HW7Solution

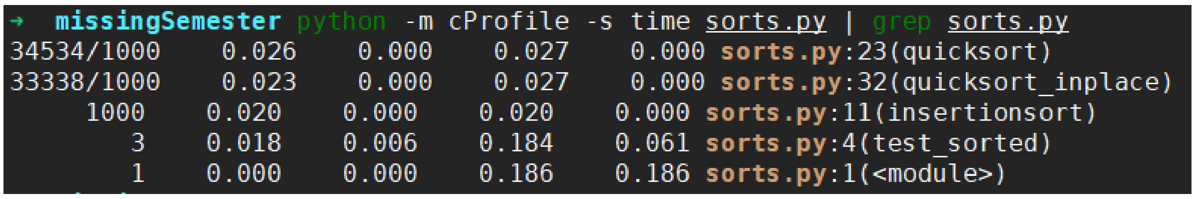
Author:金之航

StuID:2183411101

Problem

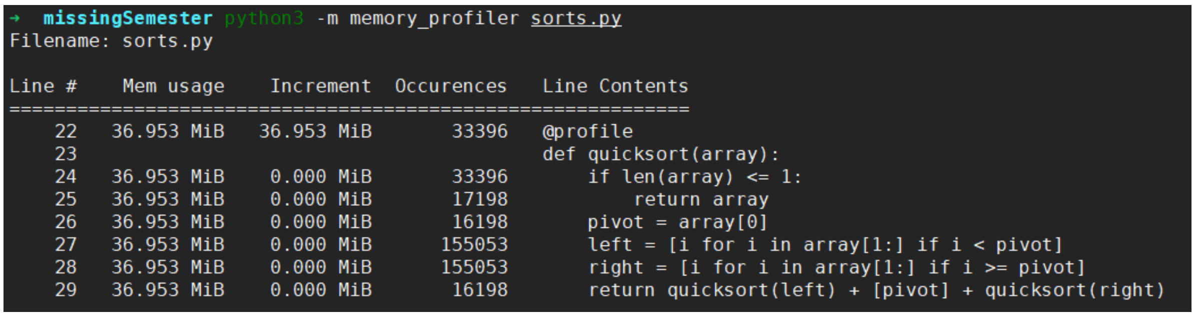
1.Sorts.py

```
python -m cProfile -s time sorts.py #按照执行时间排序
python -m cProfile -s time sorts.py | grep sorts.py
```

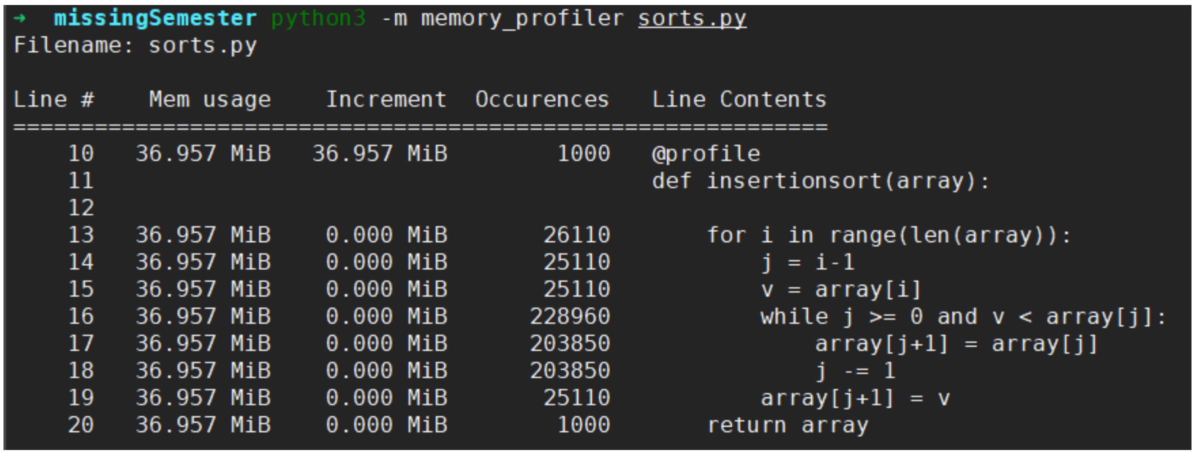


```
➜  missingSemester python -m cProfile -s time sorts.py | grep sorts.py
34534/1000    0.026    0.000    0.027    0.000 sorts.py:23(quicksort)
33338/1000    0.023    0.000    0.027    0.000 sorts.py:32(quicksort_inplace)
      1000    0.020    0.000    0.020    0.000 sorts.py:11(insertionsort)
         3    0.018    0.006    0.184    0.061 sorts.py:4(test_sorted)
         1    0.000    0.000    0.186    0.186 sorts.py:1(<module>)
```

```
pip3 install line_profile #安装line_profiler
kernprof -l -v sorts.py #为需要分析的函数添加装饰器 @profile，并执行

pip3 install memory_profiler #安装memory_profiler
python3 -m memory_profiler sorts.py #为需要分析的函数添加装饰器 @profile，并执行
```



```
➜  missingSemester python3 -m memory_profiler sorts.py
Filename: sorts.py

Line #    Mem usage    Increment  Occurences   Line Contents
============================================================
    22   36.953 MiB   36.953 MiB       33396   @profile
    23                                         def quicksort(array):
    24   36.953 MiB    0.000 MiB       33396       if len(array) <= 1:
    25   36.953 MiB    0.000 MiB       17198           return array
    26   36.953 MiB    0.000 MiB       16198       pivot = array[0]
    27   36.953 MiB    0.000 MiB      155053       left = [i for i in array[1:] if i < pivot]
    28   36.953 MiB    0.000 MiB      155053       right = [i for i in array[1:] if i >= pivot]
    29   36.953 MiB    0.000 MiB       16198       return quicksort(left) + [pivot] + quicksort(right)
```

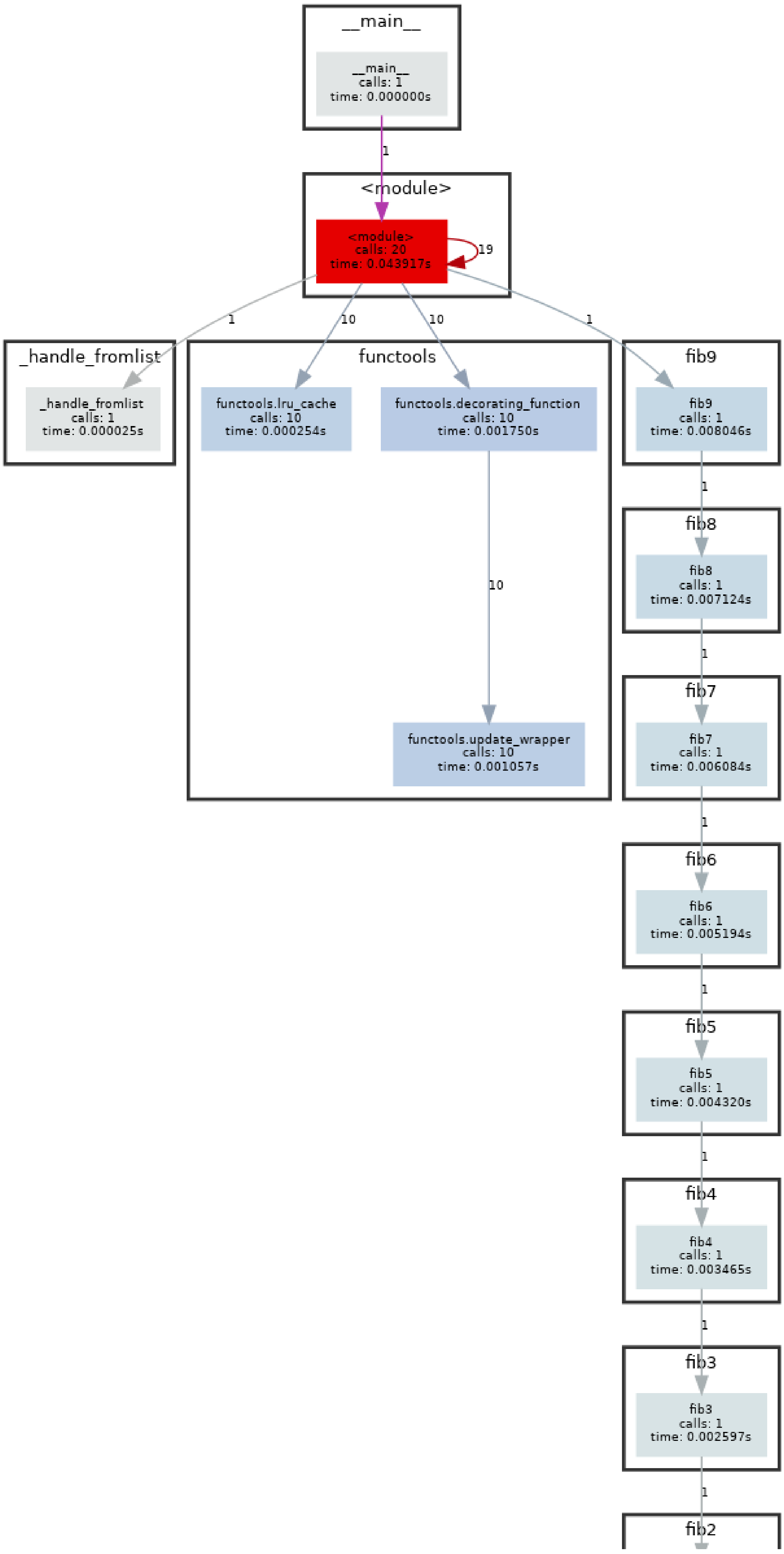


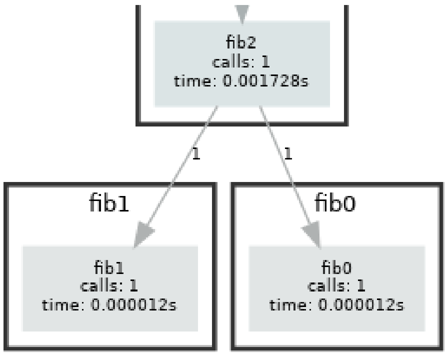
```
➜  missingSemester python3 -m memory_profiler sorts.py
Filename: sorts.py

Line #    Mem usage    Increment  Occurences   Line Contents
============================================================
    10   36.957 MiB   36.957 MiB        1000   @profile
    11                                         def insertionsort(array):
    12
    13   36.957 MiB    0.000 MiB       26110       for i in range(len(array)):
    14   36.957 MiB    0.000 MiB       25110           j = i-1
    15   36.957 MiB    0.000 MiB       25110           v = array[i]
    16   36.957 MiB    0.000 MiB      228960           while j >= 0 and v < array[j]:
    17   36.957 MiB    0.000 MiB      203850               array[j+1] = array[j]
    18   36.957 MiB    0.000 MiB      203850               j -= 1
    19   36.957 MiB    0.000 MiB       25110           array[j+1] = v
    20   36.957 MiB    0.000 MiB        1000       return array
```

插入排序的内存效率略好于快速排序，因为快速排序需要一些额外的空间来保存结果，而插入排序则是原地操作，但是耗时更高。

2.fibonacci

```
pip3 install pycallgraph
pip3 install graphviz
pycallgraph graphviz -- ./fibonacci.py
```

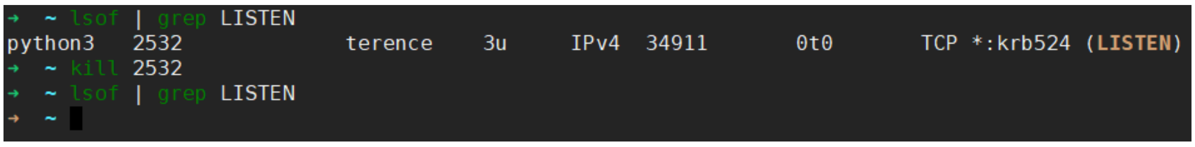

__main__
__main__
calls: 1
time: 0.000000s
1
<module>
<module>
calls: 20
time: 0.043917s
19
1
10
10
1
_handle_fromlist
_handle_fromlist
calls: 1
time: 0.000025s
functools
functools.lru_cache
calls: 10
time: 0.000254s
functools.decorating_function
calls: 10
time: 0.001750s
10
functools.update_wrapper
calls: 10
time: 0.001057s
fib9
fib9
calls: 1
time: 0.008046s
1
fib8
fib8
calls: 1
time: 0.007124s
1
fib7
fib7
calls: 1
time: 0.006084s
1
fib6
fib6
calls: 1
time: 0.005194s
1
fib5
fib5
calls: 1
time: 0.004320s
1
fib4
fib4
calls: 1
time: 0.003465s
1
fib3
fib3
calls: 1
time: 0.002597s
1
fib2



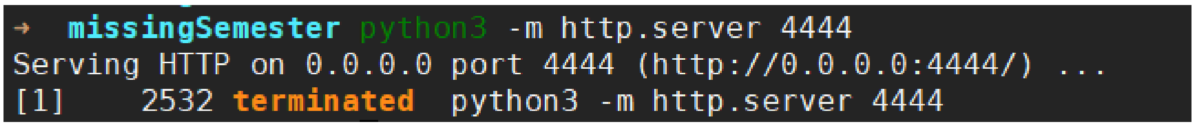
Generated by Python Call Graph v1.0.1
http://pycallgraph.slowchop.com

3.kill http.server

```
python -m http.server 4444
lsof | grep LISTEN
```



```
➜  ~ lsof | grep LISTEN
python3   2532              terence    3u      IPv4  34911        0t0        TCP *:krb524 (LISTEN)
➜  ~ kill 2532
➜  ~ lsof | grep LISTEN
➜  ~ 
```



```
➜  missingSemester python3 -m http.server 4444
Serving HTTP on 0.0.0.0 port 4444 (http://0.0.0.0:4444/) ...
[1]    2532 terminated  python3 -m http.server 4444
```

4.stress -c 3

```
stress -c 3
taskset --cpu-list 0,2 stress -c 3
man taskset #taskset 命令可以将任务绑定到指定CPU核心
```