Prof. Dr. Rüdiger Ehlers Francisco Palau, M.Sc.

Applied Computational Engines 2017 – Assignment Sheet 3

Due: Part 1 of the project: Monday, **30th April 2018**, 7:30 am
The rest: Monday, **7th May 2018**, 7:30 am

Please indicate your **name** and **email address**. You can work in **groups** of up to **three** students. Only one submission per group is necessary. However, in the tutorials every group member must be able to present the solutions to each problem solved by your group.

Please submit your solutions either

- by e-mail to `fpalau@uni-bremen.de` **and** `rehlers@uni-bremen.de`, or
- on paper in **letter box 52** (Francisco Palau-Romero) on floor 6 of the MZH building.

Note that you will need 50% of the points on all exercise sheets in order to take the "Fachgespräch" OR the oral exam. We may ask you to present your solutions in the tutorial, especially if you work in a group. We aim for asking everyone taking part in the course to present at least twice during the course of the semester.

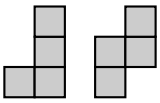
Project: Cutting Fabric (100 pts.)

It is time to apply SAT solving in practice! For this exercise, you are asked to program a tool that computes SAT instances that encode the search for efficient ways to cut fabric into pieces of predefined shapes.

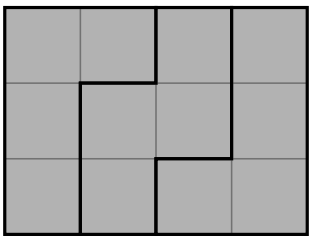
A factory gets very long rolls of fabric of a fixed width. They manufacture car seats, which are sewed together from parts of different shapes. These shapes are cut out from the fabric rolls, and of course the factory managers want to make sure that there are no left-overs after cutting out the pieces as these would have to be thrown away.

For this project, we are not concerned with how many parts of which shapes are cutted out, but rather that we are not wasting any part of the fabric roll. All shapes consist of blocks of 10x10cm each, which simplifies the problem. The fabric is of width w (in blocks), and we want to compute a *pattern* that repeats every h steps.

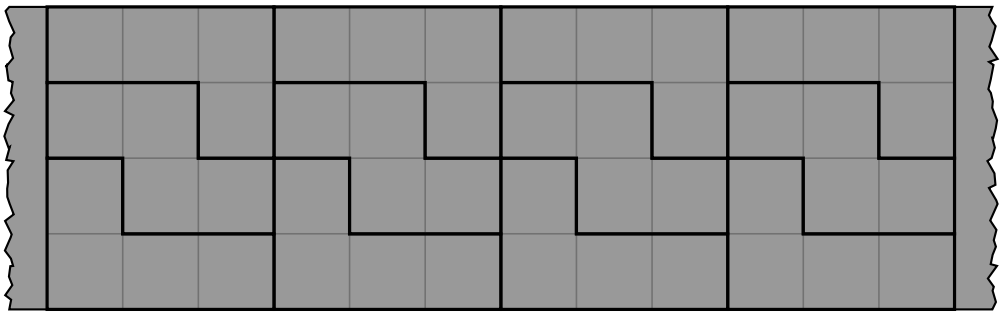
As an example, consider the following shapes:



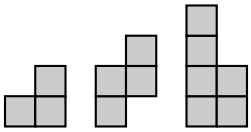
For $w = 4$ and $h = 3$, we could compute the following **cutting plan**:



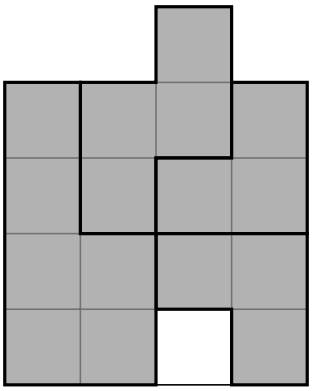
The shapes fill the 4×3 large *workspace* perfectly. Since we have a long roll of fabric, we can now cut the whole roll by repeating the cutting plan:



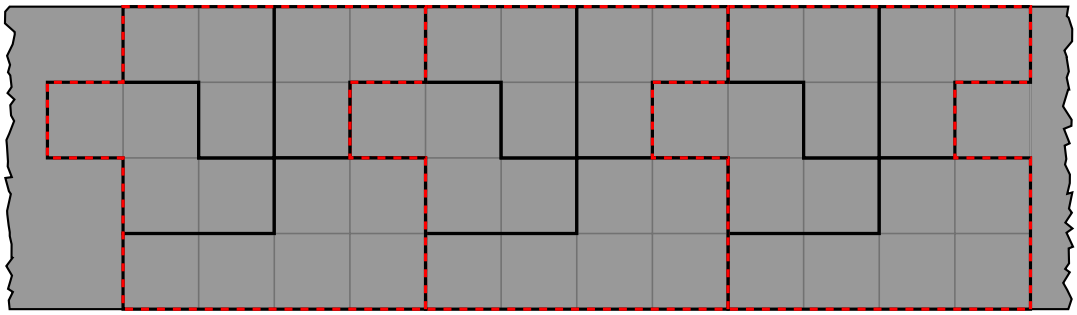
What we are really interested in is a cutting plan that can be used for the whole roll, so it is OK if some shape exceeds the boundaries of the workspace if it is planned for at the other end of the workspace. Let us look at the following example with the following shapes:



For $w = 4$ and $h = 4$, we could compute the following cutting plan:



This cutting plan is valid as well, as it can be used on the fabric roll in the following way:



So except that a bit of fabric is lost at the two ends of the roll, everything is fine. The loss is acceptable as the rolls are very long. Note that the shapes can only wrap around in the **height** dimension of the workspace. The width of the fabric is fixed, so the shapes cannot wrap around in that dimension.

The fabric has two differently coated sides, so the shapes cannot be mirrored. However, all shapes can be rotated.

Problem Description: Your task is to build a program (in your favourite commonly used programming language) that takes three inputs, namely

1. the width of the fabric roll,
2. the height of the workspace (for planning), and
3. the shapes of the parts to cut out,

and then builds a SAT instance (in DIMACS format) that is satisfiable if and only if a cutting plan exists that does not waste any space on the fabric roll (except possibly once at each end as in the example above).

The tool is supposed to take one input file. The input file is a text file and has a very simple structure:

- The first line contains the width of the fabric roll
- The second line contains the height of the workspace
- The third line contains the number of different shape that can be cut out
- The lines afterwards contain the description for each shape. Every shape has a rectangular base shape, and blocks that are part of the shape are denoted by a `+`, while the others are denoted by a `.` character. The shape ends with an empty line.

Overall, the input file for the first example above could look as follows:

```
4
3
2
.+
.+
++

.+
++
+.
```

Later this week, we will upload a couple of challenge problems to

`http://motesy.cs.uni-bremen.de/ace2018`

that you can use to benchmark your solution. The problem encoding that your tool performs should be good enough so that most of the resulting SAT instances can be solved by modern solvers on reasonably modern computers in reasonable time (yes, this is fuzzy on purpose – if in doubt whether your solution is good enough, please ask).

We are aware that this explanation may leave some questions open. So please send us e-mails with questions!

Please solve the project in two steps:

- First, explain how many SAT variables you intend to use in your program and what these SAT variables will represent. **This part is due on the 30^{th} of April and worth 15 points.**
- Then, design and implement your solution. **This part is due on the 7^{th} of May and worth 85 points.** You do not have to use the variable encoding/meanings from the first part of the question in case you either (1) found a better encoding in the meantime, or (2) you want to replace your encoding by one that we suggested to you by e-mail. As long as the final program works as intended and the reasoning capabilities of the SAT solver are made good use of to find the solution, any encoding will do. Please submit the source code for your program so that we can test it.