750H, 6 cores, 2.60 GHz, RAM: 16 GB, 31, 5)

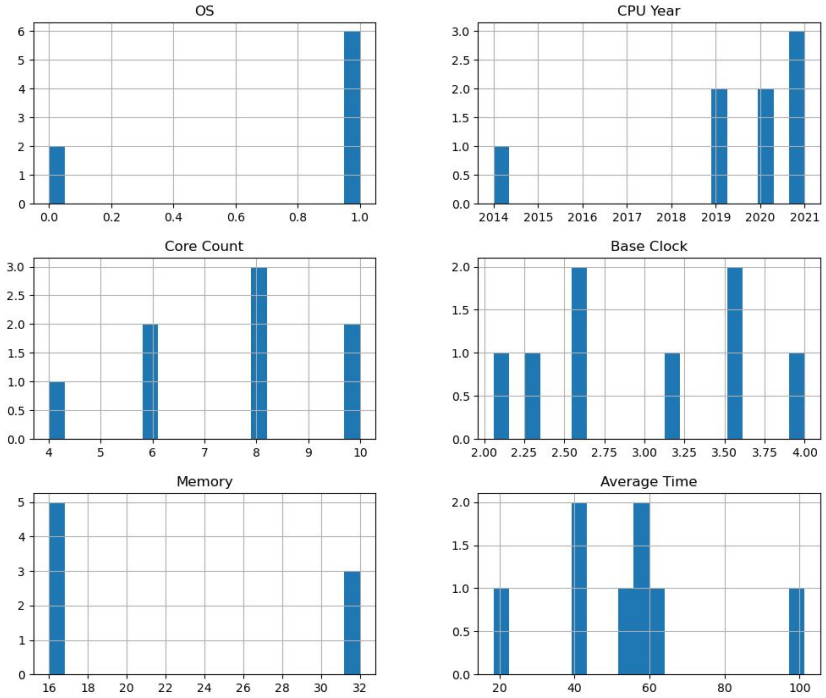
Youssef (OS: Windows 11, Processor: Intel i7 11800H, 8 cores, 2.30 GHz, RAM: 32 GB, 86, 20)


Performance of insertionSort in Python vs. C++
Runtimes (milliseconds)
0
25
50
75
100
125
150
175
Python
C++
84.77
35.0
92.31
29.0
69.47
10.0
70.73
11.0
89.84
25.0
172.71
30.0
31.24
5.62
86.48
20.46
Mac M1 Max
Intel i7 9750H
Intel i7 11700K
Intel i9 10850K
Intel i7 4790k
AMD Ryzen 5800H
Intel i7 9750H
Intel i7 11800H
Machines

Predictions

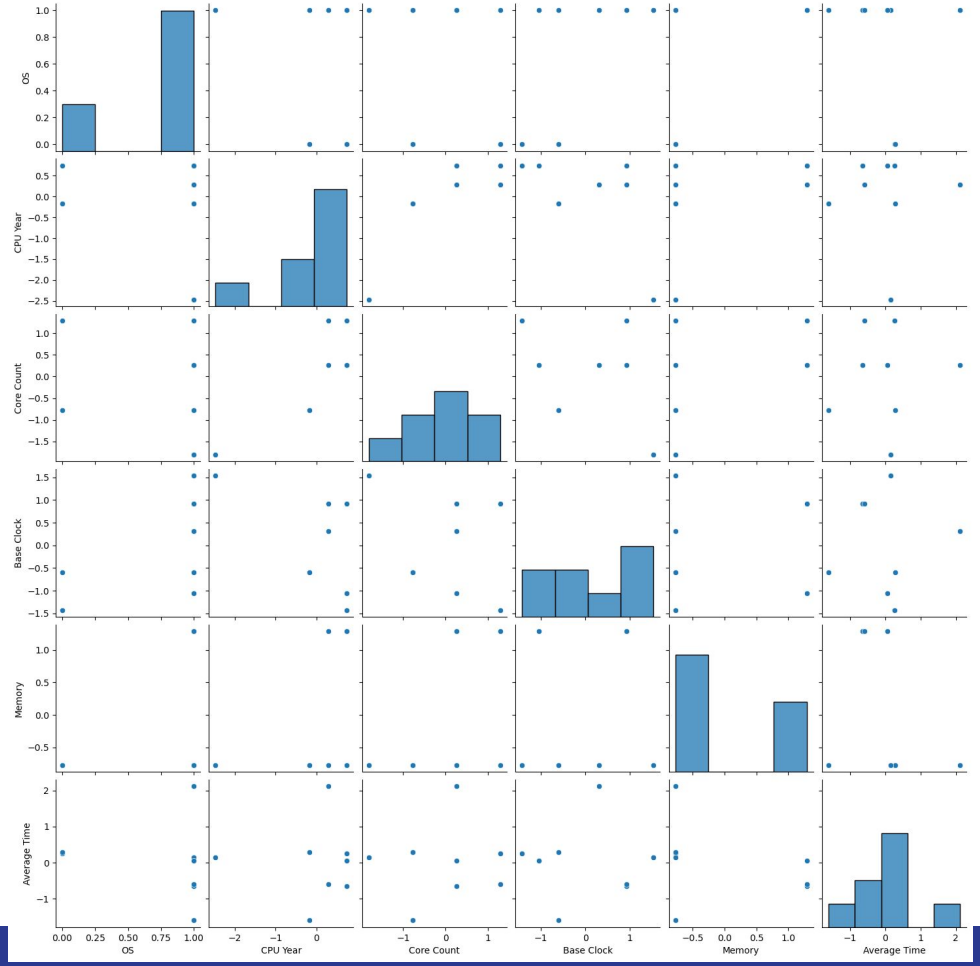
- Language: C++ would be the most time efficient language since it is a compiled language especially when compared to Python which is an interpreted language.
- Best Architecture: Windows 11, Intel i9 10850K with 10 cores @ 3.6 GHz, 32 GB.
- Worst Architecture: macOS Sonoma 14.4.1, Intel i7 9750H with 6 cores @ 2.6 GHz, 16 GB
- OS Prediction: We predicted that the Windows would be slightly more efficient than macOS.

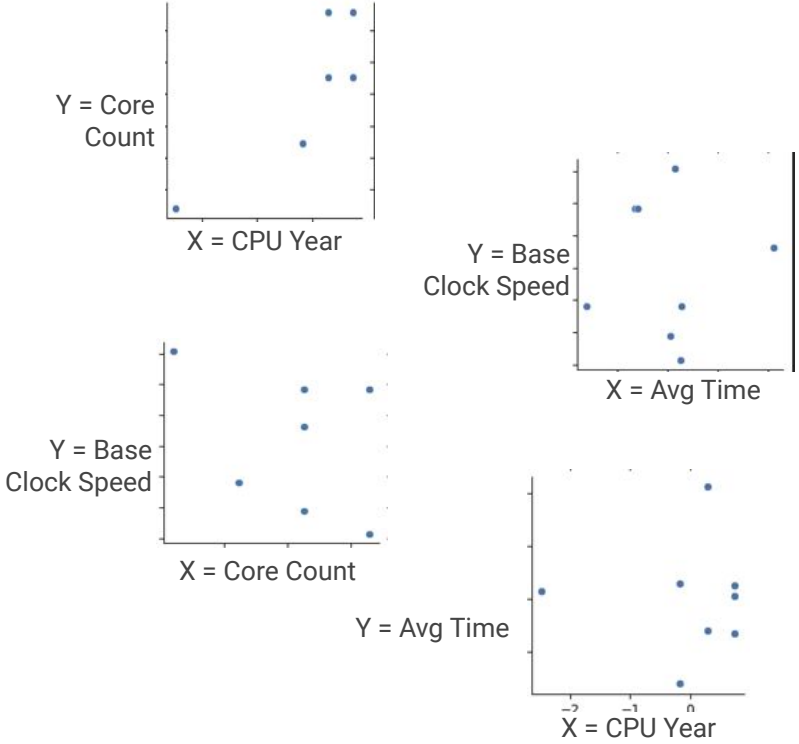
(Non-normalized) Histogram & Statistics



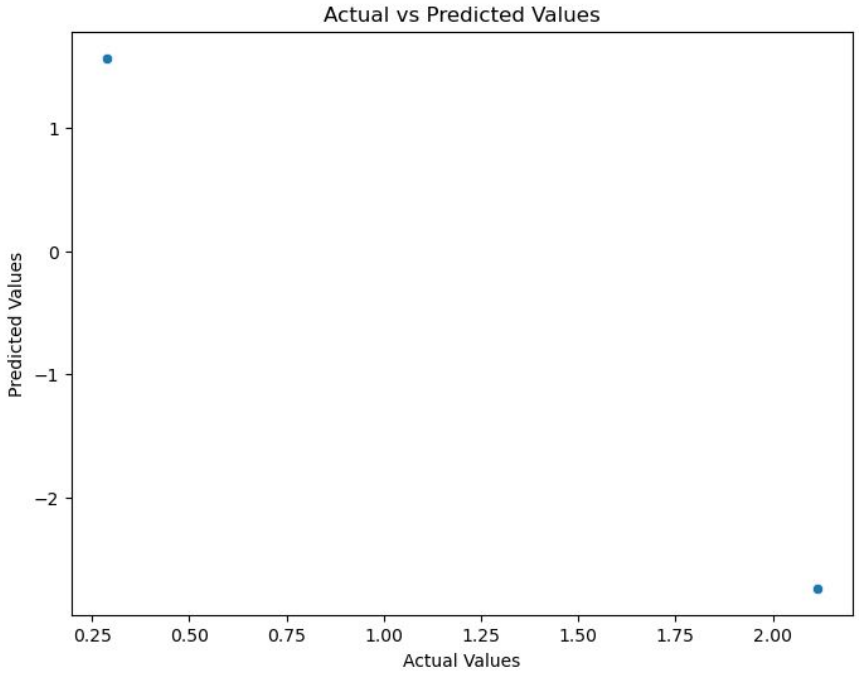
	Id	OS	Core Count	Base Clock	Memory	Python Time	C++ Time	Average Time
count	8.00000	8.00000	8.000000	8.000000	8.000000	8.000000	8.000000	8.000000
mean	4.50000	0.75000	7.500000	2.995625	22.000000	87.195750	21.273000	54.234375
std	2.44949	0.46291	2.070197	0.701861	8.280787	39.762226	10.866308	23.831878
min	1.00000	0.00000	4.000000	2.060000	16.000000	31.246000	5.623000	18.434500
25%	2.75000	0.75000	6.000000	2.526000	16.000000	70.416000	10.750000	40.583000
50%	4.50000	1.00000	8.000000	2.900500	16.000000	85.625000	24.780500	56.471750
75%	6.25000	1.00000	8.500000	3.600000	32.000000	90.462250	29.250000	60.077625
max	8.00000	1.00000	10.000000	4.000000	32.000000	172.707000	35.000000	101.353500

(Normalized) Pairplot





Linear Regression Model



Id	OS	CPU Year	Core Count	Base Clock	Memory	Python Time	C++ Time	Average Time
1	0	2021	10	2.06	16	84.77	35	59.885
2	0	2019	6	2.6	16	92.311	29	60.6555
3	1	2021	8	3.6	32	69.477	10	39.7385
4	1	2020	10	3.6	32	70.729	11	40.8645
5	1	2014	4	4	16	89.846	25	57.423
6	1	2020	8	3.201	16	172.707	30	101.3535
7	1	2019	6	2.6	16	31.246	5.623	18.4345
8	1	2021	8	2.304	32	86.48	24.561	55.5205

← we need more data!
“Garbage in, garbage out.”

Results

- Compare Total Run times between Python and C++ of Insertion Sort Algorithm running on Visual Studio
- Speed Difference between Python and C++
- Python takes longer run time on average, why is that?
- Visual Studio IDE: Used for both languages, which suggests that any differences in performance are not due to the development environment but inherent to the languages and how they handle the algorithm.
- Our prediction that the Windows machine would run our programs faster than the Mac using the same CPU, clocked at the same speed.

Questions?

What questions do y’all have, if any?