

Exploration and Presentation - Exam Assignment

Martin, Frederiksen
cph-mf237@cphbusiness.dk

Andreas, Vikke
cph-av105@cphbusiness.dk

6. maj 2021

1 C# vs Python

Vores gruppe har valgt at undersøge hastighedsforskellen imellem Python og C#. Denne undersøgelse er blevet foretaget ud fra et stykke kode vi skrev i starten af faget Diskret matematik og algoritmer. Kodestumpen blev oprindeligt skrevet i C# men i anledning af denne opgave.

Python fordele:

- Sproget er portable, hvilket betyder at det kan køres på flere platforme som fx. Linux, Macintosh, windows osv.
- Sproget tillader programmører at udvikle en grafisk brugergrænseflade.
- Sproget blev etableret for 28 år siden og derved huser sproget et stort fælleskab.
- Sproget er effektivt eftersom der kun skal skrives få linjer kode.

C# fordele:

- Sproget er skalerbart og nemt at opdatere.
- Sproget har hurtig kompilering.
- Sproget er type stærkt da det ikke tillader risikable casts.
- Sproget er et struktureret programmeringssprog.
- Sproget besider krydsplatform evner.
- Sproget har et standard bibliotek.

Python ulemper:

- Sproget er langsomt.
- Sproget kan ikke bruges til mobil computing.
- Sprogets database er primitiv og underudviklet.
- Sproget har fejl og mangler som let kommer til udtryk.
- Sproget behøver meget testing.

C# ulemper:

- Sproget er ikke ment til underordnet programmering.
- Sproget har ikke nogen ubegrænset kompiler.
- Sproget kræver en kompilering hver gang der er sket en ændring i koden.
- Sproget kræver grundig testing.
- Sprogets eksekverbare filer kan kun eksekveres på windows.

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

```
1 def sort(arr, t):
2     for i in range(1, len(arr)):
3         tempVal = arr[i]
4         j = i - 1
5
6         while j >= 0 and arr[j] > tempVal:
7             arr[j + 1] = arr[j]
8             j = j - 1
9
10        arr[j + 1] = tempVal
11    return t
```

Listing 2: Python Insertion Sort

4 Test af hastighed.

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. På Listing 3 og 4 ses koden til profileren. Det kan ses her at vi kører 1000 iterationer af 10000 metode kald, som vi finder gennemsnittet af og udregner middelværdi og difference.

Begge sprogs programmer blev kørt på en Ubuntu 20.1 i et WSL2 miljø på Windows. Windows system havde udelukkende de nødvendige ting samt WSL2 kørende, for at minimere resource forbruget fra andre programmer. Derved kunne WSL2 bruge så mange resourcer som muligt under kørslen, som ville give et mere præcis resultat af den målte hastighed.

```

1 public static double CheckMethodTime() {
2     int n = 1000;
3     int count = 10000;
4     double dummy = 0.0;
5     double st = 0.0, sst = 0.0;
6
7     Stopwatch stopWatch = new Stopwatch();
8     for(int j = 0; j < n; j++) {
9         stopWatch.Restart();
10        for(int i = 0; i < count; i++) {
11            int[] arr = new int[] {10, 22, 14, 4, 2, 50, 75, 4};
12            dummy += InsertionSort.sort(arr, i);
13        }
14        stopWatch.Stop();
15        double ns = (stopWatch.Elapsed.TotalMilliseconds * 1000000) / count;
16        st += ns;
17        sst += ns * ns;
18    }
19    double mean = st/n, sdev = Math.Sqrt((sst - mean*mean*n)/(n-1));
20    Console.WriteLine($"{mean.ToString("F1")} ns +/- {sdev.ToString("F1")} ns");
21    return dummy;
22 }

```

Listing 3: C# Profiler

```

1 def CheckMethodTime():
2     n = 1000
3     count = 10000
4     dummy = 0.0
5     st = 0.0
6     sst = 0.0
7
8     for j in range(n):
9         start = time.perf_counter_ns()
10        for i in range(count):
11            arr = [10, 22, 14, 4, 2, 50, 75, 49, 4]
12            dummy += InsertionSort.sort(arr, i)
13        stop = time.perf_counter_ns()
14        ns = (stop - start) / count
15        st += ns
16        sst += ns * ns
17
18    mean = st/n
19    sdev = math.sqrt((sst - mean*mean*n)/((n-1)))
20    print("{:.1f} ns +/- {:.1f} ns".format(mean, sdev))
21    return dummy

```

Listing 4: Python Profiler

5 Resultatet

Vores resultater - På hastighed osv. Efter kørslen af programmerne, kunne vi sammenligne resultaterne som kan ses herunder. Det ses at C# programmet køre betydeligt hurtigere end Python programmet.

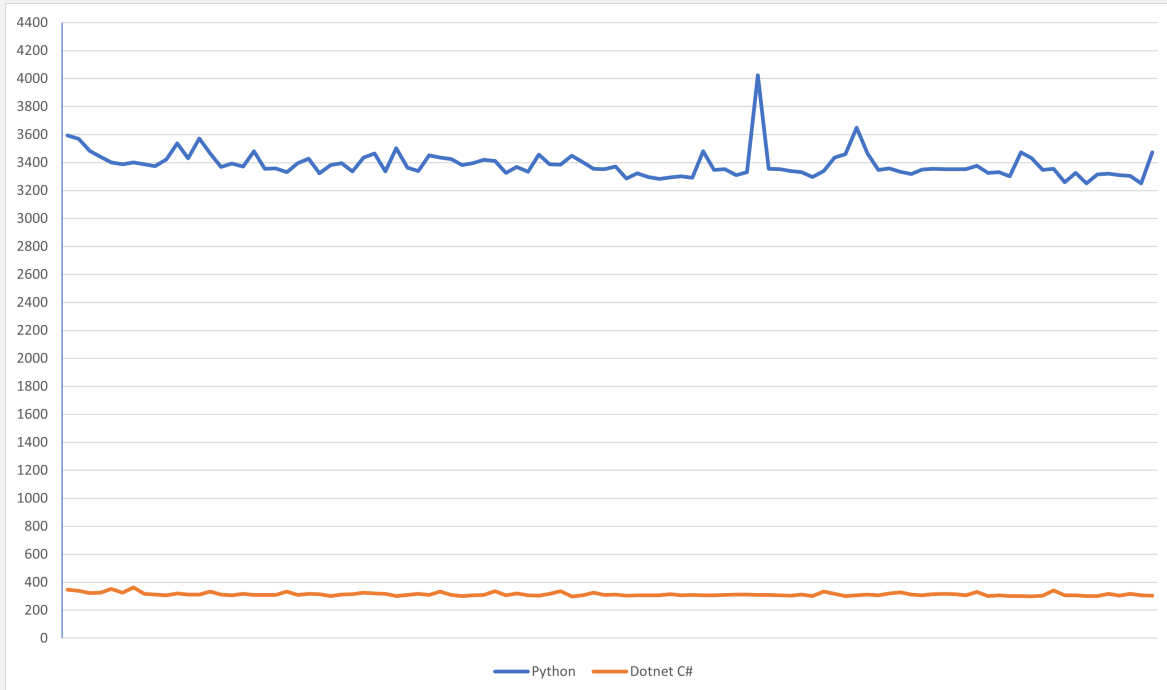
Resultatets middelværdi lå meget højere i Python, som vi havde forventet. Men udover dette er differencen også en del højere som vi ikke havde forventet. Ved at skrive alle gennemsnitsværdierne ud istedet for middelværdien kunne vi plotte de to programmersom kan ses på Figur 1. Dette gav et tydeligt indblik i stabiliteten af de to sprog, som viser at Python køre meget mere ustabil også.

C#:

311,2 ns \pm 13,1

Python:

3312,1 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].