Lossless image compression

Created by Pamela Fox

Images are all around us, from application icons to animated GIFs to photos. Image files can take up a lot of space, so computers employ a range of algorithms to compress image files.

For the simplest of images, computers can use a compression algorithm called **run-length encoding (RLE).**

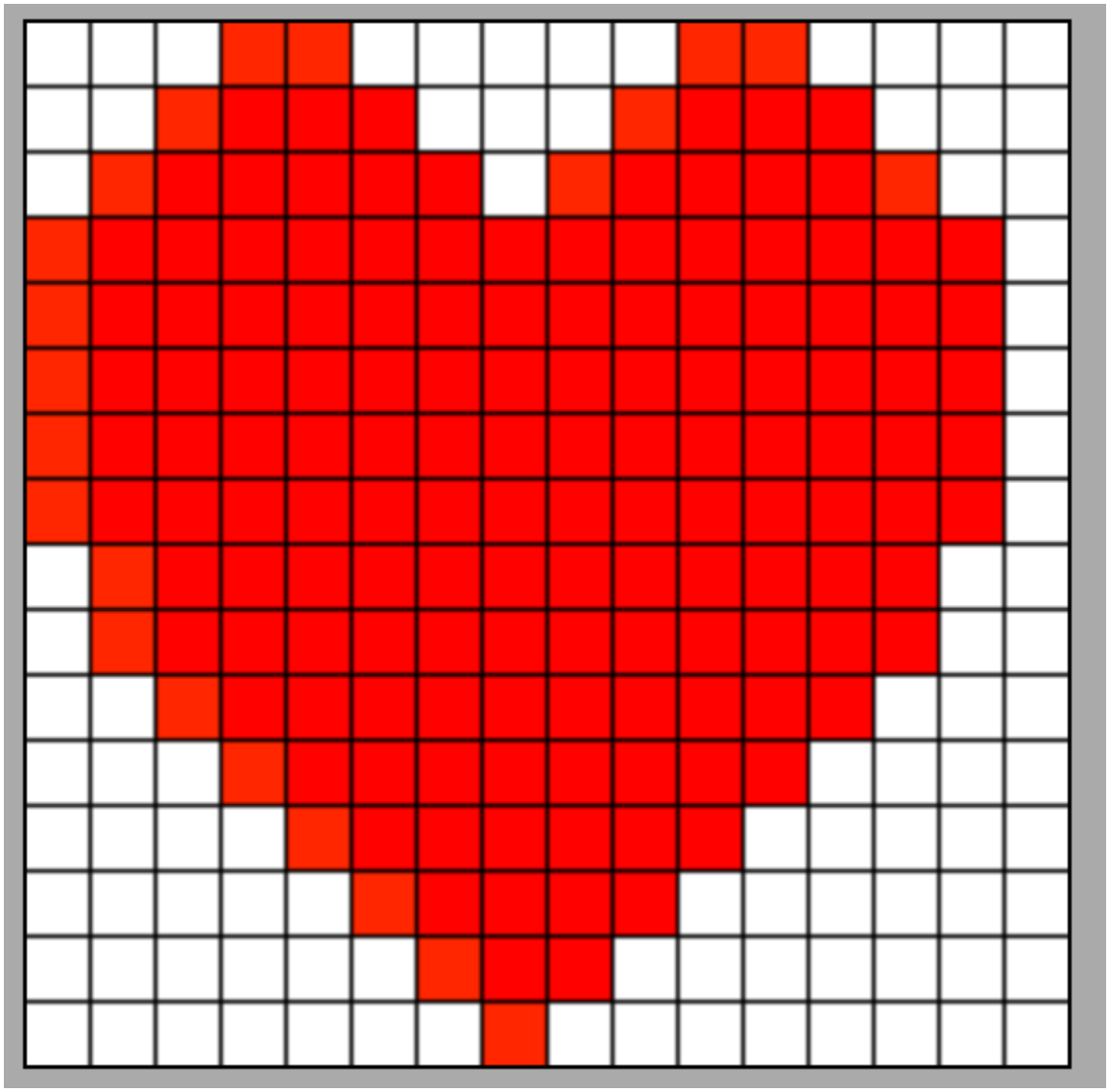
Bitmaps

Before we explore image compression, let's see how we can represent an image in binary *without* any compression.

Here's a simple image, a 16x16 heart icon:

A red heart icon with a white background

Let's zoom in and overlay a grid on top, so that it's easy to see exactly which pixels are red and which pixels are white:



The heart icon is made up of only two colors, red and white, so a computer could represent it in binary by mapping red pixels to 1111 and white pixels to 0000. This is called a **bitmap**, since it's mapping pixels to bits.

Using this method, the heart icon would be represented like so:

```
0001100000110000
0011110001111000
0111111011111100
1111111111111110
```

```
1111111111111110
1111111111111110
1111111111111110
1111111111111110
0111111111111100
0111111111111100
0011111111111000
0001111111110000
0000111111100000
0000011111000000
0000001110000000
0000000100000000
```

Imagine that you had to read the bits above out to someone who was copying them down. After a while, you might say things like "five zeroes" instead of "zero zero zero zero zero". Well, the computer can do that too...

RLE compression algorithm

In run-length encoding, the computer replaces each row with numbers that say how many consecutive pixels are the same color, *always starting with the number of white pixels*.

For example, the first row contains 3 white pixels, 2 red pixels, 5 white pixels, 2 red pixels, then 4 white pixels:

```
0001100000110000
```

This would be represented as follows:

```
3,2,5,2,4
```

The fourth row is interesting because it starts with a *red* pixel. Run-length encodings start with the number of white pixels, so this is how it'd be represented:

```
0,15,1
```

Here's the second row of the bitmap for the heart icon:

```
0011110001111000
```

How would that row be represented in RLE?

RLE decompression

When a computer uses run length encoding, it should be able to perfectly recreate the image from the compressed representation—and so should we, if we follow the computer's strategy.

Let's try it. Here's a representation of a black & white icon using RLE:

```
4, 9, 3
```

```
4, 7, 2, 1, 2
```

```
4, 7, 2, 1, 2
```

```
4, 9, 3
```

```
4, 7, 5
```

```
4, 7, 5
```

```
5, 5, 6
```

```
0, 15, 1
```

```
1, 13, 2
```

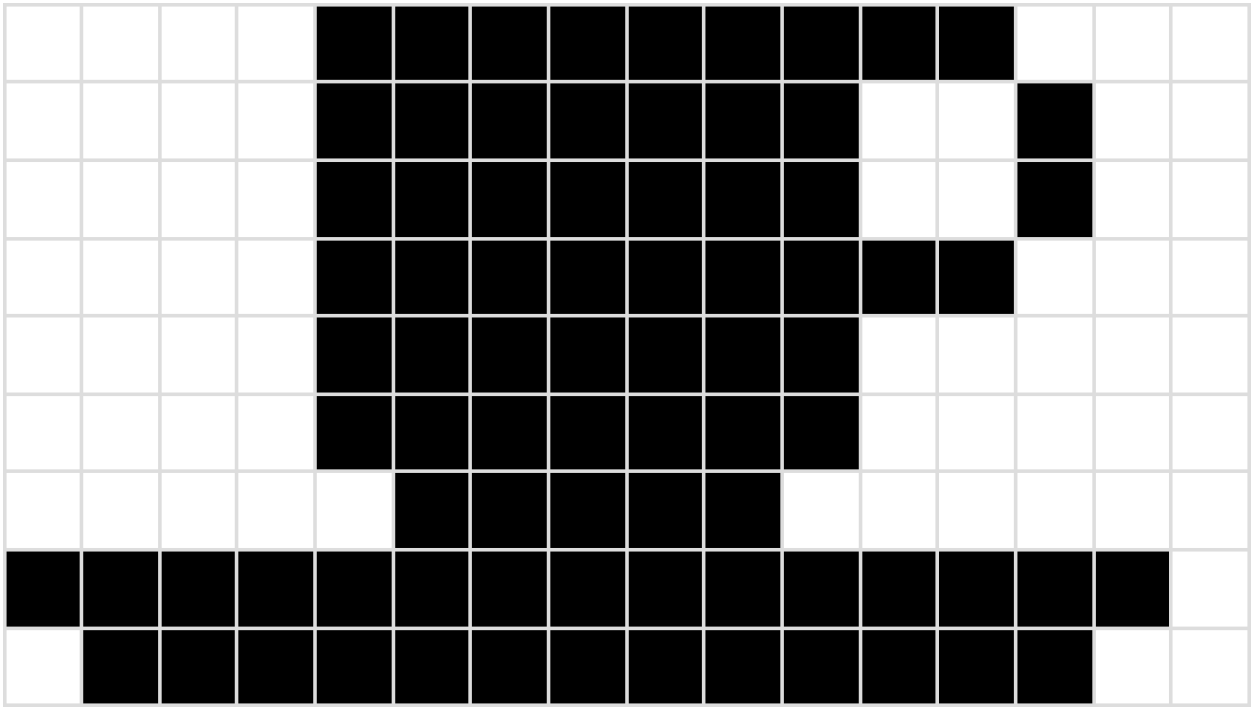
The first row has 4 white pixels, then 9 black pixels, then 3 white pixels. That looks like:

A row of 16 pixels, starting with 4 white, then 9 black, then 3 white.

The next row has 4 white pixels, then 7 black, 2 white, 1 black, and 2 white. That looks like:

A row of 16 pixels, starting with 4 white, then 7 black, then 2 white, then 1 black, and finally 2 white.

When we keep going, the final icon is a cup and saucer:



The following is a compression of a 6x6 black and white icon, using RLE:

```
2,2,2
```

```
2,2,2
0,6
0,6
2,2,2
2,2,2
```

What mathematical symbol does that icon resemble?