UNIVERSITÄT DES SAARLANDES

ZBI ZENTRUM FÜR BIOINFORMATIK

Assignment 03

Programming with Python (for Bioinformatics)

Johanna Schmitz

22.05.2024; Submit before **04.06.2024, 23:59**

Assignment 03 - Task 1

Assignment 03 - Task 1 (4 points)

You work in a parcel shop that has a robot to rearrange parcels before delivery. The robot is capable of moving parcels from one stack of parcels to another, such that the desired parcels will be at the top of all stacks.

To know which parcels to move and in what order, the robot receives the following input

```
    [E]
[B] [D]
[A] [C] [F]
 1   2   3
move 1 from 1 to 3
move 2 from 2 to 1
```

In this example, there are three stacks of parcels. Here, stack 1 contains two parcels: parcel A is on the bottom, and parcel B is on top.

Assignment 03 - Task 1 (continued)

Then, the rearrangement steps are given. In each step, a number of parcels is moved from one stack to a different stack. In the first step of the above rearrangement procedure, one parcel is moved from stack 1 to stack 3, resulting in this new configuration:

```
    [E]
    [D] [B]
[A] [C] [F]
 1   2   3
```

In the second step, two parcels are moved from stack 2 to stack 1. Parcels are **moved one at a time**, so the first parcel to be moved (E) ends up below the second parcel:

```
[D]
[E]     [B]
[A] [C] [F]
 1   2   3
```

Assignment 03 - Task 1 (continued)

1. Write a function `read_config` that given a filename with a configuration returns the starting stack of parcels and the moves.
   The stacks should be stored as a dictionary with the stack id as key and the parcel list as values. The moves should be provided as a list of tuples `(numParcels, currentStack, newStack)` (see also example tests).
2. Write a function `rearrange_parcels` with the stacks and moves as function arguments (from part 1) and returns the string of parcels that are at the top after all moves, e.g. `DCB` for the provided example.

You can assume that we will only test your function on valid input files.

Assignment 03 - Task 2

Assignment 03 - Task 2 (3 points)

Given a collection of DNA sequences with equal size n, the position weight matrix (PWM) P is a $4 \times n$ **matrix** with rows **A,C,G,T** and entry P_{ij} denoting the number of times base i occurred at position j in one of the DNA strings.

The **consensus** sequence of size n is then formed by selecting at each position the most common base. If at one position several bases have the same maximum count, the consensus sequence is not unique.

Example:

```
DNA sequences: ACCTGAC    PWM: 2 0 0 0 0 3 0    Consensus: ACCTGAC
               TCCTGTC         0 4 3 0 1 0 4               TCCTGAC
               ACCTGAC         0 0 1 0 3 0 0
               TCGTCAC         2 0 0 4 0 1 0
```

Assignment 03 - Task 2 (continued)

1. Write a function `pwm` that gets as an argument the filename and returns the position weight matrix (as a list of lists). The input file contains one DNA sequence per line.
2. Write a function `consensus_sequence(pwm)` that returns a list of all consensus sequences (in lexicographic order) for the given position weight matrix.

You can assume that we will only test your function on valid input files.

Assignment 03 - Task 3

Assignment 03 - Task 3 (3 points)

We have a very basic computer that can only perform a sequence of operations to a starting integer value n. The available operations are:

- `add x` adds `x` to the current value
- `mul x` multiplies the current value with `x`
- `div x` computes the integer division of the current value by `x`
- `mod x` computes the current value modulo `x`

Write a function `solve(n, operations)`, where `n` is the starting number and `operations` is a string describing the operations to apply to `n`. The function should return the final number after performing all operations.
Implement the task using a **match** statement for the operations.

Example

The function call `solve(10, "add 6 mul 2 div 4 mod 5 mul 3")` should return 9, because

1. $10 + 6 = 16$
2. $16 \cdot 2 = 32$
3. $32/4 = 8$
4. $8 \mod 5 = 3$
5. $3 \cdot 3 = 9$