Problem Statement

Part A - The Puzzle

I. Description

Given an empty rectangular frame of length L and width W, and N rectangular tiles, design and write a GA that arranges the tiles in the Frame in such a way that minimizes the free space.

II. Inputs

For the desired population size, a puzzle of N tiles will be generated. The initial population will be stored as a JSON file **population.json**. Individuals in the population are also JSON strings with the following keys and values:

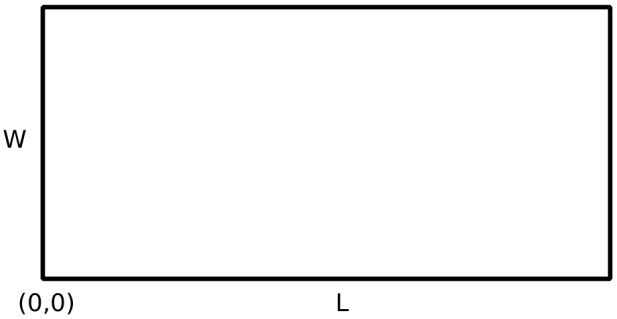
Length	Frame Length (L)
Width	Frame Width (W)
Pieces	Number of tiles (N)
Puzzle	A list of *tiles*, which is basically the arrangement of tiles in the frame (see below)

A *tile* is a list [x, y, l, w], where (x, y) are coordinates of the bottom left corner of the tile, and (l, w) are the dimensions of the tile.

Hence, a *Puzzle* is made of a list of tiles = $[[x_1, y_1, l_1, w_1].....,[x_N, y_N, l_N, w_N]]$, for a puzzle of N tiles.

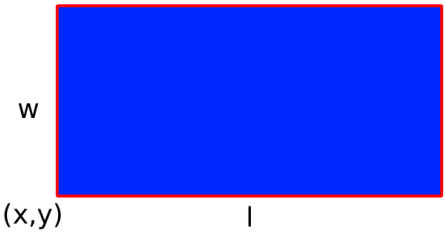
III. Assumptions:

- The bottom left coordinate of the frame is always (0, 0).

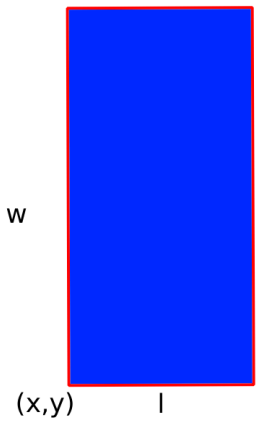


A Frame of length L and width W.

- The tiles are free to **rotate**, **swapping the length and width** but the **bottom left coordinate remains unchanged**.



A tile of length l and width w with bottom left coordinate (x,y).



A tile after rotation.

Note that you don't need to change the names of the dimensions, just swap their values, to execute a rotation.

IV. Fitness Evaluation

Fitness of an individual is a reflection of how well the tiles fit in the frame, which could be measured by finding the **percentage of free space** in the frame.

V. Supporting Code

Please open the attached **notebook** in **Google Colab** for **running the supporting codes.**

Follow the below steps:

- Go to COEN432-Assignment#2.zip
- Download and extract the main folder to the desired location.
- In a web browser go to https://colab.research.google.com/
- Select **cancel**
- From the **File menu** select **Upload notebook**
- Upload **COEN432_Assignment_2_PART_A.ipynb**
- On the extreme left of the notebook screen - **click on the folder icon**
- Click on Upload and **upload** the **Supporting_Code.zip** file

Please carefully **read through the instructions** in the notebook.

A sample code to read/write JSON files in JAVA and C++ is also shared (**Supporting_Code** folder).

For Java, use the jar file json-simple-1.1.jar .

For cpp, refer the link https://en.wikibooks.org/wiki/JsonCpp

Part B – Automata Machine

I. Description

Given a randomly generated set of rules, an 8-bit initial state, and an 8-bit goal state, write a GA that finds the set of rules that will transform the initial state to the goal state after some amount of passes.

II. Inputs

For the desired population size n, an initial state, a goal state, and n sets up rules tables will be generated. The initial population will be stored as a JSON file **automata-population.json**. The format of the JSON file is as follows:

Initial State – 8-bit binary number
Goal State – 8-bit binary number
Rules Table – 5-bit truth table with output values of 0, 1, 2, or 3

III. Assumptions:

An individual is a set of rules based off of a 5-bit truth table:

Input	Rule
00000	1
00001	0
00010	3
...	...
11110	2
11111	2

The rules work as follows

> 0 - replace middle value with a 0

> 1 - replace middle value with a 1

> 2 - delete the middle value

> 3 - replicate the middle value to the left or the right (you decide)

Here is an example with a 5-bit initial state and a 3-bit sliding window

Example: 5-bit Initial State and 3-bit Sliding Window

Input	Rule
000	0
001	1
010	2
011	3
100	3
101	2
110	1
111	0

Initial State: 01000

Goal State: 11111

Current State = copy(Initial State)

Current State: 01000

First Pass

Next State = copy(Current State)

Next State: 01000

Step	Current State	Sliding Window	Rule	Change to Next State this Pass
0	01000			01000 (note this is a copy of Current State)
1	***010***00	010	2	0_000
2	0***100***0	100	3	00***0***00
3	01***000***	000	0	000***0***0
4	***010***00	000	0	0000***0***
5	***010***00	001	1	***1***0000

During each pass ONLY change Next State - DO NOT CHANGE THE INITIAL/CURRENT STATE

Update Current State once all passes are completed.

Next State: 10000

Current State = copy(Next State)

Current State: 10000

End of First Pass

IV. Fitness Evaluation

Fitness is determined using the Minimum Edit Distance (MED) between the final state and the goal state. A lower MED means a better fitness value. For more information on the MED, see the following: [Minimum Edit Distance (MED)]

V. Supporting Code

Please open the attached **notebook** in **Google Colab** for **running the supporting codes.**

Follow the below steps:

- Go to the location where you extracted COEN432-Assignment#2.zip
- In a web browser go to https://colab.research.google.com/
- Select **cancel**
- From the **File menu** select **upload notebook**
- Upload **COEN432_Assignment_2_PART_B.ipynb**
- On the extreme left of the notebook screen - **click on the folder icon**
- Click on Upload and **upload** the **Supporting_Code.zip** file

Please carefully **read through the instructions** in the notebook.

A sample code to read/write JSON files in JAVA and C++ is also shared (**Supporting_Code** folder).

For Java, use the jar file json-simple-1.1.jar .

For cpp, refer the link https://en.wikibooks.org/wiki/JsonCpp