Profiling

Nad jednotlivými matematickými funkciami v knižnici bol prevedený profiling, aby sme otestovali ako daná funkcia zvláda väčšie vstupy, ako dlho jej to trvá a či je pre danú operáciu dostatočne optimalizovaná.

Ako príklad na znázornenie bola vybratá operácia faktoriál. Rozhodovali sme sa medzi dvomi implementáciami faktoriálu v našej matematickej knižnici. Z pohľadu nám nebolo jasné ktorú verziu implementácie použiť a preto sme spravili profiling a podľa výsledku rozhodli, ktorú implementáciu použiť. Profiling bol prevedený nad dvomi implementáciami a to rekurzívnou a iteračnou faktoriálu.

Ako nástroj pre profiling bol použitý cProfile pre jazyk Python. Po preložení bola vygenerovaná tabuľka, ktorá zobrazuje celkový čas trvania operácia, počet volaní funkcie a viac informácií.

Prvý test bolo porovnanie ako sa operácie správajú pri malom vstupe, faktoriále čisla 20.

```
         43 function calls (24 primitive calls) in 0.000 seconds

   Ordered by: call count

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
     20/1    0.000    0.000    0.000    0.000 mathlib.py:111(fact2)
       19    0.000    0.000    0.000    0.000 {built-in method len}
        1    0.000    0.000    0.000    0.000 mathlib.py:18(__init__)
        1    0.000    0.000    0.000    0.000 {built-in method setrecursionlimit}
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects}
        1    0.000    0.000    0.000    0.000 mathlib.py:105(fact)
```

Z vygenerovanej tabuľky je nám jasné, že obidve operácie boli rýchle a teda nie je tu viditelný rozdiel. Rozdiel je avšak v počte volaní funkcie, zatiaľ čo iteračná verzia faktoriálu (fact) bola zavolaná jedenkrát, pri rekurzívnej verzií (fact2) nám počet vzrástol na číslo 20.

Druhý test pri veľkom vstupe, faktoriále čísla 4000:

```
         8003 function calls (4004 primitive calls) in 0.067 seconds

   Ordered by: call count

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
   4000/1    0.049    0.000    0.049    0.049 mathlib.py:111(fact2)
     3999    0.001    0.000    0.001    0.000 {built-in method len}
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects}
        1    0.000    0.000    0.000    0.000 {built-in method setrecursionlimit}
        1    0.018    0.018    0.018    0.018 mathlib.py:105(fact)
        1    0.000    0.000    0.000    0.000 mathlib.py:18(__init__)
```

Pri takto veľkom vstupe už je značný rozdiel vidieť pri celkovom čase, kedy iteračna verzia trvala 0.018 zatiaľ čo rekurzívna, 0.049, je to teda skoro až trojnásobný rozdiel oproti iteračnej verzií. Pričom rekurzívna funkcia mala 4000 počet volaní, čo je veľmi veľa a pri ešte väčších číslach by tato operácia trvala ešte dlhšie. V takomto prípade je tento rozdiel ešte zanedbateľný, ale pri väčších vstupoch by rekurzívna verzia faktoriálu trvala veľmi dlho a mohla spôsobovať problémy. Preto sme v našej knižnici použili iteračnú verziu faktoriálu po prevedenom profilingu.