Universität Konstanz

Introduction to Computation for the Social Sciences

Assignment 8

Prof. Dr. Karsten Donnay, Marius Giebenhain, Stefan Scholz
Winter Term 2019 / 2020

Please solve the exercises below and commit your solutions to our GitHub Classroom until Jan, 14th midnight. Submit all your code in one executable file (*py* / *ipynb*) and your text in one text file (*txt* / *md* / *pdf*). You can score up to 10 points in this assignment. You will get individual feedback in your repository.

Exercise 1: Sieve of Eratosthenes (4 Points)

The Sieve of Eratosthenes is an efficient algorithm to find small primes by eliminating non-prime numbers iteratively. It uses the fact that any multiple of a given number (larger than 1) is necessarily not a prime number.

a) Write a Python function `sieve(n)` that implements the Sieve of Eratosthenes algorithm for a given maximum natural number `n`. The function should return a dictionary of the form:

```
{'prime':[prime numbers], 'non-prime':[non-prime numbers]}
```

i.e. lists of prime and non-prime numbers between 1 and `n`.

b) Use the module *time* in Python to compare the runtime of the sieve algorithm for `n` = 1,000 and `n` = 10,000. Print the respective runtimes.

c) Write a Unit Test that includes a `setUp()` function. Within the `setUp()` function, read in the correct values for `sieve(200)` from the text file *primes_check.txt*. The file contains the prime numbers up to 200 in the second line and the non-prime numbers up to 200 in the fourth line. Use two lists of the correct prime and non-prime values to write a test method that checks the correctness of your `sieve(n)` function.

d) Explain why the sieve algorithm is not very efficient for large `n`?

Exercise 2: Mergesort and Binary Search (4 Points)

In the lecture, we discussed binary search and the mergesort sorting algorithm. Please look up further details on the implementation of *binary search*[1] as needed.

a) Write a Python function `mergesort(lst)` that implements the mergesort algorithm to sort a list `lst`.

b) Write a Python function `search(n,x)` that uses the mergesort algorithm in combination with a binary search approach to enable searching an integer `n`

in an unsorted list of integers `x`. The function returns `TRUE` if the list `x` contains `n` and `FALSE` otherwise.

c) Write a Unit Test that checks the correct execution of your `search()` function using the following test data:

   i. `32,[45,19187,232,8974,32,547,9081,2,67,421]`
   expected result: `TRUE`

   ii. `191,[345,10,754743,435,321,65,2690,1234,5]`
   expected result: `FALSE`

d) Explain the overall efficiency of your search method in *O()* notation.

Exercise 3: Theoretical Programming Concepts (2 Points)

In this exercise, we review some basic theoretical programming concepts. Please explain briefly what we understand by

a) structured data abstraction

b) control abstraction

Why is abstraction an important concept in programming?

[1] http://interactivepython.org/runestone/static/pythonds/SortSearch/TheBinarySearch.html