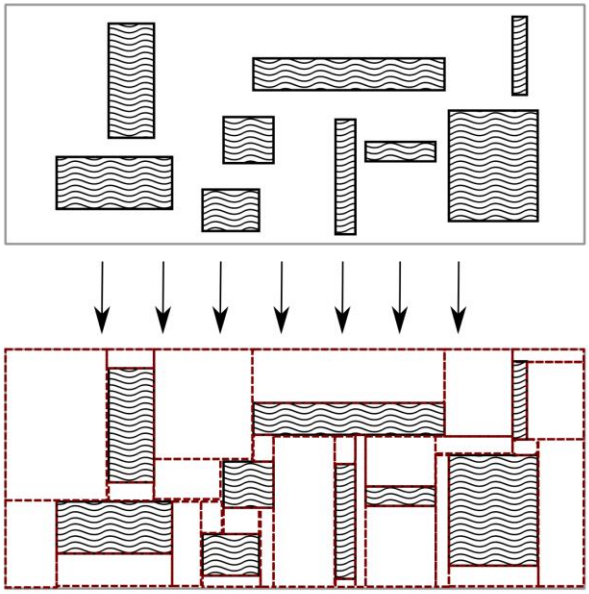
Home assignment:

- In general, the task is to implement an algorithm which takes as input a set of non-overlapping rectangles within a larger frame, and fills the gaps with complementary, non-overlapping rectangles, such that any point inside the frame is contained within one of the rectangles. An example is shown below, with the dashed red lines marking the sides of the complementary rectangles.



- **Specifics:**

  Given the coordinates of an axis-aligned rectangular frame **R**, and **N**>=0 smaller, non-overlapping rectangles **{r_n}** (also axis-aligned) bound by **R:**

  - Return a list of complimentary non-overlapping rectangles **{c_n }** such that the union of **{c_n}** and **{r_n}** will cover the entirety of **R**.
  - Write a unit test function to test your algorithm with randomly generated input:
    - Randomized **R** and **{r_n}** may have 0 area. They are to be considered lines and are legal input (in some cases the output list might have lines as well).
  - Notes:
    - All input and output rectangles are axis-aligned.
    - **{r_n}** can be empty, or already filling the whole of **R**.
    - *There is no single correct way* to fill the frame, and there are no optimization requirements (e.g., minimal number of output rectangles), or any other restrictions. So which method you opt for is up to you.

- Bonus: what changes need to be made to the function if **R** is not axis-aligned (all the smaller rectangles - r_n and c_n - are axis-aligned to **R**).

Assignment submission rules:

- The assignment should be written in Python - version 3.6 or later.
- You are free to use any tools or libraries you find online.
- This assignment comes to showcase your ability to deduce solutions; assess your self-sufficiency; and your ability to write clean, and tested code – please stay true to the spirit of the test.
- The functional code and the unit test *should* be in different files. (Optional)
- The submission should include simple instructions (a readme file) for how to run the code, e.g., "`python3.6 my_unit_test.py`". This file must include any uncommon libraries/packages we might need to install + how to install them.
- ** The results of running the instructions in the readme should be enough to convince us that the code works as intended. **We will not write extra code to analyze or test your work.** Something that simply prints a set of numbers, or a "*test passed*" message, will not be accepted.
- Time permitting, your code should be documented.
- The assignment file(s) should be archived (outlook doesn't like PY files) and sent as an email attachment to nikolai@imagry.co and e.sweid@imagry.co .

Good luck!