Dynamic Languages - Übungen Blatt 3

Abgabedatum: `10. Mai 2010`

Aufgabe 1 - Game of Life

(2 Punkte)

Implement a Game of Life simulation: http://de.wikipedia.org/wiki/Conways_Spiel_des_Lebens

The board should be infinite. Use a set of coordinates `(n, m)` representing the position of all live cells, and write a function that computes a new set representing the next generation.

For *sets*, you can use the built-in type `set` (see http://docs.python.org/library/stdtypes.html#set-types-set-frozenset).

To see the results, also write a function that turns such a set into a multiline string, with spaces or 'X' signs. See the test for details.

Aufgabe 2 - Pygame

(3 Punkte)

1. Write a graphical viewer for Game of Life using the Pygame library (pygame.org). As an example of how to use the library, we added the file `pygame_demo.py`.

2. Write a function that does the reverse of the `lifestring` function, taking a string as an argument and turning it into a set of life cells. There are tests for this in `blatt4.py`.

3. Combine the two above into a commandline program that takes a file containing the description of the initial configuration of the board and runs it with the pygame viewer. There is an example file `factory.life`.

Aufgabe 3 - Huffman Coding

(3 Punkte)

Huffman Coding is a way to encode sequences of characters into bits in such a way that the common characters need less bits than the less common ones:

`http://en.wikipedia.org/wiki/Huffman_coding`

Implement a Huffman encoder and decoder. For this you will have to implement classes that can represent the huffman trees. You will also need the following functions:

- `make_tree` which turns a dictionary mapping characters to frequencies into a huffman tree.
- `make_mapping` which turns a tree into a dictionary mapping input characters to strings of `0`s and `1`s.
- `encode` which encodes a string given such a mapping.
- `decode` which decodes an encoded string given a huffman tree.

See the tests in `blatt3.py`.

Aufgabe 4 - Fibonacci Folge

(2 Punkte)

In Ruby the Hash constructor can take a block as an optional parameter. This block is executed when a key is accessed that does not correspond to a hash entry (see http://ruby-doc.org/ruby-1.9/classes/Hash.html#M000369). Using this feature implenment a Hash that calculates the fibonacci sequence up to the value of the accessed key. Running time should not be exponential.