EXTRACTION OF LINES AND REGIONS FROM GREY TONE LINE DRAWING IMAGES*

L. T. Watson+
K. Arvind+
R. W. Ehrich+
R. M. Haralick

Tech Report #CS830008

EXTRACTION OF LINES AND REGIONS FROM GREY TONE LINE DRAWING IMAGES*

L. T. Watson[+]

K. Arvind[+]

R. W. Ehrich[+]

R. M. Haralick[□]

[+]Department of Computer Science
[□]Department of Electrical Engineering
Virginia Polytechnic Institute & State University
Blacksburg, VA 24061

June 26, 1983

Abstract. An algorithm is described for extracting lines from grey level digitizations of industrial drawings. The algorithm is robust, non iterative, and sequential, and includes procedures for differentiating shaded areas from lines. Examples are given for complex regions of a typical mechanical drawing.

Key words: adaptive thresholding, grey tone intensity surface, line drawing, line extraction, line tracker, region extraction, region growing.

*This work was supported by Control Data Corporation Grant 82V101.

1. INTRODUCTION

There are many applications such as mapping, drafting, image compression, and computer vision that require robust algorithms for extracting lines and boundaries from images. While a very substantial effort has been made to solve that problem under the adverse image conditions typical in computer vision applications, most popular techniques have many computational disadvantages when image conditions are good. When there is no need to be concerned with serious line fragmentation due to noise, it is possible to deal more directly with the problems of line drawing semantics, including curvature, endpoints, junctions, intersections, variable width, smoothness, straightness, and problems of approximation and encoding.

The particular application that motivated the work reported here is the problem of converting industrial line drawings from hard copy into highly compressed graphical representations involving a small number of primitives such as lines, curves, and regions. This problem vanishes when electronic drafting systems are used to generate the original drawings. However, the reality is that many technical drawings are still created and communicated on paper, and relatively little use is being made of electronically stored representations. As a consequence, revision and updating is difficult to do, and field documentation of large, high-technology systems such as aircraft, space vehicles, buildings, and computers all too frequently consists of containers, files, or even entire rooms full of hard copy reproductions.

The algorithms described in this paper make use of grey level digitizations rather than binary digitizations to maximize the amount of image information available to the interpretation procedures and to minimize the intelligence required of the sensors themselves. The resolution is high, and the images are assumed to be large - at least 1024 by 1024 picture elements. No attempt is made to repair poorly executed drawings. Because of the large data volume and in order to facilitate processing on inexpensive microcomputers, the use of iterative algorithms, including grey level and binary thinning, is precluded, and every possible attempt is made to make processing sequential by scan line. It is also assumed that the line drawings to be processed may contain small, solid, shaded (not textured) regions or regions of parallel adjacent lines too fine to resolve individually. Before presenting specific algorithms, a review of some of the existing approaches for extracting lines from high-contrast digitizations is given.

Currently there are a number of systems for the extracting and encoding of line structured data. These systems frequently contain dedicated hardware such as fast optical scanners that do fast local raster scanning of image data. Black et. al.[1] discuss a general purpose follower for line structured data which is table driven using a PEPR flying spot scanner. Fulford et. al.[2] describe the FASTRAK system, which is an interactive line following digitizer, scanning a reduction of a map with a laser beam. The system depends upon human interaction and intervention. SysCan, a system described by Leberl et. al. (3), features KartoScan, a raster scanner using white light and a CCD array sensor. It is an operational automated system which converts maps and drawings into digital format, and the digital data is edited, stored, retrieved and generally manipulated.

Holdermann et. al. (4), Peleg et. al. (5), Ting et. al. (6), Wang et. al. (7) and Weszka (8) describe general preprocessing techniques such as thresholding and other forms of filtering applied to binary and gray tone images.

Complete systems for the extraction of line structured data from binary and gray scale images are described in the literature. A coding method for the vector representation of engineering drawings is discussed by Ramachandran (9). This algorithm does not distinguish between lines and regions, and moreover, the final form of extracted information, which is an approximation, is not suitable for parametric representation (by splines, for example). Compression ratios of about 35:1 were achieved, which are not very satisfactory for line drawings. Woetzel (10) describes an automatic method for the scanning of cartographic maps and extracting the linework. The method only works on binary images and uses a fast thinning algorithm which may distort the line structure (this is not very critical for maps). Furthermore regions cannot be extracted. Dudani (11) describes a contour following algorithm that can be adapted as a line following algorithm. However, the algorithm does not handle thick lines.

The paper by Freeman (12) is a good tutorial on line drawings and also explains the concept of chain codes. Graham (13) and Huang (14) discuss methods for digital transmission. Maxwell (15) attempts to evolve natural descriptors for line drawings for efficient human-computer communication in the domain of computer graphics. Watson et. al. (16) describe a method for the topographic labelling of gray scale image characteristics such as peaks, ridges, valleys, etc., which can be applied to line drawings.

This paper describes a system which extracts the linework and solid regions from large gray tone images of line drawings using non iterative fast algorithms with minimal storage requirements.

2. ALGORITHMS

A large digitized image produced by a collimated white light scanner was initially blurred by a Gaussian filter and then resampled at every other pixel (2:1 sampling) to produce the input image that was used to develop the algorithms. This image (see Figure 1a) was of size 1024 by 1024, and consisted of thin and thick lines and a few regions with a uniform gray tone value except at the borders. The gray tone intensity surface of a line was ridge-like from the Gaussian filtering. The gray tones in the image ranged from 0 to 255 in value, and average line intensities varied in different parts of the image. Line separation was as low as one pixel in some portions of the image, and the valleys that separated these close lines had gray tone values in between those of the lines and the zero background.

DOUBLE ADAPTIVE THRESHOLDING

Because of the small differences in height between valleys and ridges and variation in average intensities over the image, simple thresholding using a single cutoff value fails to resolve all the lines and suggests the use of local adaptive thresholding. A good local characteristic is the average of pixel values in a neighborhood centered around the pixel being tested. Thresholding can be described as determining which are the object pixels (typically represented by ones in the output image) and which are the background pixels (represented by zeros in the output image). A strategy whereby the threshold cutoff was computed by multiplying some constant times the average gray tone over a square n by n window centered at the current pixel produced very good results. Decreasing the window size has the effect of increasing the resolution, making the thresholding more sensitive to local features. Since low background pixel values have the effect of depressing the computed average, thereby diluting large peaks, only those pixels in the neighborhood which are greater than a certain low threshold are used in the computation of the average. This lower threshold is chosen to be slightly higher than the background pixel values. Thus there are two thresholds, the upper threshold which is computed by multiplying the window average by a cutoff factor, and the lower threshold--hence the name <u>double adaptive thresholding</u>. By choosing a value slightly greater than one for the cutoff factor, only the ridgeline pixels of the convex ridges have values more than the computed cutoff and therefore appear in the output as object pixels. Region pixels do not pass the thresholding condition because of the flat nature of the regions. Here we make a small modification of the algorithm. Pixels which do not pass the thresholding condition but which are greater than a certain region threshold (chosen to be consistent with region pixel values) are marked as region pixels in the output.

Thresholding that normally produces a binary image does not produce lines exactly one pixel thick, making line tracking rather difficult. One course of action would be to thin the binary image produced by the thresholding. A large variety of thinning algorithms are described in the literature, each has its own flavor from the point of view of the accuracy and aesthetic appeal of the skeletons they produce from the original binary image. Often the result is unappealing, as for example when curved lines are reduced to thin jagged lines. The problem is not with the thinning, but with the fact that once we create a binary image from thresholding, the information contained in gray tones of the original image is lost and no distinction can be made between weak and strong object pixels which are all ones. Hence, in the double adaptive thresholding algorithm, all pixels which are above their computed cutoff values (i.e., the object pixels) are represented in the output by their original gray tone values. Region pixels are represented by the negative of their original gray tone values. The reason for retaining the gray tone values for region pixels will be apparent later when we describe region pruning.

The double adaptive thesholding algorithm (DAT) works well for lines when their gray tone intensity surfaces have a convex ridge shape, which can be artificially produced by Gaussian filtering, local averaging, or other known blurring techniques. The DAT algorithm is given below.

```
n := 3  (* window size *)
factor := 1.063
region_threshold := 200
low_threshold := 6
avg := average of all pixel values
       > low_theshold in an n by n
       neighborhood of the current
       pixel
cur_pixel := current input pixel
             value
out_pixel := current output pixel
             value

IF cur_pixel <= ( avg * factor )
THEN IF cur_pixel > region_threshold
     THEN out_pixel := ( -cur_pixel )
```

```
        ELSE out_pixel := 0
    ELSE out_pixel := cur_pixel.
```

REGION DETERMINATION

The DAT, though it handles regions well , has the effect of marking some stray pixels as region pixels. In addition, there may also be holes at the boundaries of the regions. In the next two steps, called 'growing' and 'shrinking', stray region pixels are eliminated and small holes in the regions are filled. Both these steps are based on the connectivity of region pixels. In the 'growing' operation each non-zero non-region pixel is marked as a region pixel if more than k=2 of its eight neighbors are region pixels. This has the effect of filling up small holes and growing the regions. Stray region pixels are not affected because of their low region connectivity. In the 'shrinking' operation, which is complementary to and follows the 'growing' operation, a region pixel is changed to a non-region pixel if less than k=4 of its eight neighbors are region pixels. Stray region pixels are converted to non-region (i.e., line) pixels in this operation. The two steps described above which constitute region determination are shown below.

```
n := 3          (* window size *)
cur_pixel := current input pixel value
out_pixel := current output pixel value
num_neg := number of negative neighbors
           in an n by n neighborhood of
```

```
                the current pixel not counting
                the current pixel.
    const1 := 3
    const2 := 4

    Grow:
    IF num_neg >= const1 AND cur_pixel > 0
    THEN out_pixel := ( -cur_pixel ) (* region *)
    else out_pixel := ( cur_pixel ).

    Shrink: (* with output of Grow as input *)
    IF cur_pixel < 0 AND num_neg >= const2
    THEN out_pixel :=  cur_pixel  (* region *)
    ELSE out_pixel := abs ( cur_pixel )
```

<u>REGION EXTRACTION</u>

The region determination step is followed by region extraction and line extraction. The regions can be represented by following and encoding their contours. Dudani (11) describes a method for region extraction using boundary following. Alternatively, simple run length coding or one of its more complex variations could be used. Such schemes work well because of good correlation of pixel runs between adjacent scan lines in the regions.

<u>LINE EXTRACTION</u>

Using a simple thinning algorithm to reduce the line width to one pixel followed by a simple line tracker is unsatisfactory for the reasons given above. A more complicated line tracker which tracks the ridges of the gray tone intensity surfaces of the lines is described below in two steps.

1) Finding the starting pixel for a new line.

The image is scanned from left to right and top to bottom. Thus, line by line each pixel is examined to check whether it is a candidate for starting a line. The following conditions must be satisfied by an unmarked pixel (called the candidate pixel) in order to be a starting point of a line:

a) Its value is more than a certain threshold to indicate which pixel values are background and which are not.

b) Its value is more than a factor (a good value is 0.7) times the average of its non-zero marked or unmarked eight neighbors. (When a pixel is tracked it is marked.)

c) Its value is more than a factor (0.9) times the average of its marked eight neighbors. This condition ensures that the pixel is strong compared to nearby tracked vectors.

d) Let pixel P1 be the unmarked, untested eight neighbor of the candidate pixel with maximum gray tone value. This selection implies a probable direction for a new line. Directions of the eight neighbors with respect to the candidate pixel are labeled as follows:

Let d be the direction of P1 with respect to the candidate pixel. P1 must also satisfy conditions a) through c).

e) Let P11, P12, P13 be the neighbors in the directions d-1, d, d+1 (modulo 8) respectively from P1. At least one of these pixels P1j must not be marked, must not have a marked eight neighbor in the directions s-1 and s+1 from P1 (s is the direction of P1j from P1), and must satisfy conditions a) through c). Otherwise consider P1 tested, and attempt to satisfy d) and e) with a different P1, until all possibilities have been tested.

If conditions a) through e) are satisfied by the candidate pixel, then it is marked along with P1 and the line tracker is invoked to continue tracking from pixel P1.

2) Tracking a line .

Let P1 represent the previously marked pixel, P2 the current marked pixel, and P3 the next pixel sought by the line tracker. Let d1 be the direction to P1 from the pixel marked prior to P1, d2 the direction from P1 to P2, and d3 the direction from P2 to P3. From among the unmarked neighbors of P2 in the directions d2-2 (mod 8), d2-1 (mod 8), d2, d2+1 (mod 8), d2+2 (mod 8), P3 is chosen as the pixel (if it exists) with maximum gray tone value. P3 is marked if it satisfies the conditions:

a) P3's gray tone value is greater than a certain threshold which is slightly more than typical background values.

b) P3 has no marked eight neighbors in the directions d3-1 (mod 8) and d3+1 (mod 8) from P2.

c) min {abs(d3-d1), 8-abs(d3-d1)} <= 2.

If P3 does not exist or does not satisfy conditions a) through c), then the current vector is terminated and a starting pixel for the next line is sought. If no such pixel is found, the line tracker is terminated.

The algorithm for the line tracker is shown below.

```
STEP 1. Determination of the starting point.

cur_pix := current pixel
VALUE ( cur_pix ) :== value of current pixel
cur_val := VALUE ( cur_pix )
n := 3   (* window size *)
avg_marked := average of all the marked pixel
              values in an n by n neighborhood
              of the current pixel
avg_nonzero := average of all the non-zero
               pixel values in an n by n
               neighborhood of the current pixel
cutoff1 := 0.9 * avg_marked
cutoff2 := 0.7 * avg_nonzero
CUTOFF ( cur_pix ) :== MAX ( 50, cutoff1, cutoff2 )

IF cur_pix is not marked AND
```

```
   cur_val > CUTOFF ( cur_pix )
THEN WHILE ( some unmarked neighbor of cur_pix
           not tested ) DO

    BEGIN
     temp_pix := unmarked, untested neighbor of
                  cur_pix with maximum value
     temp_val := VALUE ( temp_pix )
     IF temp_val > CUTOFF ( temp_pix )
     THEN
     BEGIN
      cur_dir := direction from cur_pix to
                  temp_pix
     pix1 := pixel in dir. cur_dir-1 from temp_pix
     pix2 := pixel in dir. cur_dir from temp_pix
     pix3 := pixel in dir. cur_dir+1 from temp_pix
     IF VALUE ( pix1 ) > CUTOFF ( pix1 ) AND pix1
              not marked AND pix1 has no adjacent
              marked neighbors (* adjacent means
              in the direction from temp_pix +1
              or -1 *)
        OR VALUE ( pix2 ) > CUTOFF ( pix2 ) AND pix2
              not marked AND pix2 has no adjacent
              marked neighbors
        OR VALUE ( pix3 ) > CUTOFF ( pix3 ) AND pix3
              not marked AND pix3 has no adjacent
              marked neighbors
    THEN mark cur_pix, temp_pix, and call tracker
```

```
      END
      ELSE designate temp_pix as tested
     END (* WHILE *)

STEP 2. Tracking the vector.

cur_dir := current direction
prev_dir := previous direction
threshold := 50
next_pix := unmarked neighbor in direction d from
            cur_pix, abs ( d-cur_dir (mod 8) ) <= 2,
            with maximum value.
IF next_pix exists
THEN
BEGIN
 next_dir := direction from cur_pix to next_pix
 next_val := VALUE ( next_pix )
END
ELSE next_val := 0
IF next_val > threshold AND
   abs ( next_dir-prev_dir (mod 8) ) <= 2 AND
   neighbors of next_pix in directions
   next_dir-1, next_dir+1 from cur_pix are not
   marked
THEN
BEGIN
 mark next_pix
 cur_pix := next_pix
```

```
    prev_dir := cur_dir
    cur_dir := next_dir
    continue tracking the current vector
  END
  ELSE
  BEGIN
    terminate tracking the current vector
    determine the starting point of next vector
  END.
```

The line tracker, in addition to marking the pixels in the image, also outputs the new chain code or absolute coordinates of the pixels. The output format depends on the user's requirements. Skiansky et. al. (17) discuss a method for fast polygonal approximation of the pixels as the pixels are being tracked. With their scheme a straight line would be represented by its two end points.

3. RESULTS

Fig. 1a shows one of the images used for testing the algorithms. Fig. 1b shows a close up view of the spool of wire in the same image. This is the most crucial part of the image since the line separation here is very low, and the lines merge at the edge of the coil to form textured regions. The image shown in Fig. 1a is large, nearly 2048 by 2048 pixels. The lines are sparse in most of the image except in the spool of wire. The 1024 by 1024 test image (to which the algorithms were applied) was obtained by blurring the original image using a Gaussian filter of standard deviation 1.3, and then resampling the filtered image at every other pixel. Compression ratios higher than 2:1 cause Moire patterns in the spool of wire, attributed to the classical problem of aliasing.

The test image corresponded very closely with what would have been produced by some industrial imaging hardware already in place. The larger image was taken as the starting point simply for the purposes of comparison and testing.

Fig. 2a displays the gray tone values of a portion of the spool of wire, where the upper portion resembles a "bobbin". Note the presence of a considerable amount of noise, some of which was present in the original image, and the rest was added by the filtering. A well known iterative sharpening algorithm EHNUM [19] (an iterative algorithm which replaces each pixel's gray tone value by the nearest of the max or min of its eight neighbors' gray tone values) was applied to this noisy image. The result is illustrated in Fig. 2b. Although the image in Fig. 2b does have a cleaner, sharper appearance, the lines are now rather jagged because the gray tone values of many insignificant pixels have now been raised. There is a more serious defect, namely, although there are only two vertical lines at the top in the input image Fig. 2a, the sharpening algorithm EHNUM has found three.

Fig. 3 shows the result of applying the topographic labeling algorithm [16] to the image in Fig. 2a. The algorithm fits a two-dimensional cubic polynomial to the gray tone values in a square 5 by 5 window (this window size can be changed) centered around each pixel. This polynomial represents a surface which best fits, in a discrete least squares sense, the pixel data in the window. The topography of the surface at the position of the central pixel is now determined using partial derivatives of the polynomial. By tuning threshold parameters in the algorithm a good representation of the ridges was obtained.

Fig. 4a shows a dense part of the coil. The lines are not well resolved and there is a lot of noise between the lines. The topographic labeling algorithm was applied to this image with the same parameter values that were used to obtain the image in Fig. 3 (see Fig. 4b). The detection of the gray tone surface ridges, which clearly exist in Fig. 4a, is obviously extremely poor in Fig. 4b. This illustrates the sensitive nature of the algorithm to the threshold parameter values, making it unsuitable for the present application.

The DAT was used with very good results on the image in Fig. 4a. Fig. 5 shows this result, in which the lines have been accentuated very well. The negative pixels represent the regions on which the GROW and SHRINK algorithms were applied to finally get the regions shown in Fig. 6. These extracted regions were quite consistent with the perceived regions in the original unprocessed image.

The line tracker was now applied to the image in Fig. 5 after the regions (shown in Fig. 6) had been removed. Pixels which the line tracker marked are shown with negative values in Fig. 7. For further illustration, the same tracked image is shown in Fig. 8, but with asterisks used to indicate the pixels marked by the tracker. Observe that the tracked lines very faithfully represent the lines in the original unprocessed image.

4. CONCLUSION.

The difficulties associated with real digitized line drawing images, as opposed to artificially generated images, are significant. Many papers on line drawings have used binary images as their starting point, but the present work shows that producing a good binary image from a real digitized gray tone image is highly nontrivial. Any practical production algorithm must clearly begin with noisy gray tone images. Standard thinning, sharpening, and medial axis transformations were tried, and (despite occasional exemplary performances) none were found to be uniformly good.

For certain industrial applications it may not be practical or economic to connect the imaging equipment to mainframe computers with high speed transmission lines. Thus the low level processing (extraction of the lines and regions) must be done with limited local compute power and storage. The DAT algorithm presented here meets these requirements by being reasonably cheap and non iterative, although several sequential passes through the image are required. Numerous experiments on different types of line drawings also strongly suggest that a good algorithm must be adaptive, and the adaptive nature of the DAT is crucial to its success.

The overall problem being addressed here for real digitized gray tone images of line drawings consists of (1) recognition and extraction of lines and solid regions (textured regions are not considered), (2) the compression and transmission of the line and region data, and (3) the high level representation (e.g., as graphics primitives) of the line drawing (lines, curves, regions). This paper represents a solution to (1) requiring only limited local compute power and storage. The next goals are an economic solution to (2) using slow transmission speeds (300 baud), and a powerfull and sophisticated high level encoding of the line drawing as graphical primitives in, e.g., CDC's TUTOR system.

5. REFERENCES

1. W. T. Black, T. P. Clement, J. F. Harris, B. Llewellyn and G. Preston, A general purpose follower for line-structured data, Unpublished conference proceedings.

2. N. C. Fulford, The fastrak automatic digitizing system, Pattern Recognition 14, 65-74 (1981).

3. F. W. Leberl and D. Olson, Raster scanning for operational digitizing of graphical data, Photogramm. Eng. and Remote Sensing 48, 615-627 (1982).

4. F. Holdermann and H. Kazmierczak, Preprocessing of gray scale pictures, Comput. Graphics and Image Process. 1, 66-80 (1972).

5. S. Peleg and A. Rosenfeld, A min-max medial axis transformation, IEEE Trans. Pattern Anal. and Mach. Intell. PAMI-1, 88-89 (1979).

6. D. Ting and B. Prasada, Digital processing techniques for encoding of graphics, Proc. IEEE 68, 756-767 (1980).

7. S. Wang, A. Y. Wu and A. Rosenfeld, Image approximation from gray scale 'medial axes' , IEEE Trans. Pattern Anal. and Mach. Intell. PAMI-3, 687-697 (1981).

8. J. Weszka, A survey of threshold selection techniques, Comput. Graphics and Image Process. 7, 259-265 (1978).

9. K. Ramachandran, A coding method for vector representation of engineering drawings, Proc. IEEE 68, 813-817 (1980).

10. G. Woetzel, A fast and economic scan-to-line conversion algorithm, Comput. Graphics 12, 125-129 (1978).

11. S. A. Dudani, Region extraction using boundary following, Unpublished conference proceedings.

12. H. Freeman, Computer processing of line drawings, Comput. Surv. 6, 59-97 (1974).

13. D. N. Graham, Image transmission by two dimensional contour coding, Proc. IEEE 55, 336-346 (1967).

14. T. S. Huang, Digital transmission of halftone pictures, Comput. Graphics and Image Process. 3, 195-202 (1974).

15. P. C. Maxwell, The perception and description of line drawings by computer, Comput. Graphics and Image Process. 1, 31-46 (1972).

16. L. T. Watson, T. J. Laffey and R. M. Haralick, Topographic classification of digital image intensity surfaces using generalized splines and the discrete cosine transformation, Comput. Vision, Graphics and Image Process., to appear.

17. J. Skiansky and V. Gonzales, Fast polygonal approximation of digitized curves, Pattern Recognition 12, 327-331 (1981).

18. H. P. Kramer and J. B. Bruckner, Iterations of a non-linear transformation for enhancement of digital images, Pattern Recognition 7, 53-58, (1975).

LIST OF FIGURE CAPTIONS

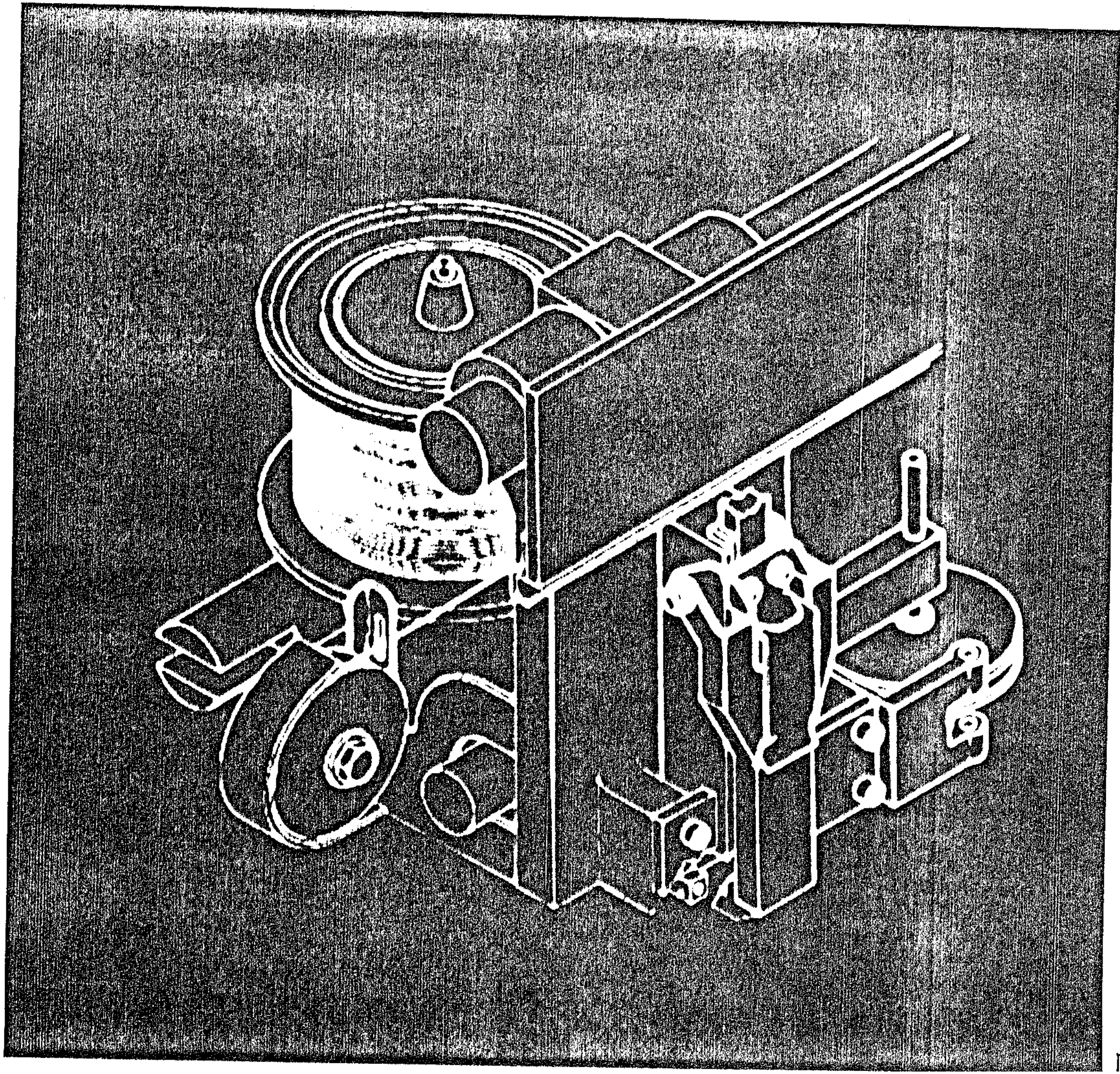
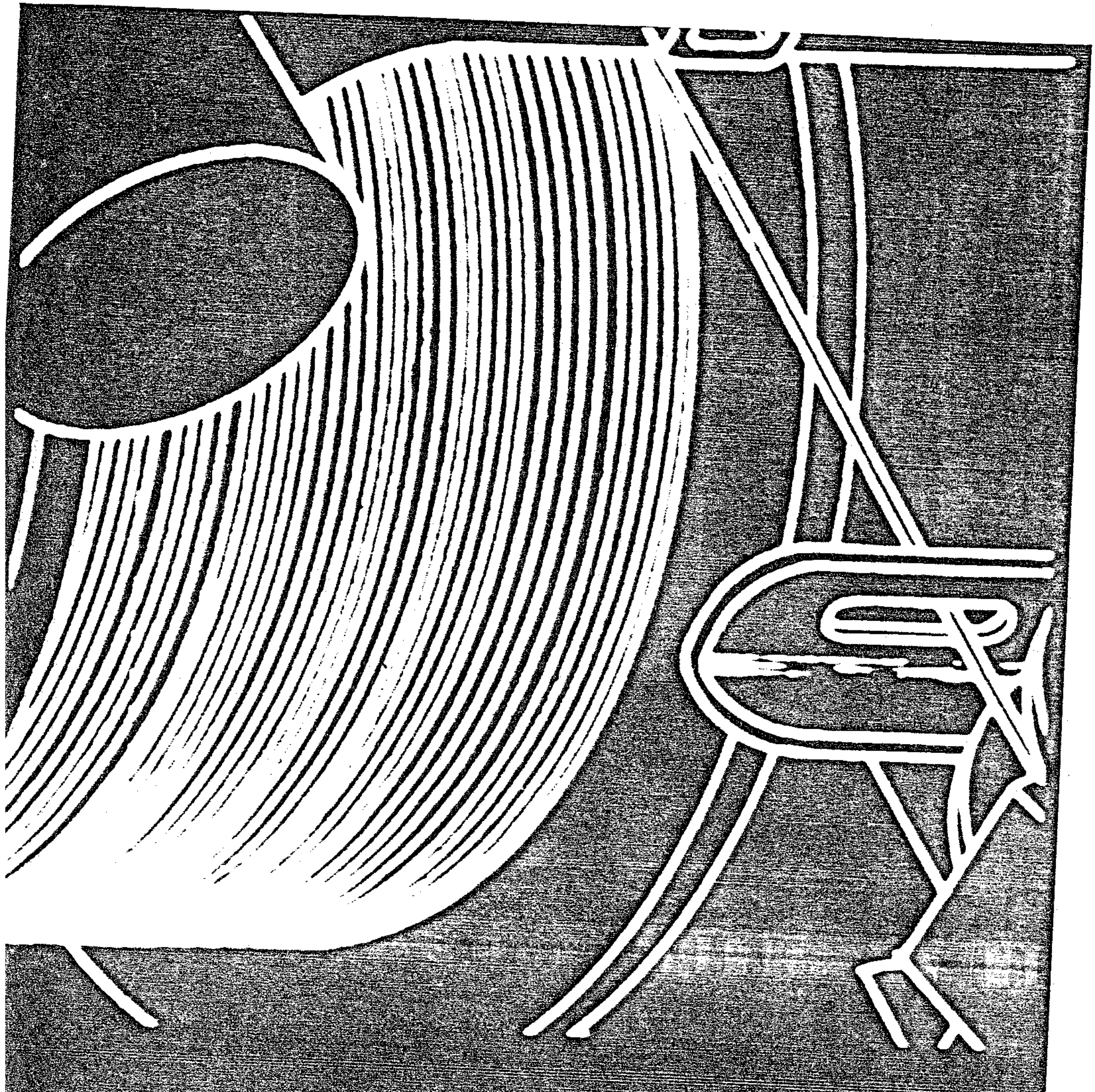


Fig. 1a



0	0	0	0	0	5	93	78	12	20	49	6	0	0	0	0	0	0
0	0	0	0	0	4	86	77	6	24	72	10	0	0	0	0	0	0
0	0	0	0	0	4	81	71	5	26	82	12	0	0	0	0	0	0
0	0	0	0	0	4	75	51	16	36	107	19	0	0	0	0	0	0
0	0	0	0	0	4	74	58	11	36	129	26	0	0	0	0	0	0
0	0	0	0	0	4	78	67	34	35	89	16	0	0	0	0	0	0
0	0	0	0	0	4	80	67	78	41	53	6	0	0	0	0	0	0
0	0	0	0	0	5	89	84	92	51	62	8	0	0	0	0	0	0
0	0	0	0	0	5	91	101	118	59	85	14	0	0	0	0	0	0
0	0	0	0	0	4	84	88	125	61	92	16	0	0	0	0	0	0
0	0	0	0	0	4	71	64	130	66	95	14	0	0	0	0	0	0
0	0	0	0	0	3	57	53	127	62	78	10	0	0	0	0	0	0
0	0	0	0	0	3	46	47	136	64	85	10	0	0	0	0	0	0
0	0	0	0	0	4	74	71	147	69	104	16	0	0	0	0	0	0
0	0	0	0	0	5	87	91	151	70	107	17	0	0	0	0	0	0
0	0	0	0	0	5	95	106	155	72	109	16	0	0	0	0	0	0
0	0	0	0	0	5	88	103	153	74	129	23	0	0	0	0	0	2
0	0	0	0	0	5	90	106	159	76	143	28	0	0	0	0	1	31
0	0	0	0	0	5	95	112	162	75	135	25	0	0	0	0	22	138
0	0	0	0	0	5	94	107	164	75	129	23	0	0	0	12	101	218
5	0	0	0	0	5	93	100	163	77	139	26	0	1	15	97	209	187
84	14	2	0	0	5	99	109	166	78	143	27	1	21	105	214	201	74
202	122	60	17	3	7	106	115	168	85	144	30	33	132	221	199	79	9
120	181	200	145	74	46	145	131	179	117	174	98	165	227	186	73	9	10
9	41	125	195	208	192	221	189	219	204	237	227	224	147	48	8	24	105
1	1	13	44	107	176	223	238	245	242	235	193	102	25	9	44	142	222
32	6	1	1	7	30	74	113	125	116	94	41	12	31	100	191	225	164
171	100	42	14	3	1	3	5	6	6	10	26	72	168	222	201	114	32
166	205	196	141	77	32	21	24	33	49	109	171	214	204	130	56	10	1
25	76	158	208	211	183	165	172	194	210	233	216	152	66	12	2	0	0
0	3	20	50	107	143	179	189	184	181	152	74	19	3	0	0	0	0
0	0	0	1	5	13	27	31	29	27	17	4	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	4	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
148	94	44	20	6	1	0	0	0	0	0	0	0	0	0	0	0	0
196	214	200	159	101	43	15	4	1	0	0	0	0	0	0	0	0	0
72	99	132	184	212	196	144	90	40	15	3	1	0	0	0	0	0	0
182	133	83	69	95	137	192	209	192	142	83	32	8	2	0	0	0	0
149	199	212	189	136	79	66	90	131	183	204	177	119	61	23	6	1	0
25	50	107	155	207	210	176	114	60	56	91	152	199	204	164	98	37	9
125	77	36	26	62	113	157	199	199	161	105	63	63	118	184	211	184	121
194	211	178	117	60	32	27	55	117	176	196	187	135	73	61	96	156	202
45	108	163	200	202	172	107	48	29	32	51	119	186	198	172	114	65	65
10	10	22	49	106	158	195	197	159	91	40	22	41	94	166	207	193	129
128	93	37	10	9	21	54	113	167	195	186	135	87	43	35	73	130	189
158	188	170	116	82	40	16	9	25	58	119	179	201	186	127	64	32	47
21	47	102	155	188	185	133	67	35	15	12	31	72	147	199	202	157	80
27	9	8	22	48	108	166	188	187	133	84	35	11	18	48	114	184	204
169	117	61	26	7	8	26	58	121	177	197	177	113	40	14	11	37	90
125	176	187	163	96	43	26	10	10	30	66	130	185	173	133	67	29	11
11	30	67	118	158	174	169	119	62	27	9	13	43	105	170	189	171	97
57	15	7	10	27	66	126	186	199	171	106	39	10	10	28	68	135	182
174	131	104	64	31	18	14	40	91	147	188	171	113	75	32	10	15	45
60	104	151	173	170	140	86	40	15	20	44	93	113	134	164	113	46	10

Fig. 2a

0	0	0	0	0	0	130	130	0	0	129	0	0	0	0	0	0	0
0	0	0	0	0	0	130	130	0	0	129	0	0	0	0	0	0	0
0	0	0	0	0	0	136	136	0	0	129	0	0	0	0	0	0	0
0	0	0	0	0	0	147	147	0	0	129	0	0	0	0	0	0	0
0	0	0	0	0	0	151	151	0	0	129	0	0	0	0	0	0	0
0	0	0	0	0	0	151	155	0	0	129	0	0	0	0	0	0	0
0	0	0	0	0	0	151	155	155	0	129	0	0	0	0	0	0	0
0	0	0	0	0	0	155	130	159	0	143	0	0	0	0	0	0	0
0	0	0	0	0	0	155	162	162	0	143	0	0	0	0	0	0	0
0	0	0	0	0	0	155	46	164	3	143	0	0	0	0	0	0	0
0	0	0	0	0	0	151	46	164	5	143	0	0	0	0	0	0	0
0	0	0	0	0	0	151	46	166	5	143	0	0	0	0	0	0	0
0	0	0	0	0	0	136	46	168	5	144	0	0	0	0	0	0	0
0	0	0	0	0	0	145	46	179	11	174	0	0	0	0	0	0	0
0	0	0	0	0	0	221	46	219	11	237	0	0	0	0	0	0	0
0	0	0	0	0	0	238	46	245	35	242	0	0	0	0	0	0	0
0	0	0	0	0	0	245	46	245	41	245	0	0	0	0	0	0	0
0	0	0	0	0	0	245	46	245	51	245	0	0	0	0	0	0	0
0	0	0	0	0	0	245	46	245	59	245	0	0	0	0	0	0	237
0	0	0	0	0	0	245	46	245	61	245	0	0	0	0	0	0	245
0	0	0	0	0	0	245	46	245	62	245	0	0	0	0	0	245	245
0	0	0	0	0	0	245	46	245	62	245	0	0	0	0	245	245	6
245	245	0	0	0	0	245	47	245	64	245	0	0	245	245	245	6	6
245	245	245	245	0	0	245	71	245	69	245	0	245	245	245	3	6	6
1	1	245	245	245	245	245	245	245	245	245	245	245	245	1	3	3	6
1	1	1	1	1	245	245	245	245	245	245	245	1	1	1	1	233	233
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	233	233	233
233	1	1	1	1	1	1	1	6	6	1	1	1	233	233	233	0	0
233	233	233	233	1	1	1	1	1	1	1	233	233	233	233	0	0	0
0	0	233	233	233	233	233	233	233	233	233	233	233	0	0	0	0	0
0	0	0	0	0	233	233	233	233	233	233	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
214	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
214	214	214	214	0	0	0	0	0	0	0	0	0	0	0	0	0	0
56	56	56	214	214	214	214	0	0	0	0	0	0	0	0	0	0	0
212	56	56	56	56	56	214	214	214	214	0	0	0	0	0	0	0	0
212	212	212	212	56	56	56	56	214	214	214	214	212	0	0	0	0	0
25	22	22	212	212	212	212	56	56	56	56	214	212	212	212	0	0	0
211	22	22	22	22	22	212	212	212	212	56	56	56	56	212	211	211	211
211	211	211	211	22	22	22	22	212	212	212	212	212	56	56	56	211	211
9	9	211	211	211	211	22	22	22	22	22	212	212	210	210	56	56	56
9	9	9	9	211	211	211	211	211	22	22	22	22	22	210	210	207	207
197	9	9	9	9	9	9	211	211	211	211	211	22	22	22	22	207	207
197	197	197	197	9	9	9	9	9	9	211	211	204	204	204	22	22	22
7	7	197	197	197	197	197	9	9	9	9	9	9	204	204	204	204	22
7	7	7	7	7	197	197	197	197	197	9	9	9	9	9	204	204	204
199	199	7	7	7	7	7	7	197	197	197	197	197	9	9	9	9	9
199	199	199	199	199	7	7	7	7	7	7	197	197	197	197	9	9	9
7	7	7	199	199	199	199	199	7	7	7	7	7	197	197	197	197	197
7	7	7	7	7	7	199	199	199	199	199	7	7	7	7	8	197	197
211	211	211	7	7	7	7	7	7	199	199	199	199	199	7	8	9	9
1	211	211	211	211	211	7	7	7	7	7	199	199	199	199	199	9	9

1	1	1	1	9	1	*7*	*7*	9	1	1	1	1	1	1	1	1	1
1	1	1	1	9	1	*7*	*7*	9	1	*7*	1	9	1	1	1	1	1
1	1	1	1	9	1	*7*	*7*	9	1	*7*	1	9	1	1	1	1	1
1	1	1	1	9	1	*7*	*7*	9	1	*7*	1	9	1	1	1	1	1
1	1	1	1	9	1	*7*	*7*	9	1	*7*	1	9	1	1	1	1	1
1	1	1	1	9	1	*7*	*7*	9	1	*7*	1	9	1	1	1	1	1
1	1	1	1	9	1	*7*	*7*	1	1	*7*	1	9	1	1	1	1	1
1	1	1	1	9	1	1	*7*	1	1	1	1	9	1	1	1	1	1
1	1	1	1	9	1	1	*7*	1	*7*	1	1	9	1	1	1	1	1
1	1	1	1	9	1	1	*7*	1	*7*	1	1	9	1	1	1	1	1
1	1	1	1	9	1	1	*7*	1	*7*	1	1	9	1	1	1	1	1
1	1	1	1	9	1	1	*7*	1	*7*	1	1	9	1	1	1	1	1
1	1	1	1	9	1	1	*7*	1	*7*	1	1	9	1	1	1	1	1
1	1	1	1	9	1	1	*7*	1	*7*	1	1	9	1	1	1	1	1
1	1	1	1	9	1	1	*7*	1	*7*	1	1	9	1	1	1	1	1
1	1	1	1	9	1	1	*7*	1	*7*	1	1	9	1	1	1	9	9
1	1	1	1	9	9	1	*7*	1	*7*	1	1	9	1	1	9	9	9
1	1	1	1	9	9	1	*7*	1	*7*	1	1	9	1	1	9	9	*7*
1	1	1	1	9	9	1	*7*	1	*7*	1	1	9	9	9	9	1	*7*
9	9	1	1	9	9	1	*7*	1	*7*	1	1	9	9	9	1	*7*	*7*
9	9	9	9	9	9	1	*7*	1	*7*	1	1	9	9	1	*7*	*11*	*11*
7	1	9	9	10	9	1	*7*	1	*7*	9	9	9	1	*7*	*7*	*7*	1
7	*7*	*7*	1	10	9	9	*11*	9	*11*	9	9	9	*7*	*7*	*7*	1	9
7	*7*	*7*	*7*	*11*	1	1	*7*	1	*7*	1	*7*	*11*	*11*	*7*	1	9	9
9	1	*7*	*7*	*11*	*7*	*7*	8	*7*	8	*7*	*7*	*11*	*7*	9	9	9	1
9	9	9	1	*7*	*7*	*7*	8	*7*	*7*	*7*	*7*	9	9	9	9	*7*	*7*
9	9	9	9	9	9	9	1	1	9	9	9	9	9	1	*7*	*7*	*7*
7	*7*	1	9	9	9	9	9	1	9	9	9	1	*7*	*7*	*7*	*7*	1
7	*7*	*7*	*7*	*7*	1	9	9	9	9	1	*7*	*7*	*7*	*7*	1	9	9
1	*7*	*7*	*7*	*7*	*7*	*7*	*7*	*7*	*7*	*7*	*7*	*7*	*7*	1	9	9	9
9	9	9	1	*7*	*7*	*7*	*7*	*7*	*7*	*7*	1	1	9	9	9	9	1
9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	1	1	1
1	1	1	9	9	9	9	9	9	9	9	9	9	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
9	9	9	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
9	9	9	9	9	9	1	1	1	1	1	1	1	1	1	1	1	1
7	*7*	1	9	9	9	9	9	9	1	1	1	1	1	1	1	1	1
7	*7*	*7*	*7*	*7*	1	9	9	9	9	9	9	1	1	1	1	1	1
1	1	*7*	*7*	*7*	*7*	*7*	*7*	1	9	9	9	9	9	1	1	1	1
7	1	1	1	1	1	*7*	*7*	*7*	*7*	*7*	1	9	9	9	9	9	1
7	*7*	*7*	*7*	1	1	1	1	1	*7*	*7*	*7*	*7*	1	1	9	9	9
9	9	1	1	*7*	*7*	*7*	1	1	1	1	*7*	*7*	*7*	*7*	*7*	1	9
1	9	9	9	9	1	1	*7*	*7*	*7*	1	1	1	1	*7*	*7*	*7*	*7*
7	*7*	*7*	1	9	9	9	9	1	*7*	*7*	*7*	1	1	*7*	*7*	*7*	*7*
9	1	*7*	*7*	*7*	*7*	1	9	9	9	9	1	1	*7*	1	1	1	1
9	9	9	9	1	*7*	*7*	*7*	*7*	1	9	9	9	1	1	*7*	1	1
7	1	9	9	9	9	9	1	*7*	*7*	*7*	*7*	1	9	9	9	1	*7*
7	*7*	*7*	*7*	1	9	9	9	9	9	1	*7*	*7*	*7*	1	9	9	9
9	1	*7*	*7*	*7*	*7*	*7*	1	9	9	9	9	1	*7*	*7*	*7*	*7*	1
9	9	9	9	1	*7*	*7*	*7*	*7*	*7*	1	9	9	9	9	1	*7*	*7*
7	*7*	1	9	9	9	9	1	*7*	*7*	*7*	*7*	1	9	9	9	9	1
7	*7*	*7*	*7*	*7*	1	9	9	9	9	1	*7*	*7*	*7*	*7*	1	9	9
9	9	1	*7*	*7*	*7*	*7*	*7*	1	9	9	9	9	1	*7*	*7*	*7*	1
1	9	9	9	9	1	*7*	*7*	*7*	*7*	1	1	9	9	9	1	*7*	*7*
7	*7*	*7*	1	9	9	9	9	1	*7*	*7*	*7*	*11*	1	9	*11*	9	9
7	*7*	*7*	*7*	*7*	*7*	1	9	9	9	9	1	*11*	*11*	*7*	1	9	9

192	204	120	62	145	215	141	66	154	204	120	154	187	95	96	214	185	103
151	172	214	148	68	132	209	147	71	158	194	119	157	195	97	133	222	152
223	169	152	214	176	82	111	207	156	72	152	191	99	154	190	96	160	221
183	220	188	171	217	181	79	114	206	152	72	158	164	96	192	177	98	198
180	165	226	209	175	216	182	75	108	201	148	79	175	196	198	235	153	116
205	213	212	228	220	194	224	185	65	102	205	152	91	195	242	243	232	137
80	199	233	197	216	233	207	210	162	58	113	211	150	109	215	240	249	208
49	77	182	228	202	229	231	183	209	147	75	136	209	151	141	230	254	244
177	71	55	163	231	234	236	235	220	227	188	79	118	212	157	161	242	254
182	192	107	57	145	232	233	246	245	230	240	170	83	156	215	131	189	249
46	152	210	127	60	132	227	248	252	251	246	228	190	101	187	202	135	215
45	39	140	212	149	60	131	230	248	240	236	189	224	169	116	208	186	149
184	75	34	109	199	162	76	130	224	244	251	233	198	227	158	143	224	178
197	200	91	26	86	200	186	92	116	225	254	253	236	248	237	178	210	238
75	172	202	119	38	86	203	194	90	134	230	254	255	255	254	239	207	237
90	55	144	212	143	46	83	195	192	91	129	229	254	255	255	254	241	213
208	132	59	117	208	166	62	77	196	178	62	131	235	254	255	255	253	234
162	221	156	61	110	211	177	61	84	185	143	70	146	238	254	255	255	252
159	174	219	175	81	108	199	179	77	74	162	149	61	155	244	255	255	255
223	192	161	214	196	86	89	201	189	63	56	175	95	52	188	249	255	255
172	222	208	179	220	201	95	91	198	178	67	87	149	54	73	204	252	255
206	180	221	227	196	219	205	91	87	192	177	52	82	131	53	94	225	254
202	221	194	221	233	198	218	203	100	91	196	153	60	101	146	90	152	243
103	193	227	205	223	236	214	241	214	106	97	200	172	64	123	188	136	199
152	90	169	230	216	236	247	244	212	202	92	98	207	151	73	176	206	141
232	180	104	152	231	239	227	236	186	210	204	104	125	214	163	90	193	196
157	232	200	110	146	227	220	211	236	218	239	213	109	139	219	147	111	213
62	140	223	212	136	135	214	224	230	245	251	249	204	98	155	213	109	152
154	67	104	199	223	145	134	220	233	228	251	248	247	188	93	179	194	109
203	182	82	76	197	231	165	131	212	243	253	254	255	246	183	113	203	188
86	191	199	107	92	186	229	153	119	213	252	255	255	254	244	176	151	224
75	75	176	211	126	75	165	225	152	116	214	253	255	254	252	242	179	191
203	115	65	150	216	142	71	167	222	146	120	225	254	255	255	254	231	153
186	220	140	67	145	217	148	71	148	222	155	155	236	254	255	255	252	206
50	155	224	163	81	133	213	152	77	184	226	158	168	240	255	255	255	242
72	42	139	214	183	86	131	212	163	107	192	226	159	176	243	255	255	254
202	99	38	91	206	191	93	121	213	177	105	199	222	136	183	247	255	255
184	214	131	41	88	204	200	103	114	203	183	136	214	208	134	206	252	255
92	156	219	151	48	88	202	206	105	101	210	179	142	222	200	148	229	254
177	86	145	217	165	65	85	200	206	94	122	215	170	161	230	196	188	244
201	192	124	132	216	192	73	82	201	202	99	138	221	178	195	238	199	217
126	209	211	116	157	220	165	57	109	213	188	93	175	226	181	231	236	201
193	128	165	207	165	128	197	188	103	94	194	199	130	162	227	243	251	240
234	209	131	162	223	177	137	210	208	95	91	203	207	123	191	248	255	254
144	210	219	155	167	226	198	151	214	209	101	92	207	204	130	205	252	255
149	111	191	229	182	159	221	204	157	214	204	86	104	219	197	150	230	254
224	172	114	184	230	176	148	217	202	148	205	199	99	137	226	198	187	245
147	222	202	121	160	227	181	159	219	192	127	208	202	101	151	231	194	218
121	124	210	208	132	164	231	207	165	218	175	128	210	194	101	187	233	193
221	134	110	200	214	141	160	229	211	189	222	176	145	222	177	112	212	231
250	234	160	121	186	217	143	157	232	221	199	229	198	184	226	163	146	232
255	254	245	192	110	167	216	161	173	233	218	192	233	199	195	227	151	166
244	254	255	249	204	123	162	224	173	174	233	217	212	237	200	207	213	130
149	236	254	255	252	213	128	175	227	180	174	234	224	224	235	190	234	210
30	130	227	253	255	252	218	150	167	223	176	185	238	233	245	243	251	248
31	24	107	211	251	255	253	223	138	176	231	191	183	240	253	255	255	255
154	49	16	78	198	249	255	253	230	169	184	229	175	195	248	255	255	255
207	183	74	17	66	177	244	255	254	230	148	178	223	178	216	252	255	255
95	201	201	95	20	42	158	242	255	253	220	157	214	233	206	238	254	255
69	76	176	209	115	27	37	156	242	255	253	233	194	223	238	224	249	255

7	*7*	1	9	1	*7*	1	9	1	1	1	1	1	9	9	*7*	*7*	1
1	*7*	1	9	1	*7*	1	9	1	1	1	1	*7*	1	1	1	1	1
1	1	1	1	1	*7*	1	9	1	1	1	1	1	1	9	1	1	9
1	1	1	1	1	1	1	1	9	1	1	1	9	1	9	1	1	1
7	*7*	1	1	1	1	1	1	1	9	1	1	9	1	*7*	*7*	1	*7*
9	*7*	*7*	*7*	*7*	*7*	1	9	1	1	9	1	*7*	1	*7*	*7*	1	9
9	1	1	*7*	*7*	*7*	*7*	1	9	1	9	9	1	1	1	*7*	*7*	1
9	9	9	1	*7*	*7*	*7*	*7*	1	9	9	9	9	1	1	1	*7*	*7*
7	1	1	9	1	*7*	*7*	*7*	*7*	1	9	9	9	9	1	1	*7*	*7*
1	1	1	1	9	1	*7*	*7*	*7*	*7*	1	1	9	9	9	1	1	1
9	1	9	1	1	9	1	*7*	*7*	*7*	*7*	1	9	1	9	9	1	1
9	*7*	1	9	1	1	1	1	1	*7*	*7*	1	1	1	1	9	9	1
1	1	*7*	1	9	1	1	1	1	*7*	*7*	*7*	1	1	1	1	9	1
1	1	1	*7*	1	9	1	1	1	1	*7*	*7*	*7*	1	1	1	1	1
1	1	9	1	1	1	1	1	*7*	1	1	*7*	*7*	*7*	1	1	1	1
1	*7*	1	9	1	*7*	1	1	1	1	1	1	*7*	*7*	*7*	1	1	1
1	1	*7*	1	9	1	*7*	1	9	9	1	1	1	*7*	*7*	*7*	1	1
1	1	1	1	1	1	1	*7*	*7*	9	9	1	1	1	*7*	*7*	1	1
7	*7*	1	1	1	1	1	1	*7*	1	9	9	9	1	1	*7*	*7*	1
1	1	*7*	1	1	*7*	1	1	1	1	1	9	9	9	1	*7*	*7*	*7*
1	1	1	*7*	1	1	1	1	1	1	1	1	9	9	9	1	*7*	*7*
7	*7*	1	1	*7*	1	1	1	1	9	1	1	1	10	9	1	1	*7*
1	1	*7*	*7*	*7*	*7*	1	1	1	1	1	1	1	9	10	9	1	1
9	1	1	1	*7*	*7*	*7*	1	1	1	1	1	1	*7*	9	9	1	*7*
1	1	1	1	1	*7*	*7*	*7*	1	1	1	1	1	1	*7*	9	9	1
7	1	1	1	1	1	*7*	*7*	*7*	9	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	*7*	*7*	*7*	1	1	*7*	1	9	1	1	1
1	1	1	1	1	1	1	1	*7*	*7*	*7*	1	1	1	9	9	1	1
1	*7*	1	1	1	1	1	1	1	*7*	*7*	*7*	1	1	1	9	1	1
1	1	1	1	1	1	1	1	1	1	*7*	*7*	*7*	1	1	1	9	9
9	1	9	1	*7*	1	1	1	1	1	1	*7*	*7*	*7*	1	1	1	1
1	*7*	1	9	1	*7*	1	1	1	1	1	1	*7*	*7*	1	1	1	*7*
7	1	*7*	1	1	1	1	1	9	1	1	1	1	*7*	*7*	1	1	1
1	1	1	*7*	1	1	1	1	1	9	1	1	1	1	*7*	*7*	1	1
9	1	9	1	8	1	1	1	1	1	1	1	1	1	1	*7*	*7*	1
9	*7*	1	9	1	*7*	1	9	1	1	1	1	1	1	1	*7*	*7*	*7*
1	1	*7*	1	9	1	*7*	1	9	1	1	1	1	1	1	1	*7*	*7*
1	9	1	*7*	1	1	1	*7*	1	9	1	1	1	1	1	1	1	1
1	1	9	1	1	1	1	1	1	1	9	1	1	1	1	1	1	1
1	*7*	1	9	1	*7*	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	*7*	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	9	1	*7*	1	9	1	1	1	9	1	1	1	1	*7*	1	1
1	*7*	1	1	1	*7*	1	9	1	1	9	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	*7*	1	9	1	1	1	1	*7*	1
1	1	1	1	1	1	1	*7*	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	*7*	1	9	1	1	9	1	1	1	1
1	1	1	1	1	1	1	1	1	*7*	1	9	1	1	1	1	1	1
1	9	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
9	1	9	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	9	1	1	1	1	1	1	1	1	1	1	1	1	1	1
7	1	1	1	9	9	1	1	1	1	1	1	1	1	1	1	1	1
7	*7*	1	1	1	9	9	1	1	1	1	1	1	1	1	1	9	1
7	*7*	*7*	1	1	1	9	9	1	1	1	1	1	1	1	1	9	9
1	*7*	*7*	*7*	*7*	1	1	9	1	1	1	1	1	1	1	1	1	9
9	9	1	*7*	*7*	*7*	1	1	9	1	1	1	1	1	1	*7*	1	1
9	*11*	9	1	*7*	*7*	*7*	1	1	9	1	1	1	1	1	*7*	*7*	1
1	9	*11*	9	1	*7*	*7*	*7*	1	1	1	1	1	1	1	1	1	1
7	1	9	*11*	9	1	*7*	*7*	*7*	1	1	9	9	1	1	1	1	1
1	1	9	1	*11*	9	9	*7*	*7*	*7*	1	1	1	1	1	1	1	1
1	1	1	9	1	9	9	9	*7*	*7*	*7*	1	1	1	1	1	1	1

Fig. 4

```
192 204   0   0 145 215   0   0 154 204   0   0 187   0   0 214 185   0
  0   0 214   0   0   0 209   0   0 158 194   0   0 195   0   0 222   0
223   0   0 214 176   0   0 207 156   0 152 191   0   0 190   0   0 221
  0 220   0   0 217 181   0   0 206 152   0 158 164   0 192   0   0 198
  0   0 226-209   0 216 182   0   0 201   0   0 175 196   0 235   0   0
205 213-212 228-220   0 224 185   0   0 205   0   0 195 242 243 232   0
  0 199 233   0-216 233-207 210 162   0   0 211   0   0 215-240 249 208
  0   0 182 228-202-229-231   0 209   0   0   0 209   0   0 230 254 244
177   0   0 163 231 234-236-235-220 227 188   0   0 212   0   0 242 254
182 192   0   0   0 232-233-246-245-230 240   0   0   0 215   0   0 249
  0 152 210   0   0   0 227 248-252-251 246 228 190   0 187 202   0-215
  0   0 140 212 149   0   0 230 248-240-236   0 224   0   0 208   0   0
184   0   0   0 199 162   0   0 224 244-251-233   0 227   0   0 224   0
197 200   0   0   0 200 186   0   0 225 254-253-236 248 237   0-210 238
  0 172 202   0   0   0 203 194   0   0 230 254-255-255-254-239-207-237
  0   0 144 212 143   0   0 195 192   0   0 229 254-255-255-254-241-213
208   0   0   0 208 166   0   0 196 178   0   0 235 254-255-255-253-234
  0 221 156   0   0 211 177   0   0 185 143   0   0 238-254-255-255-252
  0   0 219 175   0   0 199 179   0   0 162 149   0   0 244-255-255-255
223   0   0 214 196   0   0 201 189   0   0 175   0   0 188 249-255-255
  0 222-208   0 220 201   0   0 198 178   0   0 149   0   0 204 252-255
206   0-221 227   0 219 205   0   0 192 177   0   0 131   0   0 225 254
202 221   0-221 233   0 218 203   0   0 196 153   0   0 146   0   0 243
  0 193 227-205-223-236-214 241 214   0   0 200 172   0   0 188   0   0
  0   0   0 230-216-236-247 244-212 202   0   0 207   0   0 176 206   0
232 180   0   0 231 239-227-236   0-210 204   0   0 214 163   0 193 196
  0 232 200   0   0 227-220-211-236-218 239 213   0   0 219   0   0 213
  0   0 223 212   0   0 214-224-230-245-251 249 204   0   0 213   0   0
154   0   0 199 223   0   0 220 233-228-251-248 247   0   0 179 194   0
203 182   0   0 197 231   0   0 212 243-253-254-255 246   0   0 203 188
  0 191 199   0   0 186 229   0   0 213 252-255-255-254 244   0   0 224
  0   0 176 211   0   0   0 225   0   0 214 253-255-254-252 242   0   0
203   0   0   0 216   0   0 167 222   0   0 225 254-255-255-254-231   0
186 220   0   0   0 217   0   0   0 222   0   0 236-254-255-255 252-206
  0 155 224 163   0   0 213   0   0 184 226   0   0 240-255-255-255 242
  0   0 139 214 183   0   0 212   0   0 192 226   0   0 243-255-255-254
202   0   0   0 206 191   0   0 213 177   0 199 222   0   0 247-255-255
184 214   0   0   0 204 200   0   0 203 183   0 214 208   0-206 252-255
  0   0 219 151   0   0 202 206   0   0 210   0   0 222   0   0-229-254
177   0   0 217 165   0   0 200 206   0   0 215   0   0 230   0   0-244
201 192   0   0 216 192   0   0 201 202   0   0 221   0   0 238   0-217
  0 209 211   0   0 220 165   0   0 213 188   0   0 226   0-231-236-201
193   0   0 207   0   0 197 188   0   0 194 199   0   0 227-243-251-240
234 209   0   0 223   0   0 210 208   0   0 203 207   0   0 248-255-254
  0 210 219   0   0 226   0   0 214 209   0   0 207 204   0-205 252-255
  0   0 191 229   0   0 221 204   0 214 204   0   0 219   0   0-230-254
224   0   0   0 230   0   0 217-202   0 205 199   0   0 226   0   0-245
  0 222 202   0   0 227   0   0 219   0   0 208 202   0   0 231   0-218
  0   0 210 208   0   0 231-207   0 218   0   0 210 194   0   0 233   0
221   0   0 200 214   0   0 229-211   0 222   0   0 222   0   0 212-231
250 234   0   0 186 217   0   0 232-221   0 229   0   0 226   0   0 232
-255-254 245   0   0   0 216   0   0 233-218   0 233   0   0 227   0   0
244 254-255 249-204   0   0 224   0   0 233-217-212 237   0-207 213   0
  0 236 254-255 252-213   0   0 227   0   0 234-224-224-235   0 234-210
  0   0 227 253-255 252-218   0   0 223   0   0 238-233-245-243-251 248
  0   0   0 211 251-255 253-223   0   0 231   0   0 240-253-255-255-255
154   0   0   0 198 249-255 253 230   0   0 229   0   0 248-255-255-255
207 183   0   0   0 177 244 255 254 230   0   0 223   0-216-252-255-255
  0 201 201   0   0   0   0 242 255 253-220   0-214 233-206-238-254-255
  0   0 176 209   0   0   0 156 242 255 253-233   0-223-238-224-249-255
```

Fig. 5

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	228	220	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	216	233	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	202	229	231	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	234	236	235	220	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	233	246	245	230	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	248	252	251	246	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	248	240	236	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	244	251	233	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	254	253	236	248	237	0	210	238
0	0	0	0	0	0	0	0	0	0	0	254	255	255	254	239	207	237
0	0	0	0	0	0	0	0	0	0	0	0	254	255	255	254	241	213
0	0	0	0	0	0	0	0	0	0	0	0	0	254	255	255	253	234
0	0	0	0	0	0	0	0	0	0	0	0	0	0	254	255	255	252
0	0	0	0	0	0	0	0	0	0	0	0	0	0	244	255	255	255
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	249	255	255
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	252	255
0	0	0	227	0	0	0	0	0	0	0	0	0	0	0	0	0	254
0	0	0	221	233	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	205	223	236	214	241	0	0	0	0	0	0	0	0	0	0
0	0	0	0	216	236	247	244	212	0	0	0	0	0	0	0	0	0
0	0	0	0	0	239	227	236	0	210	0	0	0	0	0	0	0	0
0	0	0	0	0	0	220	211	236	218	239	0	0	0	0	0	0	0
0	0	0	0	0	0	0	224	230	245	251	249	0	0	0	0	0	0
0	0	0	0	0	0	0	0	233	228	251	248	247	0	0	0	0	0
0	0	0	0	0	0	0	0	0	243	253	254	255	246	0	0	0	0
0	0	0	0	0	0	0	0	0	0	252	255	255	254	244	0	0	0
0	0	0	0	0	0	0	0	0	0	0	253	255	254	252	242	0	0
0	0	0	0	0	0	0	0	0	0	0	0	254	255	255	254	231	0
0	0	0	0	0	0	0	0	0	0	0	0	0	254	255	255	252	206
0	0	0	0	0	0	0	0	0	0	0	0	0	240	255	255	255	242
0	0	0	0	0	0	0	0	0	0	0	0	0	0	243	255	255	254
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	247	255	255
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	206	252	255
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	229	254
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	244
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	217
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	236	201
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	243	251	240
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	248	255	254
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	205	252	255
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	230	254
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	245
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	218
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
250	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
255	254	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
244	254	255	249	204	0	0	0	0	0	0	217	212	237	0	0	0	0
0	0	0	255	252	213	0	0	0	0	0	234	224	224	235	0	234	210
0	0	0	0	255	252	218	0	0	0	0	0	238	233	245	243	251	248
0	0	0	0	0	255	253	0	0	0	0	0	0	240	253	255	255	255
0	0	0	0	0	0	0	0	0	0	0	0	0	0	248	255	255	255
0	0	0	0	0	0	0	0	0	0	0	0	0	0	216	252	255	255
0	0	0	0	0	0	0	0	0	0	0	0	0	233	206	238	254	255
0	0	0	0	0	0	0	0	0	0	253	233	0	223	238	224	249	255

Fig. 6

```
-192-204 0 0 145-215 0 0-154-204 0 0-187 0 0-214 185 0
0 0-214 0 0 0-209 0 0 158-194 0 0-195 0 0-222 0
-223 0 0-214 176 0 0-207 156 0 152-191 0 0-190 0 0-221
0-220 0 0-217 181 0 0-206 152 0 158-164 0-192 0 0 198
0 0-226 209 0-216 182 0 0-201 0 0 175-196 0-235 0 0
205-213 212 0 0 0-224 185 0 0-205 0 0 195-242-243 232 0
0 199-233 0 0 0 207-210 162 0 0-211 0 0 215 240-249 208
0 0 182-228 0 0 0 0-209 0 0 0-209 0 0 230-254 244
177 0 0 163-231 0 0 0 0-227 188 0 0-212 0 0 242-254
-182-192 0 0 0-232 0 0 0 0-240 0 0 0-215 0 0 249
0 152-210 0 0 0-227 0 0 0 0-228 190 0 187-202 0 215
0 0 140-212 149 0 0-230 0 0 0 0-224 0 0-208 0 0
184 0 0 0-199 162 0 0-224 0 0 0 0-227 0 0-224 0
-197-200 0 0 0-200 186 0 0-225 0 0 0 0 0 0 0
0 172-202 0 0 0-203 194 0 0-230 0 0 0 0 0 0
0 0 144-212 143 0 0-195 192 0 0-229 0 0 0 0 0
-208 0 0 0-208 166 0 0-196 178 0 0-235 0 0 0 0
0-221 156 0 0-211 177 0 0-185 143 0 0-238 0 0 0 0
0 0-219 175 0 0-199 179 0 0-162 149 0 0 0 0 0
-223 0 0-214 196 0 0-201 189 0 0-175 0 0-188 0 0 0
0-222 208 0-220 201 0 0-198 178 0 0-149 0 0-204 0 0
206 0-221 0 0-219 205 0 0-192 177 0 0-131 0 0-225 0
202-221 0 0 0 0-218-203 0 0-196 153 0 0-146 0 0-243
0 193-227 0 0 0 0 0-214 0 0-200 172 0 0-188 0 0
0 0 0-230 0 0 0 0 0-202 0 0-207 0 0 176-206 0
-232 180 0 0-231 0 0 0 0 0-204 0 0-214 163 0-193-196
0-232 200 0 0-227 0 0 0 0 0-213 0 0-219 0 0-213
0 0-223-212 0 0-214 0 0 0 0 0-204 0 0-213 0 0
154 0 0 199-223 0 0-220 0 0 0 0 0 0 0 179-194 0
-203 182 0 0 197-231 0 0-212 0 0 0 0 0 0 0-203-188
0-191-199 0 0 186-229 0 0-213 0 0 0 0 0 0 0-224
0 0 176-211 0 0 0-225 0 0-214 0 0 0 0 0 0 0
-203 0 0 0-216 0 0 167-222 0 0-225 0 0 0 0 0 0
186-220 0 0 0-217 0 0 0-222 0 0-236 0 0 0 0 0
0 155-224 163 0 0-213 0 0 184-226 0 0 0 0 0 0
0 0 139-214 183 0 0-212 0 0 192-226 0 0 0 0 0
-202 0 0 0-206 191 0 0-213 177 0 199-222 0 0 0 0
184-214 0 0 0-204 200 0 0-203 183 0-214 208 0 0 0
0 0-219 151 0 0-202-206 0 0-210 0 0-222 0 0 0
177 0 0-217 165 0 0 200-206 0 0-215 0 0-230 0 0
-201 192 0 0-216 192 0 0 201-202 0 0-221 0 0-238 0 0
0-209-211 0 0-220 165 0 0-213 188 0 0-226 0-231 0 0
193 0 0-207 0 0-197 188 0 0-194 199 0 0-227 0 0 0
-234 209 0 0-223 0 0-210 208 0 0-203 207 0 0 0 0
0-210-219 0 0-226 0 0-214 209 0 0-207 204 0 0 0
0 0 191-229 0 0-221 204 0-214 204 0 0-219 0 0 0
-224 0 0 0-230 0 0-217 202 0-205 199 0 0-226 0 0
0-222 202 0 0-227 0 0-219 0 0-208 202 0 0-231 0 0
0 0-210-208 0 0-231 207 0-218 0 0-210 194 0 0-233 0
-221 0 0 200-214 0 0-229 211 0-222 0 0-222 0 0 212-231
0-234 0 0 186-217 0 0-232 221 0-229 0 0-226 0 0 232
0 0-245 0 0 0-216 0 0-233 218 0-233 0 0-227 0 0
0 0 0 0 0 0 0-224 0 0-233 0 0 0 0 207-213 0
0-236-254 0 0 0 0 0-227 0 0 0 0 0 0 0 0
0 0 227-253 0 0 0 0 0-223 0 0 0 0 0 0 0
0 0 0 211-251 0 0 223 0 0-231 0 0 0 0 0 0
154 0 0 0 198-249-255-253 230 0 0-229 0 0 0 0 0
-207 183 0 0 0 177 244-255-254 230 0 0-223 0 0 0 0
0-201-201 0 0 0 0 242-255 253 220 0-214 0 0 0 0
0 0 176-209 0 0 0 156 242-255 0 0 0 0 0 0 0
```

Fig. 7

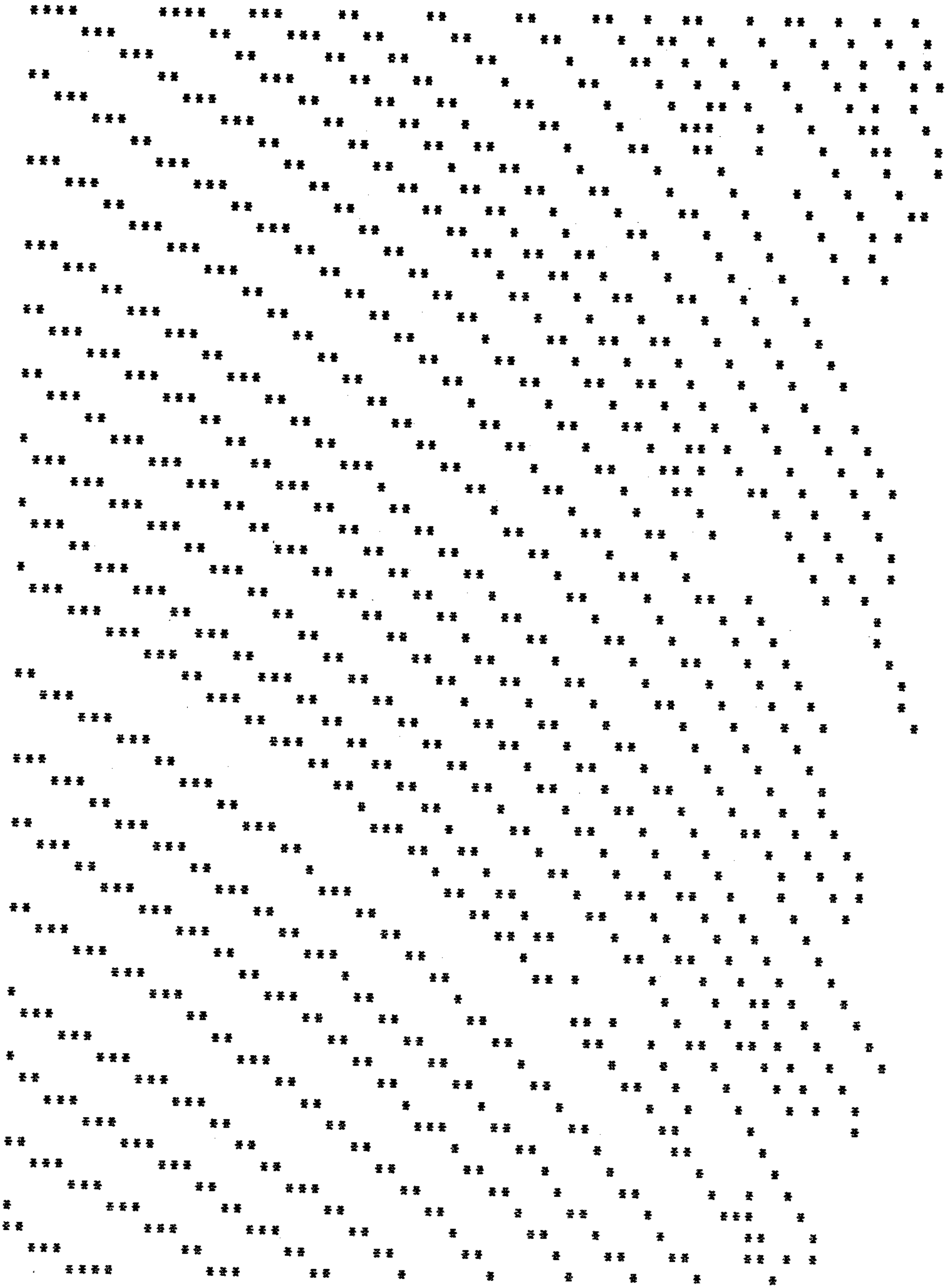


Fig. 8