Exploration and Presentation - Assignment 2

Martin Frederiksen

3. maj 2021

1 C# vs Python

I Dette afsnit vil vi gerne redegøre for den genrelle forskel på c# og python.

2 Hypotese for hastighed af de to sprog.

Python og C# er bygget på to forskellige typer programmeringssprog, som har stor indflydelse på hastigheden af programmet man laver. C# er et compiled sprog hvilket vil sige at koden er oversat til maskinkode ved brugen af en compiler. På denne måde vil man få et meget effektivt program, som kan køres flere gange. Python er i modsætning til C# et interpreted programmeringssprog, som skal analysere, fortolke og eksekvere for hver gang programmet bliver kørt.

> "One of the jobs of a designer is to weigh the strengths and weaknesses of each language and then decide which part of an application is best served by a particular language" — IBM Corporation[1]

Som IBM nævner i [1] er det en vigtig kvalitet at kunne se styrker og svagheder for hver type sprog, og ud fra det vælge hvilket sprog der er bedst egnet til den type applikation der skal laves. Med disse ting i mente forudser vi at vores InsertionSort algoritme, vil kunne køre hurtigere i C# end Python. Grundet at Python vil skulle bruge længere tid på at eksekvere koden end C#, da den skal fortolke koden linje for linje.

3 Insertion sort

Vi har valgt at teste hastigheden af C# og Python, ved hjælp af Insertion Sort. Valget blev truffet ud fra kodemængden og hvad vi i forvejen havde liggende. Insertion sort kan skrives i én metode som gør at koden ikke ændre sig betydeligt meget mellem de to sprog. Som det ses nedenfor, er Insertion Sort koden i C# og Python næsten identiske og ingen af dem importere nogle eksterne pakker fra en tredjepart som kan gøre at hastigheden svækkes.

```
1  public static double sort(int[] arr, int t) {
2      for (int i = 1; i < arr.Length; ++i) {
3          int tempVal = arr[i];
4          int j = i - 1;
5
6          while (j >= 0 && arr[j] > tempVal) {
7              arr[j + 1] = arr[j];
8              j = j - 1;
9          }
10         arr[j + 1] = tempVal;
11     }
12     return t;
13 }
```

Listing 1: C# Insertion Sort

```python
def sort(arr, t):
    for i in range(1, len(arr)):
        tempVal = arr[i]
        j = i - 1

        while j >= 0 and arr[j] > tempVal:
            arr[j + 1] = arr[j]
            j = j - 1

        arr[j + 1] = tempVal
    return t
```

Listing 2: Python Insertion Sort

4 Test af hastighed.

Vores test(profiler) - Hvad gør vi og hvordan tester vi. Enviroment - windows/mac/linux - hvad køre ellers imens osv. For at teste hastigheden skal der bruges en profiler der kan måle gennemsnitstiden af metoden. Denne profiler er ikke helt ens i C# og Python, men koden vil teste hastigheden af Insertion Sort metoden kun. Nedenfor ses de Profiler i C# og Python.

```csharp
public static double CheckMethodTime() {
    int n = 1000;
    int count = 10000;
    double dummy = 0.0;
    double st = 0.0, sst= 0.0;

    Stopwatch stopWatch = new Stopwatch();
    for(int j = 0; j < n; j++) {
        stopWatch.Restart();
        for(int i = 0; i < count; i++) {
            int[] arr = new int[] {10, 22, 14, 4, 2, 50, 75, 4};
            dummy += InsertionSort.sort(arr, i);
        }
        stopWatch.Stop();
        double ns = (stopWatch.Elapsed.TotalMilliseconds * 1000000) / count;
        st += ns;
        sst += ns * ns;
    }
    double mean = st/n, sdev = Math.Sqrt((sst - mean*mean*n)/(n-1));
    Console.WriteLine($"{mean.ToString("F1")} ns +/- {sdev.ToString("F1")} ns");
    return dummy;
}
```

Listing 3: C# Profiler

```python
def CheckMethodTime():
    n = 1000
    count = 10000
    dummy = 0.0
    st = 0.0
    sst = 0.0

    for j in range(n):
        start = time.perf_counter_ns()
        for i in range(count):
            arr = [10, 22, 14, 4, 2, 50, 75, 49, 4]
            dummy += InsertionSort.sort(arr, i)
        stop = time.perf_counter_ns()
        ns = (stop - start) / count
        st += ns
        sst += ns * ns

    mean = st/n
    sdev = math.sqrt((sst - mean*mean*n)/((n-1)))
    print("{:.1f} ns +/- {:.1f} ns".format(mean, sdev))
    return dummy
```

Listing 4: Python Profiler

5 Resultatet

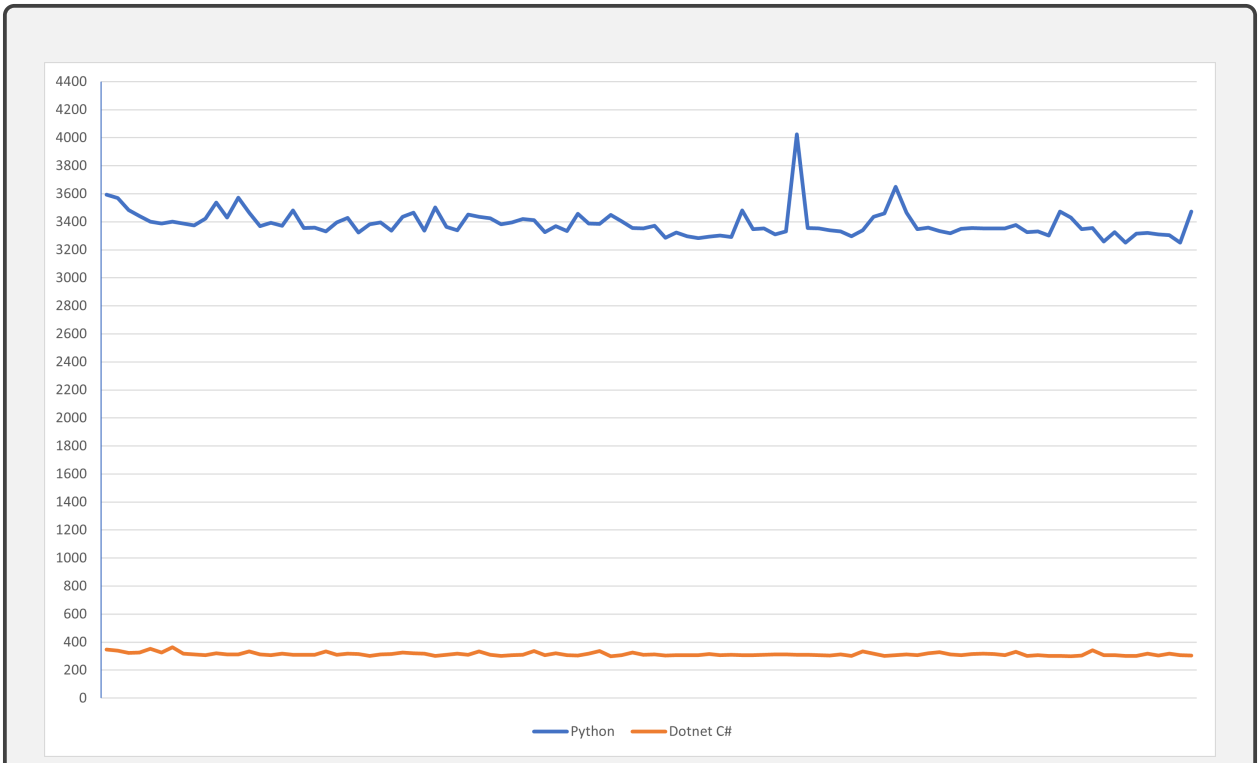
Vores resultater - På hastighed osv.

C#:

$$311,2 \text{ ns} \pm 13,1$$

Python:

$$3312,1 \text{ ns} \pm 88,9$$



Figur 1: Plot af hastighed på Insertion Sort

6 Todo

1. Forskellen på c# og python. Den genrelle forskel.
2. Hastighed - fra internettet (prediction) (husk source)
3. Vores program - SelectionSort hvorfor vil i køre det hvorfor er det en god ting? Den er kort at skrive så der er mindre kode mæssigt der kan gå galt. Basal sorting.
4. Vores test(profiler) - Hvad gør vi og hvordan tester vi. Enviroment - windows/mac/linux - hvad køre ellers imens osv.
5. Vores resultater - På hastighed osv.
6. Konklussion - Er vores antagelse korrekt? test

Litteratur

[1] IBM Corporation. Compiled versus interpreted languages. https://www.ibm.com/docs/en/zos-basic-skills?topic=zos-compiled-versus-interpreted-languages, 2010. [Online; accessed 03-Maj-2021].