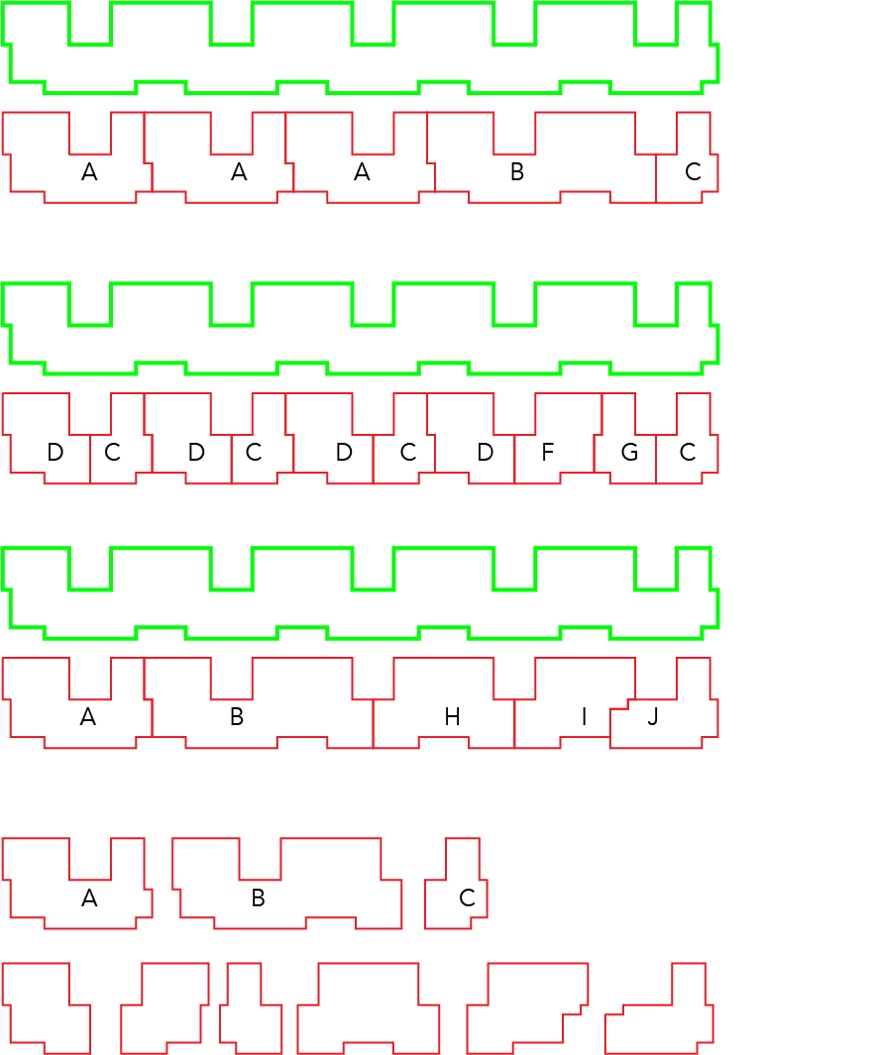
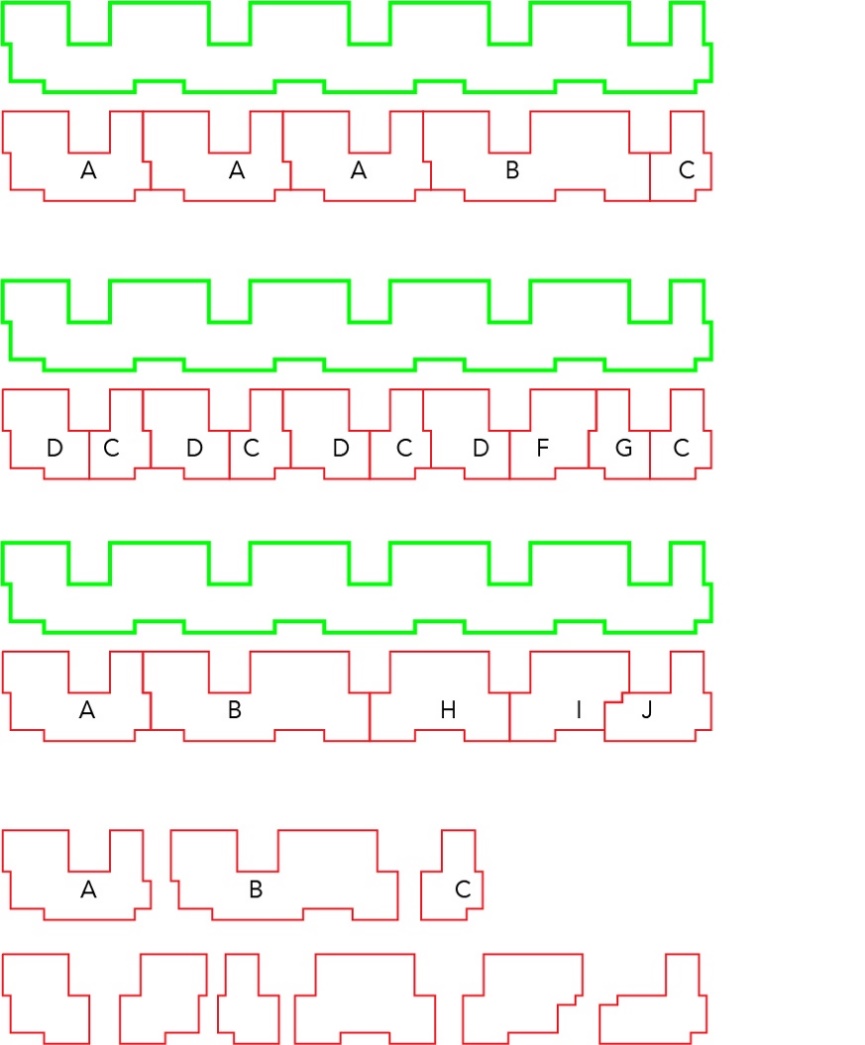
**Problem 1: Packing optimination**





1. To create a data structure that can be used to store a geometric 2d shape with rectangles.

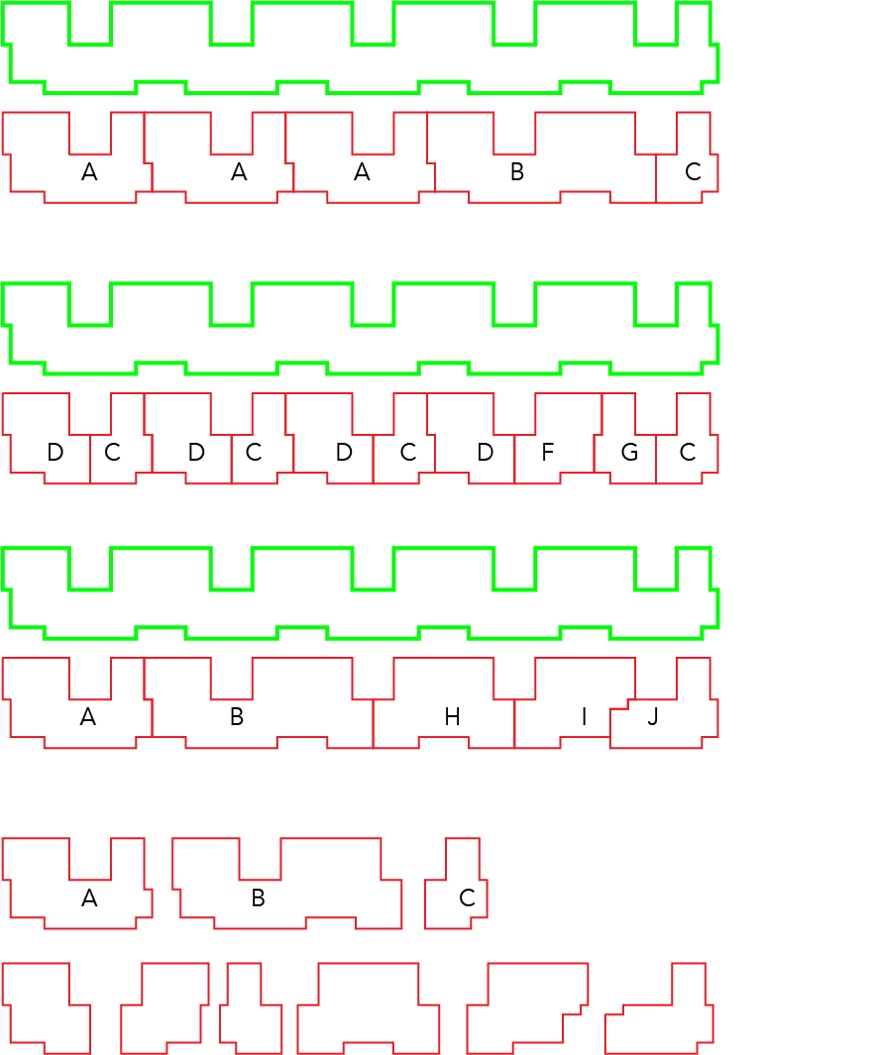
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |

Fig: representation of A shape

Every 2d geometric shape can be drawn with a combination of small squares of unit sine ( same as pixels)

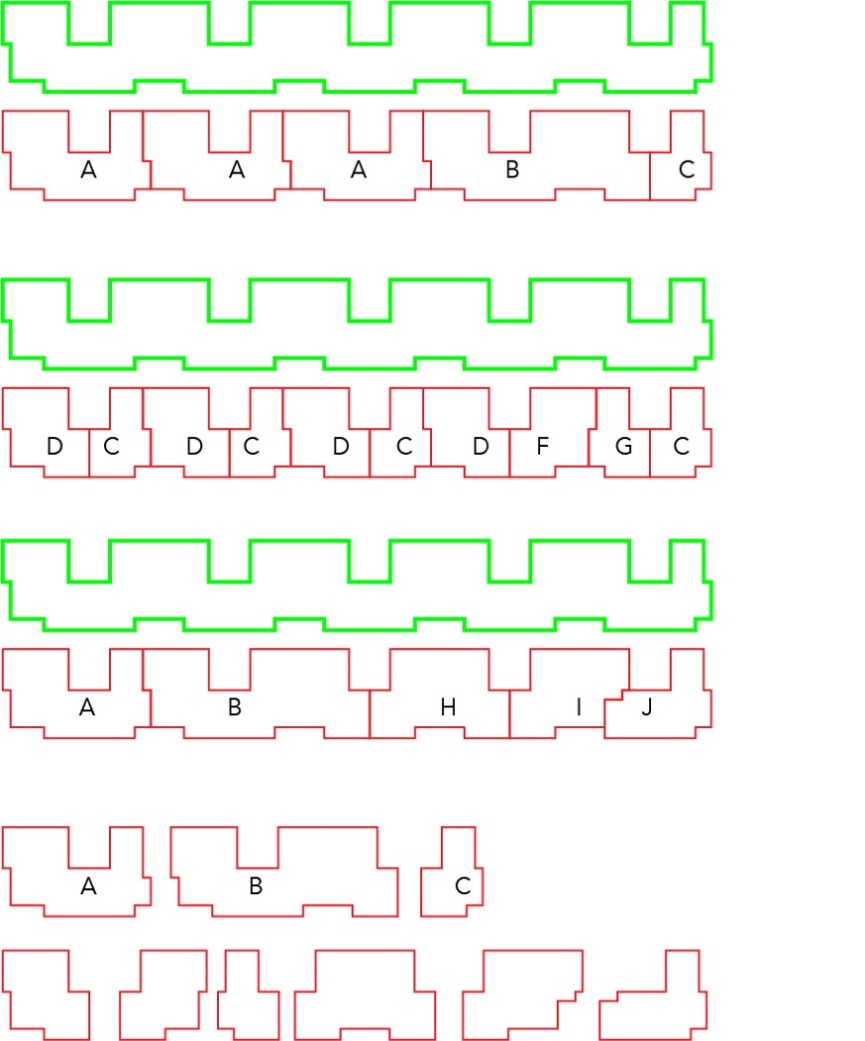
1 represents that the pixel exists in the shape .

0 represents that the pixel doesn’t exist in the shape.



N

1. To find possible ways of inserting A B C kind of shapes into a bigger shape (call it N).



Algorithm

Step 1- define a [5][14]

Step 2- define n [5][100]

Step 3 - x=1

Y=1

z=1

L[100]=0

Step 4 -

For j=0; j<100;j++

For i =0; i <5; i++

For k=0; k<14;k++

{

If a[i][k] = N[i][j+k]

L[j]=1

Else

{ L[j]=0

Break THE K LOOP}

}

Step 5 – for j=0; j<100;j++

If L[j]=1

//Shape A can be placed between j to j+14