The BITG Message Codes

Hanjie Li[1] and Jonathan H. Jiang[2]

[1.] Department of Aerospace and Ocean Engineering,
Virginia Polytechnic Institute and State University, Blacksburg, VA 24061, USA

[2.] Jet Propulsion Laboratory,
California Institute of Technology, Pasadena, CA 91108, USA

This document contains the software codes and execute instructions for generating the proposed Beacon in the Galaxy (BITG) Message. Future update and modification may be needed depending on the actual plan and implementation needs.

1. Instruction for executing the code:

The codes shown in this document are developed using MATLAB. The codes should be placed in the same folder to the matrices and images that need to be imported.

The codes in Section 2 are combined. Trouble might occur if it is being executed as a single piece. Future update will be needed depending on the actual plan and implementation needs for sending the BITG message. For now, the best way to use them is to separate them into several code files and execute them independently according to their own purposes. In section 2, codes from different files are separated by three blank lines.

In the first two lines of each piece of code, “clear” empties the Workspace that stores all the variables, while “clc” cleans up the Command Window that shows the commands being run and unsuppressed operations. When Code B need to use a variable in Code A, the code just been run, the “clear” command in Code B needs to be commented out by adding a “%” in front of it.

The comments within the codes define the properties of the code and describes the function of it. “Dependent” means that the code imports values from Excels and other external sources, or from another code. “Automated” means that the user does not need to change the contents of the code to make it work properly. Oppositely a “not automated” code requires the user to change certain contents of it to work as the user desires. Instructions to change the contents are included in the comments at the end of the lines. Even in the “automated” codes, the user is still able to block some unnecessary functions of the code by commenting the corresponding lines out, as the instructions suggest.

The codes are the “general methods” used to create matrices and the actual BITG Message, thus may not be exclusively serving such a purpose. They may seek updates to reduce their length and to improve the interactions with the user so that a less professional user can easily make them work properly or use them for similar purposes.

2. The BITG MATLAB Codes

```
% ========================================================
% Latest Update: 04/23/2022; Hanjie Li, Jonathan H. Jiang
% Contacts: esdm.lijas@gmail.com;Jonathan.H.Jiang@jpl.nasa.gov
% --------------------------------------------------------

% World_Map
% ---------
clear
clc
%This is an independent code. This is an automated code.
%This code constructs the map of the world within itself
%thus does not need to import matrices from the folder.

%World Map Matrix
%-----------------
Map=[zeros(1,24) ones(1,4) zeros(1,40);
    zeros(1,16) ones(1,4) 0 ones(1,9) zeros(1,38);
    zeros(1,14) ones(1,5) 0 ones(1,9) zeros(1,5) ones(1,4)
zeros(1,3) ones(1,3) zeros(1,5) 1 1 zeros(1,17);
    zeros(1,10) ones(1,3) 0 0 1 1 0 0 ones(1,10) zeros(1,5)
ones(1,4) zeros(1,12) 1 1 zeros(1,16);
    zeros(1,9) 1 1 0 0 ones(1,4) 0 0 ones(1,10) zeros(1,13)
ones(1,3) zeros(1,3) ones(1,6) zeros(1,3) ones(1,3)
zeros(1,8);
    zeros(1,8) ones(1,4) 0 1 0 ones(1,3) zeros(1,3) ones(1,8)
zeros(1,13) 1 0 0 ones(1,11) 0 0 1 zeros(1,9);
    0 ones(1,18) 0 0 ones(1,7) zeros(1,7) ones(1,3) zeros(1,5)
ones(1,19) 0 ones(1,3) 0 0;
    ones(1,17) 0 1 1 0 0 ones(1,5) zeros(1,7) ones(1,6) 0
ones(1,26) 0;
    0 ones(1,19) 0 0 ones(1,3) 0 0 1 1 zeros(1,5) ones(1,34);
    ones(1,15) 0 0 ones(1,3) 0 0 1 1 zeros(1,9) 1 1 0
ones(1,29) 0 0 1;
    0 ones(1,3) 0 0 ones(1,9) 0 0 ones(1,3) zeros(1,3) 1
zeros(1,7) 1 0 1 1 0 ones(1,24) 0 1 1 zeros(1,5);
    0 1 zeros(1,4) ones(1,15) zeros(1,9) 1 1 0 0 ones(1,25) 0
0 1 1 zeros(1,5);
    1 zeros(1,6) ones(1,15) zeros(1,8) ones(1,29) 0 0 1
zeros(1,6);
    zeros(1,8) ones(1,14) zeros(1,9) ones(1,28) 0 0 1
zeros(1,6);
    zeros(1,8) ones(1,13) zeros(1,10) ones(1,6) 0 0 1 1 0
ones(1,18) zeros(1,8);
    zeros(1,8) ones(1,11) zeros(1,11) ones(1,3) 0 ones(1,7) 0
ones(1,14) 0 0 1 zeros(1,9);
    zeros(1,9) ones(1,9) zeros(1,12) ones(1,4) zeros(1,4)
ones(1,21) zeros(1,9);
    zeros(1,10) ones(1,7) zeros(1,13) ones(1,25) 0 1
zeros(1,11);
    zeros(1,10) ones(1,4) 0 0 1 zeros(1,12) ones(1,26)
zeros(1,13);
    0 1 1 zeros(1,9) 1 1 0 1 1 zeros(1,11) ones(1,15) 0 0
ones(1,10) zeros(1,13);
    zeros(1,12) ones(1,4) 0 1 1 zeros(1,9) ones(1,15) 0 0
ones(1,3) 0 ones(1,3) 0 0 1 zeros(1,13);
```

```
    zeros(1,15) 1 1 zeros(1,11) ones(1,13) zeros(1,5) 1
zeros(1,3) ones(1,3) 0 1 1 zeros(1,12);
    zeros(1,15) ones(1,7) zeros(1,7) ones(1,13) zeros(1,4) 1
zeros(1,4) 1 0 0 1 1 zeros(1,12);
    zeros(1,16) ones(1,7) zeros(1,7) ones(1,11) zeros(1,6) 1 0
0 1 0 0 1 0 1 zeros(1,12);
    zeros(1,16) ones(1,8) zeros(1,9) ones(1,7) zeros(1,10)
ones(1,8) zeros(1,10);
    zeros(1,16) ones(1,10) zeros(1,8) ones(1,6) zeros(1,11) 1
1 0 1 0 0 ones(1,5) zeros(1,6);
    zeros(1,17) ones(1,9) zeros(1,8) ones(1,6) zeros(1,13) 1 0
0 1 0 1 1 0 0 1 zeros(1,5);
    zeros(1,17) ones(1,8) zeros(1,9) ones(1,7) zeros(1,14)
ones(1,4) zeros(1,9);
    zeros(1,18) ones(1,7) zeros(1,9) ones(1,5) 0 1 zeros(1,13)
ones(1,6) zeros(1,8);
    zeros(1,18) ones(1,6) zeros(1,10) ones(1,5) 0 1
zeros(1,12) ones(1,8) zeros(1,7);
    zeros(1,18) ones(1,5) zeros(1,12) ones(1,3) zeros(1,15)
ones(1,8) zeros(1,7);
    zeros(1,18) ones(1,4) zeros(1,13) 1 1 zeros(1,16)
ones(1,8) zeros(1,7);
    zeros(1,18) ones(1,3) zeros(1,37) ones(1,3) zeros(1,3) 1 1
0 0;
    zeros(1,18) 1 1 zeros(1,44) 1 zeros(1,3);
    zeros(1,18) 1 1 zeros(1,39) 1 zeros(1,3) 1 zeros(1,4);
    zeros(1,18) 1 1 zeros(1,48);
    zeros(1,18) ones(1,3) zeros(1,47);
    zeros(1,19) 1 zeros(1,48)];

%Graphing
%--------
figure(3)
%Comment out either line of the two below: with or without RLI
%image([zeros(1,68);ones(1,68);zeros(1,68);ones(1,68);
%zeros(1,68);Map]*200)
image(Map*200)
%Enlarge to distinguish the colour scale axis equal
%Make sure that the x- and y-axis are in the same scale
%colorbar %Comment out if do not want to see the colour scale
%bar

% Image_Reading
% -------------
%clear
clc

%This code is a dependent code.
%This code is not an automated code.
%This code analyses the images that need to be reproduced
%in the BITG
%Message with the resolution customized. This code does
%not carry export mechanism.

%% Part 1, message image reproduction
%-----------------------------------
res_1=127; %resolution setting
```

```
%"DDM_PXX" is the name of the targeted image
TarImg=imresize(DDM_P11,[res_1 res_1]);
%resizing the image, no change to colour distribution
BW=imbinarize(TarImg); %from unit8 to double
BW=1-BW(:,:,1);
%use red light portion, flip black and white. this is the 0
%and 1 matrix to send figure(4)
image([zeros(1,res_1);ones(1,res_1);zeros(1,res_1);ones(1,res_
1);zeros(1,res_1);BW]*200)
%form desired image axis equal
%Make sure that the x- and y-axis are in the same scale

%% Part 2, world map reproduction
% --------------------------------
TarImg2=imresize(digital_world,[38 68]);
%resizing the image, no change to colour distribution
BW2=imbinarize(TarImg2); %from unit8 to double
BW2=1-BW2(:,:,1);
%use red light portion, flip black and white. this is the 0
%and 1 matrix to send figure(5)
image(BW2*200) %form desired image
%Make sure that the x- and y-axis are in the same scale
%axis equal

% Matrix_View
% -----------
%clear
clc

%This is a dependent code. This is not an automated code.
%This code is used to view the contents stored in a specific
%matrix. In extra, the code provides a choice to export the
%image to high-resolution PDF.

%Visualisation
%-------------
%Manually change the file name
DISP=readmatrix('DDM_P1_a.xlsx');
target=figure(9);
image(DISP*200); %Enlarge to distinguish the colour scale
%Make sure that the x- and y-axis are in the same scale
%axis equal axis off
%Comment out if want to see the axis

%Export
%------
exportgraphics(target,'DDM_P1.pdf')
%Comment out if do not want to export to PDF

% Canvas_Editing
% --------------
%clear
clc

%This code is a dependent code. It is not an automated code.
%This code provides method to quickly adjust the matrices at
```

```
%various locations. The target file need to be manually typed
%in before running.

%Image processing unit
%---------------------
filename='PCR_5.xlsx';
BW=readmatrix(filename);
h=image(BW*200);
%grid on
axis equal
for k=1:10 %number of pixels to adjust each time
    p=ginput(1); %coordinates of mouse click
    BW(round(p(1,2)),round(p(1,1)))=1; %=0: set to zero, =1:
set to one
    set(h,'CData',BW*200);
    %view changes on image draw now;
end

xlswrite(filename,BW);

%Reference:
%"Challenge: How to interact with MATLAB figure using
%mouse events?", Massimo Zanetti on 8 Oct 2016
%https://www.mathworks.com/matlabcentral/answers/306359-
%challenge-how-to-interact-with-matlab-figure-using-mouse-
%events

% Block_Editing
% -------------
%clear
clc

%This code is a dependent code. It is not an automated code.
%This code provides methods to replace a large rectangular
%area of the BITG Message with a correspondingly sized
%content from another existing matrix.
%The contents for replacements need to be manually
%imported/loaded before running.

%% Modular modification
% ----------------------
Load=input('Load existing matrices? "1" for Yes and others for
No');

if Load==1
    filename=input('Type in file name with quotation marks');
    BW=readmatrix(filename);
else
    %Creating blank 127*127 page with frame

BW=[ones(1,127);ones(125,1),zeros(125,125),ones(125,1);ones(1,
127)];
end

%% Loading existing matrices
% -------------------------
%DDM_P8=readmatrix('DDM_P8.xlsx');
```

```
PCR_5=readmatrix('PCR_5.xlsx');

%% Assigning/replacing elements
% ----------------------------
BW(8:119,:)=PCR_5(15:126,:);
%BW(92:98,103:125)=BW(83:89,103:125);
%BW(83:125,42:92)=BW(83:125,75:125);
%BW(83:125,103:125)=DDM_P6(79:121,59:81);
%BW(99:120,5:35)=DDM_P6(17:38,6:36);
%BW(81:87,16:51)=BW(81:87,10:45);

%% Visualising
% ------------
figure(10)
image(BW*200)
axis equal
%Make sure that the x- and y-axis are in the same scale

Save=input('Save matrix to local storage? "1" for Yes and
others for No');
if Save==1
    filename=input('Type in file name with quotation marks');
    %Saving
    xlswrite(filename,BW);
end

% Location_Stamp_Main
%clear
clc

%This code is a dependent code.
%This is an automated code.
%This code loads the location stamp provided in Excel,
%processes it to the desired format for the display, and
%export it to an Excel file storing the matrix.

%Location Stamp
%---------------
L=18; %Coordinate Length
TL=85; %Row Length
Lin=ones(1,TL); %Solid-line row
Sta=[1,1,0,0]; %Start of row
Sep=[0,0,1,1,0,0]; %Separator
End=[0,0,1,1]; %End of row
GEn=[Sep,zeros(1,5),End]; %"Grand" end of row
Sun=[ones(1,5);
    0,1,0,1,0;
    1,0,0,0,0;
    1,1,0,0,0;
    1,1,1,0,1;
    0,0,0,0,1;
    0,0,1,1,1]; %Symbol for the sun

format long
LS=int64(readmatrix('Location_Stamp.xlsx'));
%Load original matrix
```

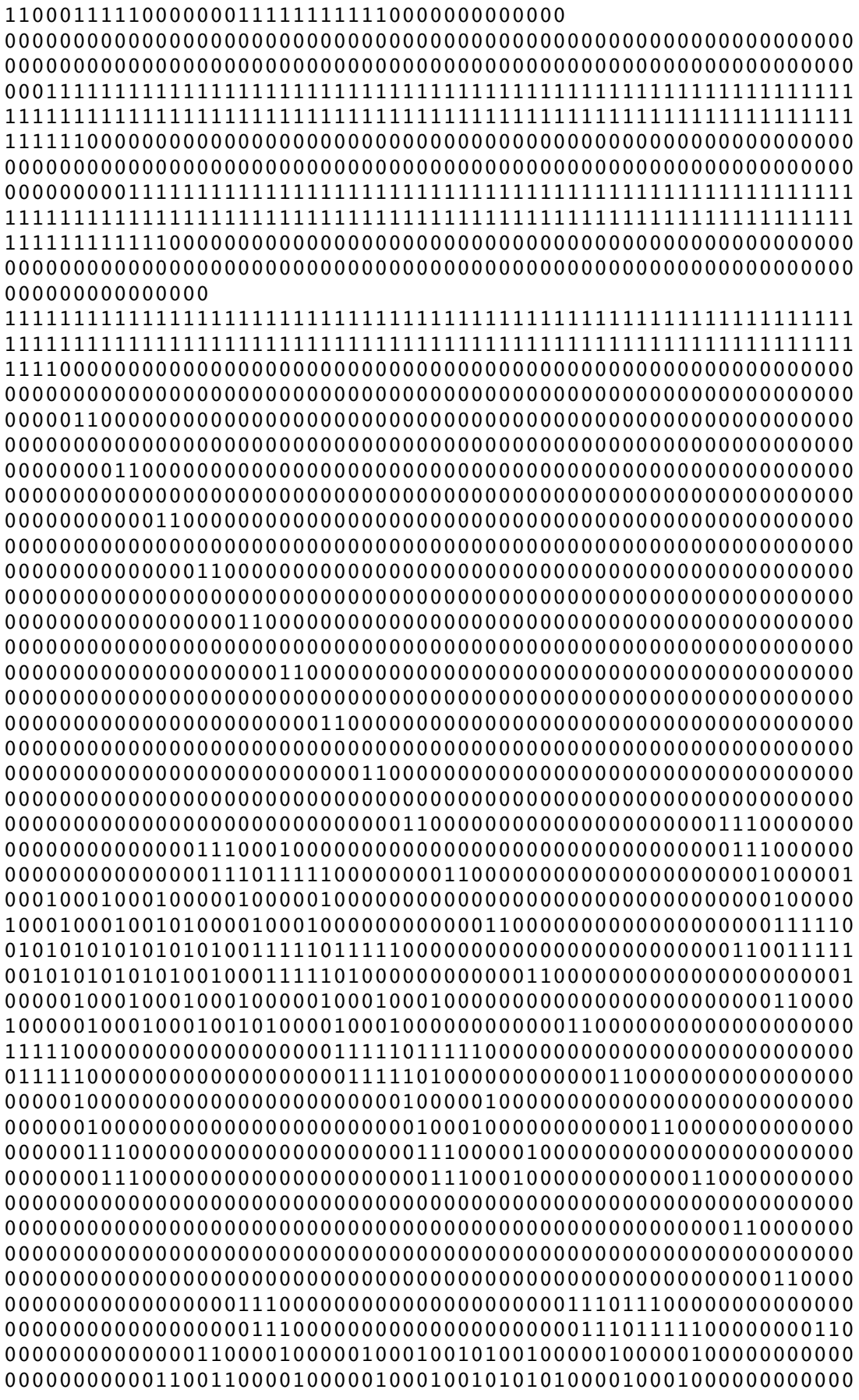
```
%Load coordinates
%----------------
%Load the 1st column of the Excel file, the x-coordinates
x=LS(:,1);

l=length(x); %Get the number of x-coordinates
%Create a matrix with each element as a space for a digit
%of the corrdinatesformat long
u=num2str(int64(10^17*ones(size(x))));

for i=1:l %Sweep row by row
    content=num2str(x(i)); %Load the coordinate
    ol=length(content); %Determine the number of digits shown
    %Calculate the difference from the desired number of
    %digits being displayed
    dl=18-ol;
    while dl~=0
    %Loop around to insert missing '0's in the front omitted
    %by Excel
        content=join(['0',content]);
        ol=length(content);
        dl=18-ol;
    end
    u(i,:)=content; %Store digits in the matrix
end

%Load the 3rd column of the Excel file, the y-coordinates
y=LS(:,3);
l=length(y); %Get the number of y-coordinates
%Create a matrix with each element as a space for a digit of
%the corrdinates format long

v=num2str(int64(10^17*ones(size(y))));

for i=1:l %Sweep row by row
    content=num2str(y(i)); %Load the coordinate
    ol=length(content); %Determine the number of digits shown
    %Calculate the difference from the desired number of
    %digits being displayed
    dl=18-ol;
    %Loop around to insert missing '0's in the front omitted
    %by Excel
    while dl~=0
        content=join(['0',content]);
        ol=length(content);
        dl=18-ol;
    end
    v(i,:)=content; %Store digits in the matrix
end

%Load the 5th column of the Excel file, the z-coordinates
z=LS(:,5);
l=length(z); %Get the number of z-coordinates
%Create a matrix with each element as a space for a digit of
%the corrdinates format long
w=num2str(int64(10^17*ones(size(z))));
```

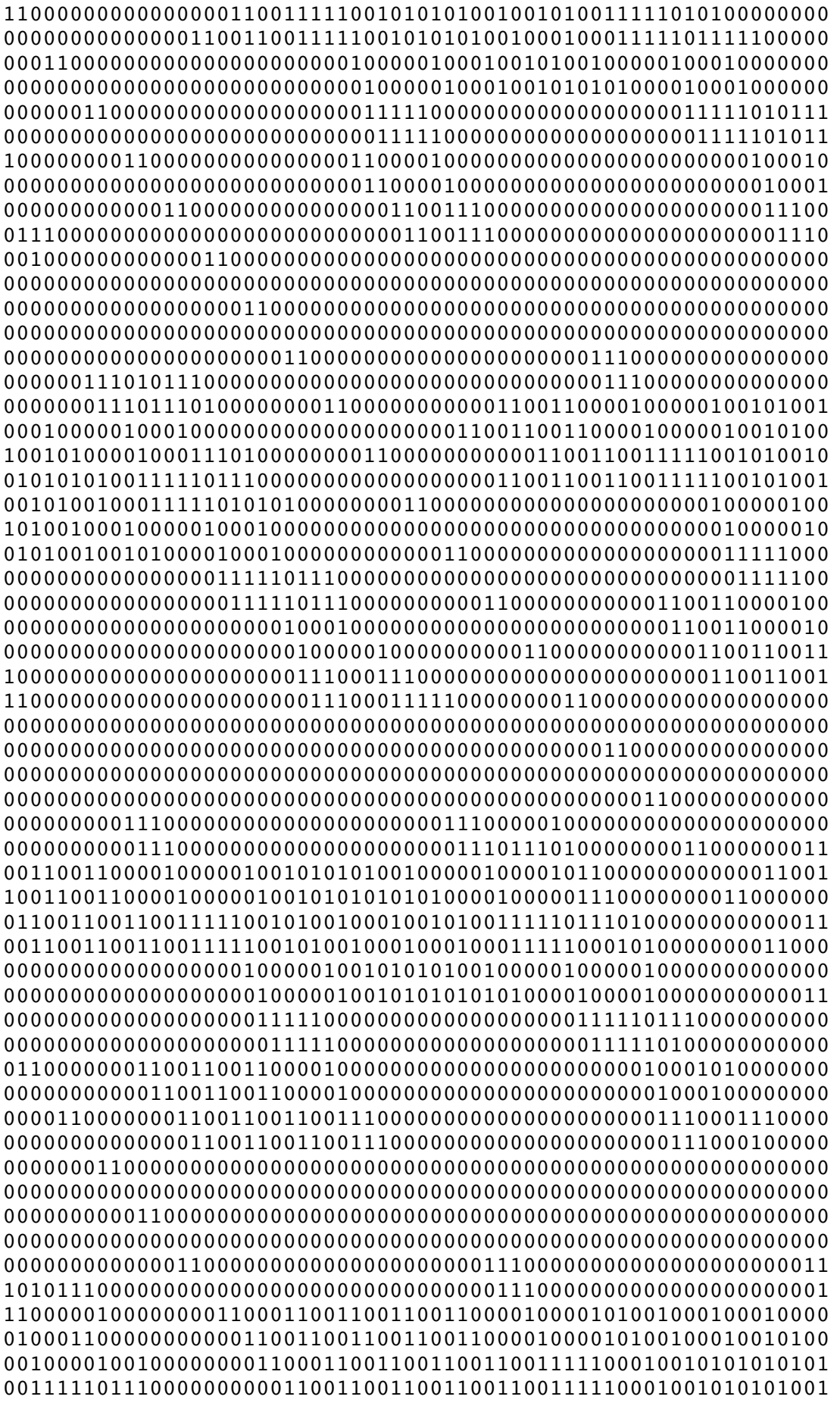
```
for i=1:l %Sweep row by row
    content=num2str(z(i)); %Load the coordinate
    ol=length(content); %Determine the number of digits shown
    %Calculate the difference from the desired number of
    %digits being displayed
    dl=18-ol;
    while dl~=0 %Loop around to insert missing '0's in the
front omitted by Excel
        content=join(['0',content]);
        ol=length(content);
        dl=18-ol;
    end
    w(i,:)=content; %Store digits in the matrix
end

%The first few lines of the location stamp
%----------------------------------------
GCC=[Lin;
Sta,ones(1,L),Sep,ones(1,L),Sep,ones(1,L),Sep,ones(1,5),End;
%Format indicator
    Lin;
    Lin;

Sta,spr('000011011010010010'),Sep,spr('000111111011101001'),Se
p,spr('000111010011110000'),Sep,Sun(1,:),End; %1

Sta,zeros(1,L),Sep,zeros(1,L),Sep,zeros(1,L),Sep,Sun(2,:),End;
Sta,zeros(1,L),Sep,zeros(1,L),Sep,zeros(1,L),Sep,Sun(3,:),End;
Sta,zeros(1,L),Sep,zeros(1,L),Sep,zeros(1,L),Sep,Sun(4,:),End;
Sta,zeros(1,L),Sep,zeros(1,L),Sep,zeros(1,L),Sep,Sun(5,:),End;
Sta,zeros(1,L),Sep,zeros(1,L),Sep,zeros(1,L),Sep,Sun(6,:),End;
Sta,zeros(1,L),Sep,zeros(1,L),Sep,zeros(1,L),Sep,Sun(7,:),End;
    Lin]; %Sun coordinate

%Weaving matrix of the location stamp
%-----------------------------------
for j=2:l %Append the 3D coordinates row by row
    GCC=[GCC;
        Sta,spr(u(j,:)),Sep,spr(v(j,:)),Sep,spr(w(j,:)),GEn;
        Lin];
end

%Store display matrix
xlswrite('Location_Stamp_Display.xlsx',GCC)

%Enlarge to distinguish the colour scale axis equal
%Make sure that the x- and y-axis are in the same scale
%axis off Comment out if want to see the axis
image(GCC*200)

function array=spr(binary) %String spreading function
sL=length(binary);
array=zeros(1,sL);
for i=1:sL
    array(i)=str2num(binary(i));
end
end
```

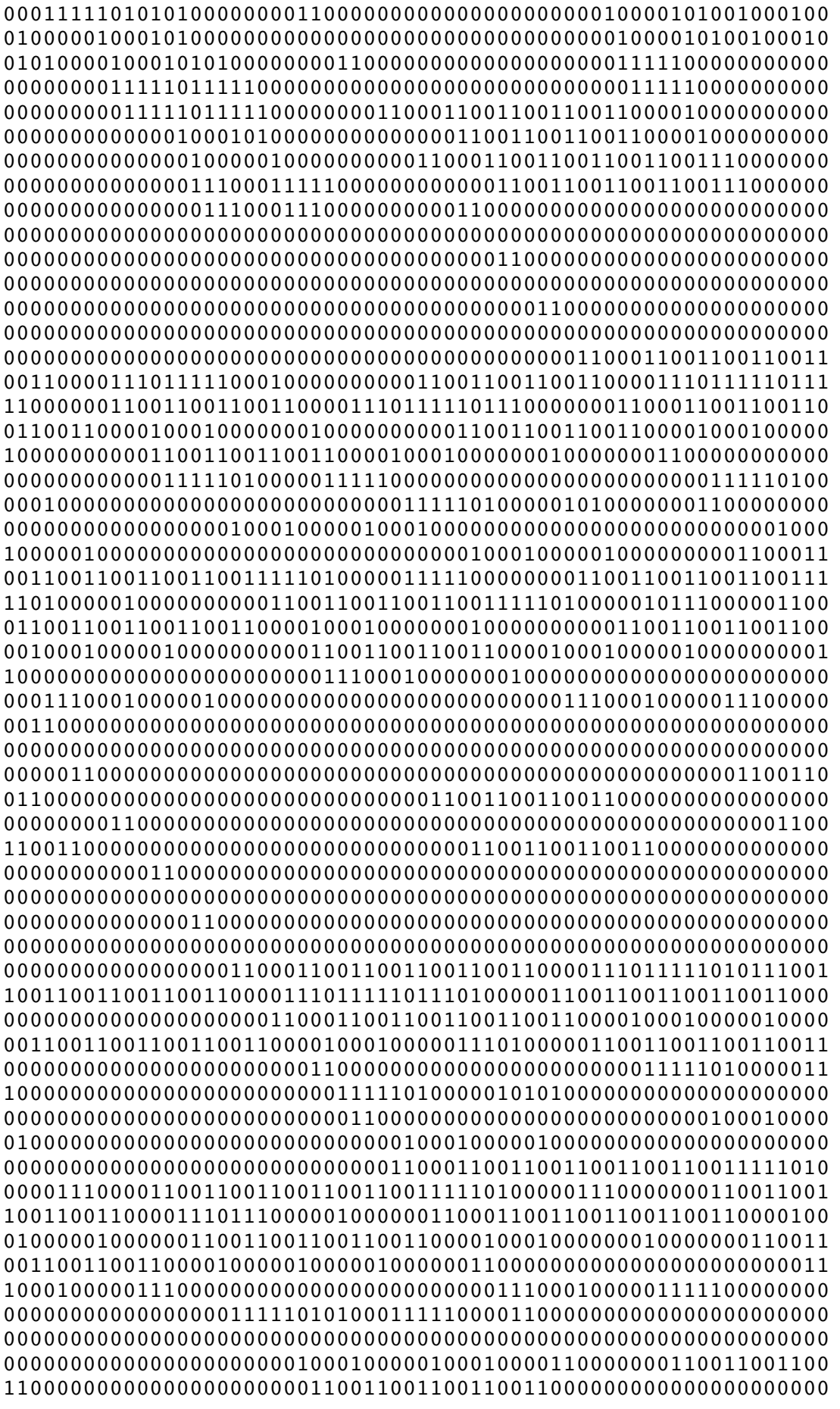
```
% Matrix_Sythesizing
% ------------------
clear
clc

%This is a dependent code. This is an automated code.
%This code synthesizes all the matrices in the folder to
%create a complete message.

%% Message Content Loading
%-------------------------
%Header
%------
Header_2b=[1 1 zeros(1,3) ones(1,5) zeros(1,7) ones(1,11)
zeros(1,13)];
%Row Length Indicator (RLI) 1
RLI_1=[zeros(1,127);ones(1,127);zeros(1,127);ones(1,127);zeros
(1,127)];
%Page 1
P_1=readmatrix('DDM_P1_a.xlsx');
%Page 2
P_2=readmatrix('DDM_P2_a.xlsx');
%Page 3
P_3=readmatrix('DDM_P3_a.xlsx');
%Page 4
P_4=readmatrix('DDM_P4_a.xlsx');
%Page 5
P_5=readmatrix('PCR_5.xlsx');
%Page 6
P_6=readmatrix('PCR_1.xlsx');
%Page 7
P_7=readmatrix('DDM_P17_a.xlsx');
%Page 8
P_8=readmatrix('PCR_2.xlsx');
%Page 9
P_9=readmatrix('PCR_3.xlsx');
%Row Length Indicator (RLI) 2
RLI_2=[zeros(1,72);ones(1,72);zeros(1,72);ones(1,72);zeros(1,7
2)];
%Page 10
P_10=readmatrix('World_Map.xlsx');
%Row Length Indicator (RLI) 3
RLI_3=RLI_1;
%Page_11
P_11=readmatrix('DDM_P14_a.xlsx');
%Page_12
P_12=readmatrix('PCR_4.xlsx');
%Row Length Indicator (RLI) 4
RLI_4=[zeros(1,85);ones(1,85);zeros(1,85);ones(1,85);zeros(1,8
5)];
%Page_13
P_13=readmatrix('Location_Stamp_Display.xlsx');
%Ending
Ending=Header_2b;
```

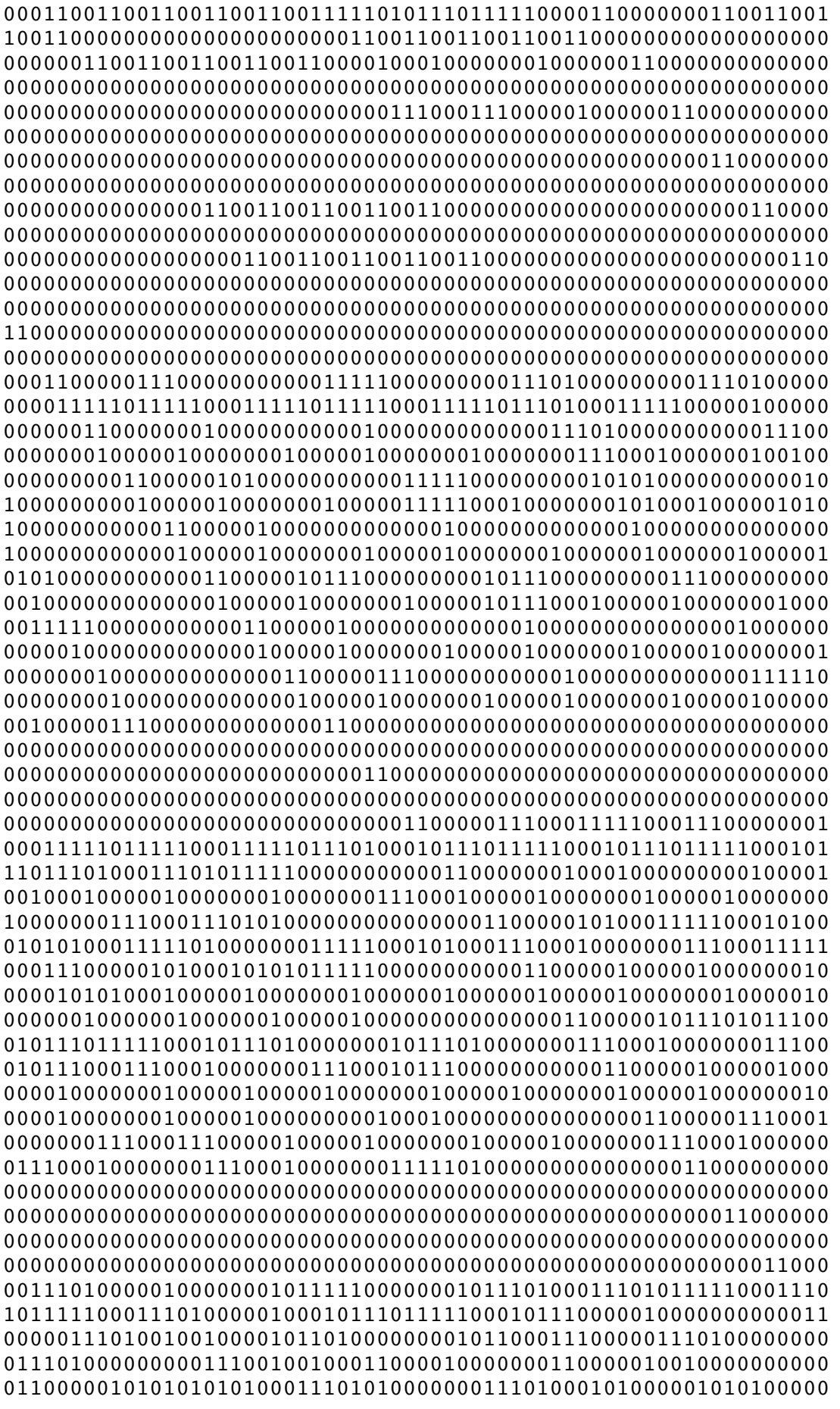
```
%% Automated code generating the final matrix
% ----------------------------------------
Updated_Arecibo_Display=[Header_2b, 0.75*ones(1,127-41);
    RLI_1;
    P_1;
    P_2;
    P_3;
    P_4;
    P_5;
    P_6;
    P_7;
    P_8;
    P_9;
    RLI_2, 0.75*ones(5,127-72);
    P_10, 0.75*ones(42,127-72);
    RLI_3;
    P_11;
    P_12;
    RLI_4, 0.75*ones(5,127-85);
    P_13, 0.75*ones(252,127-85);
    Ending, 0.75*ones(1,127-41)];
image(Updated_Arecibo_Display*200)
%Enlarge to distinguish the colour scale axis equal
%Make sure that the x- and y-axis are in the same scale
%axis off %Comment out if want to see the axis

%% Automated code generating the long string
% ----------------------------------------
Contents={Header_2b,RLI_1,P_1,P_2,P_3,P_4,P_5,P_6,P_7,P_8,P_9,
RLI_2,P_10,RLI_3,P_11,P_12,RLI_4,P_13,Ending};
String=''; %Initialise an empty string
for i=1:length(Contents) %Sweep through matrices
    current_matrix=Contents{i};
    %Obtain the dimension of the urrent matrix
    [a,b]=size(current_matrix);
    for r=1:a
        for c=1:b %Sweep through each element
           %Append elements
           String=join([String,num2str(current_matrix(r,c))]);
        end
    end
    %Add a space at the end of each section
    String=join([String,' ']);
end
%================================================
```

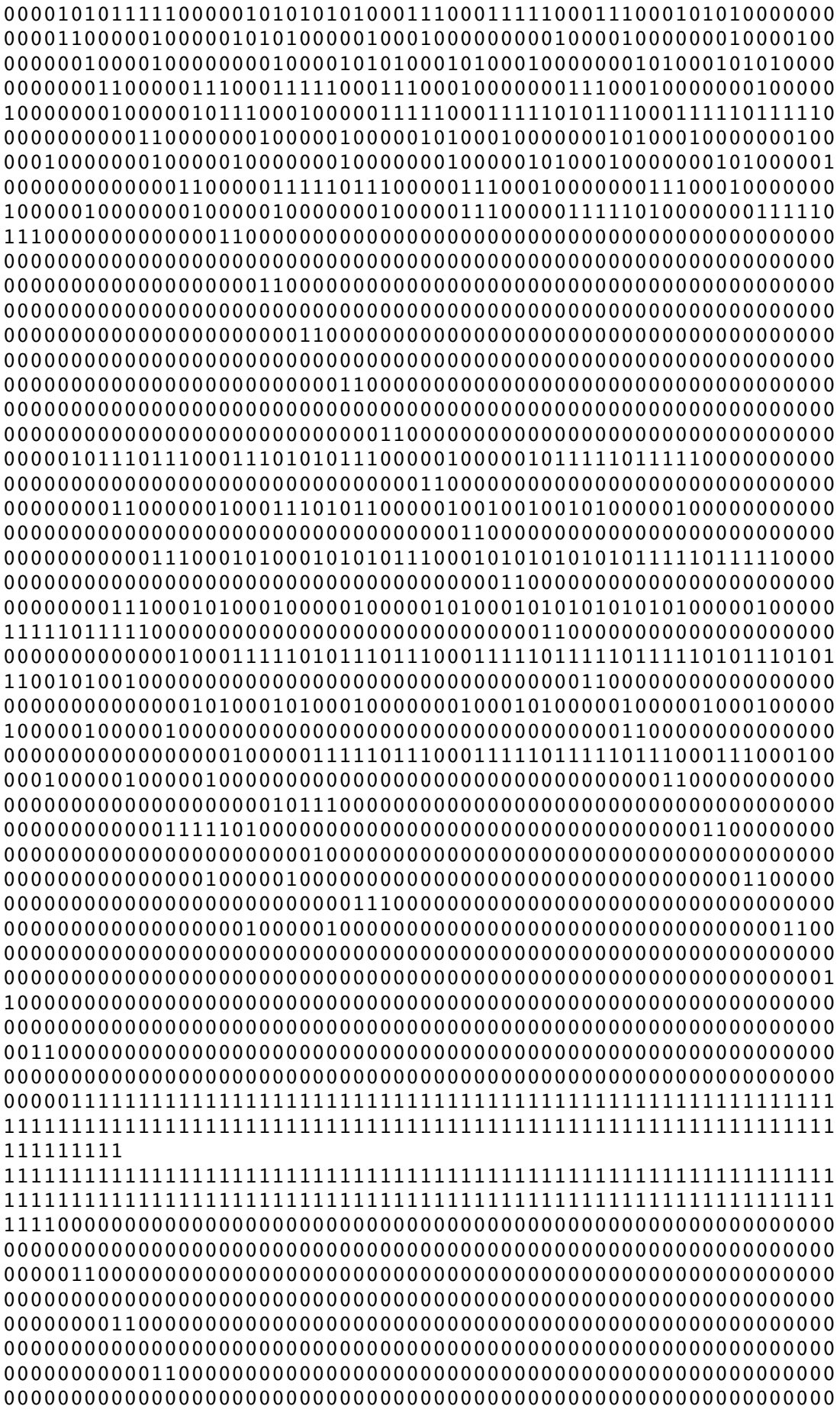
3. A Printout of the BITG Message in Binary Format

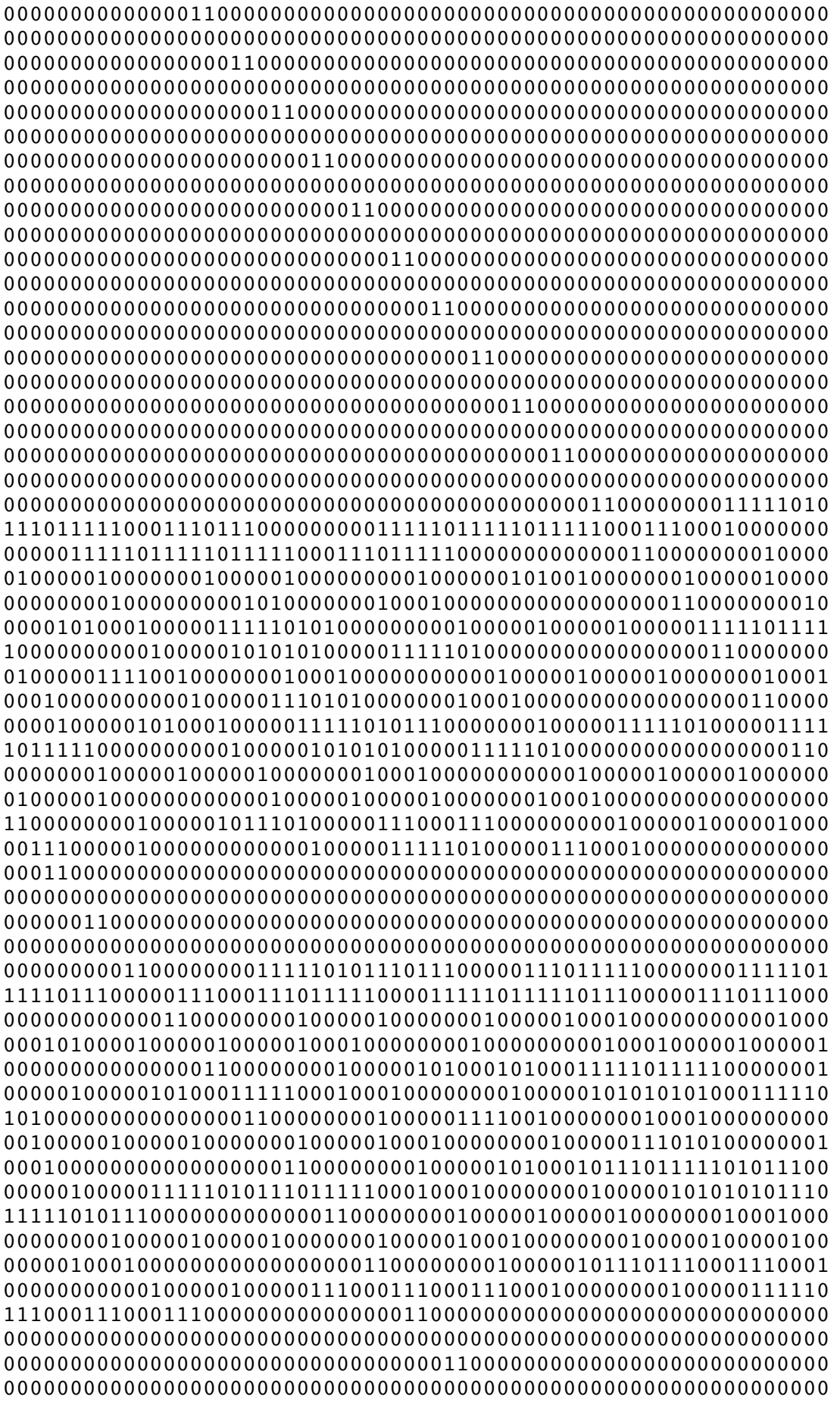


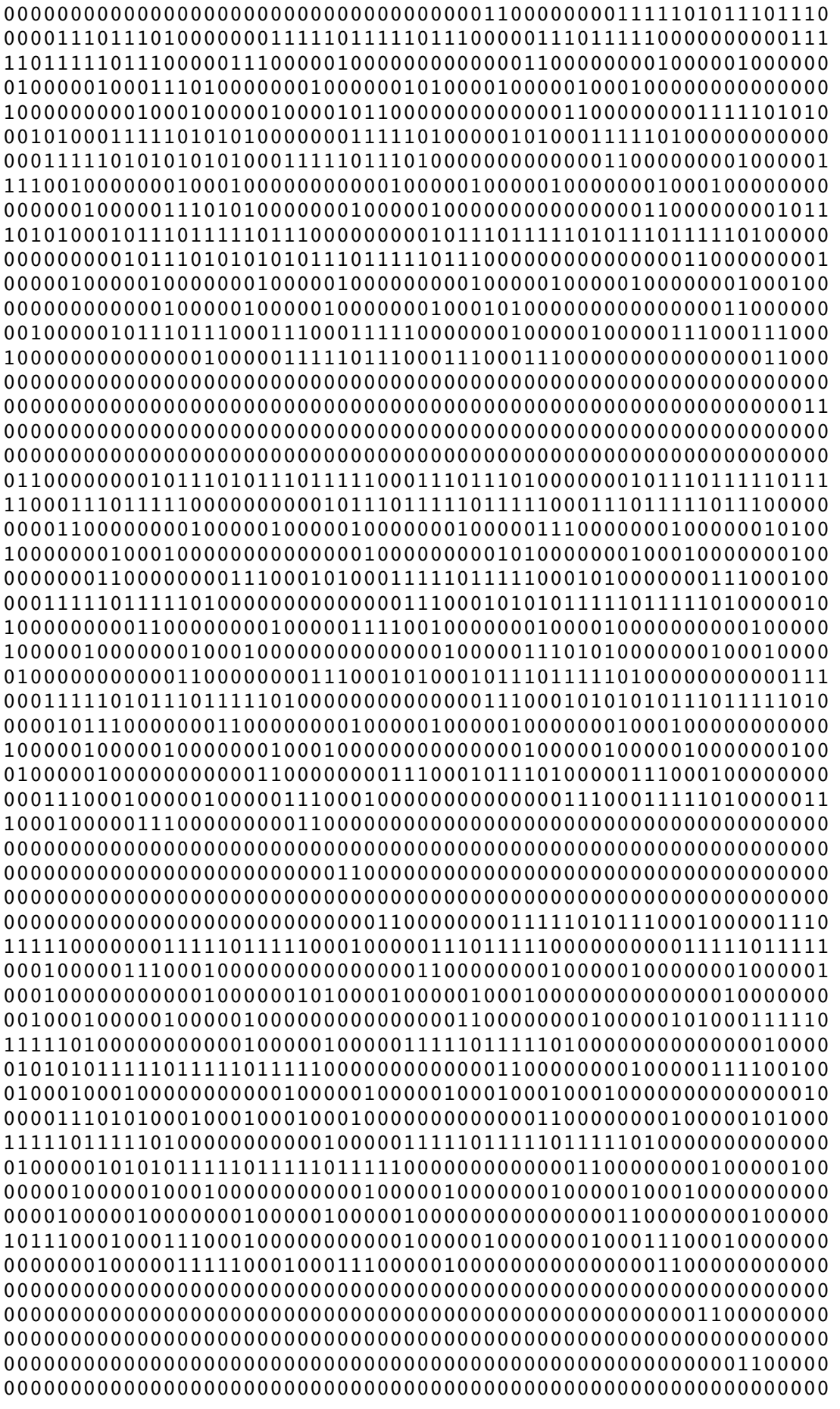


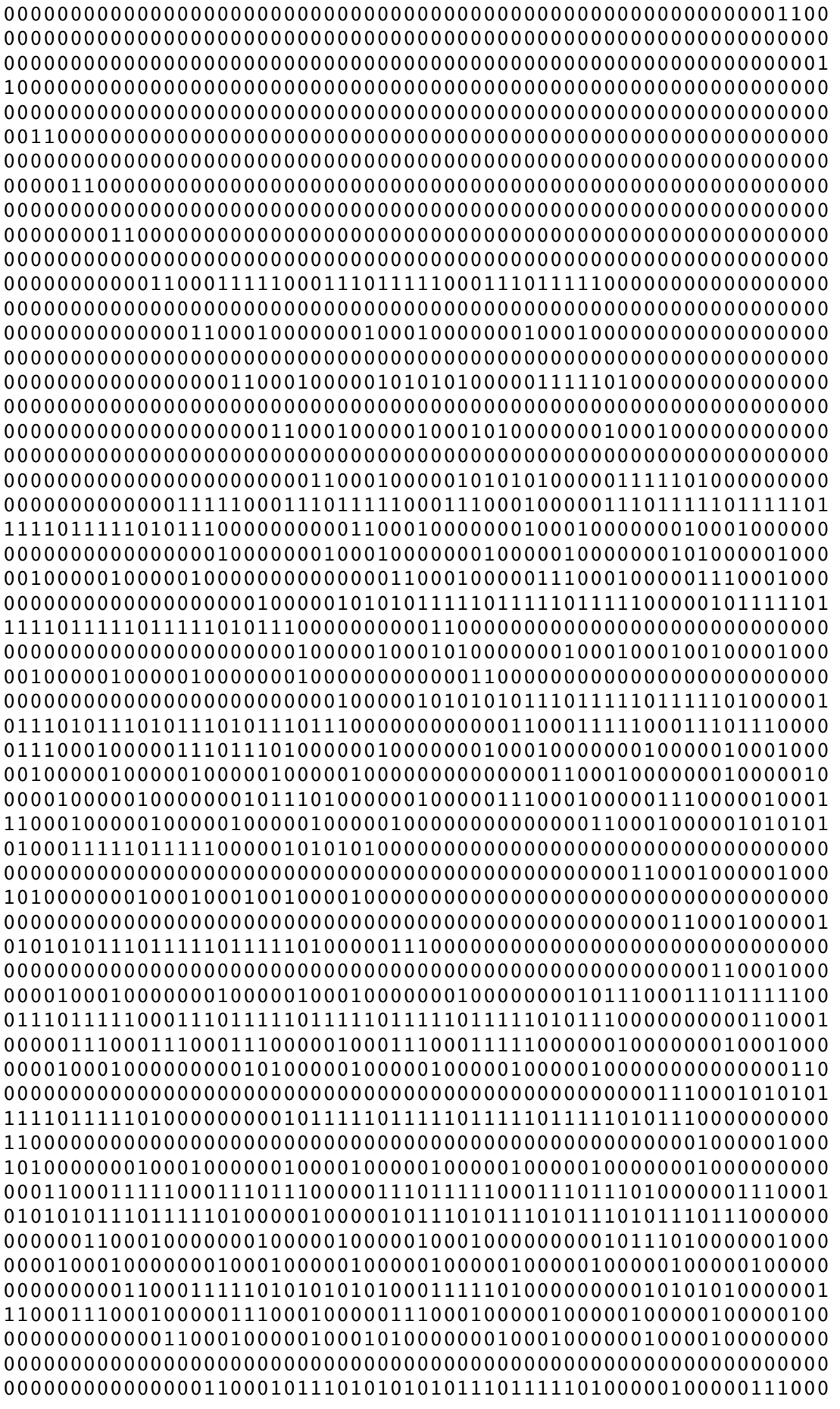


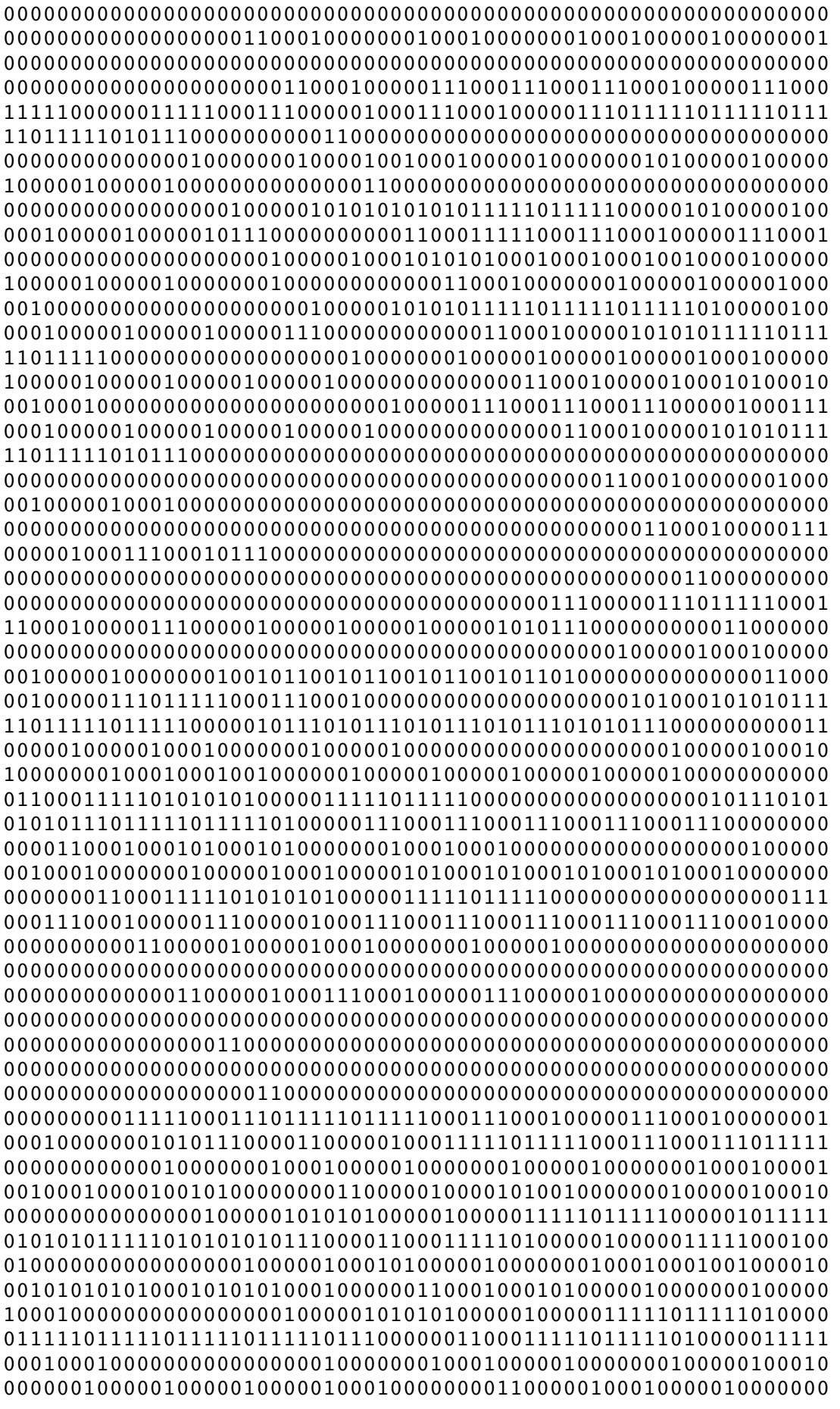


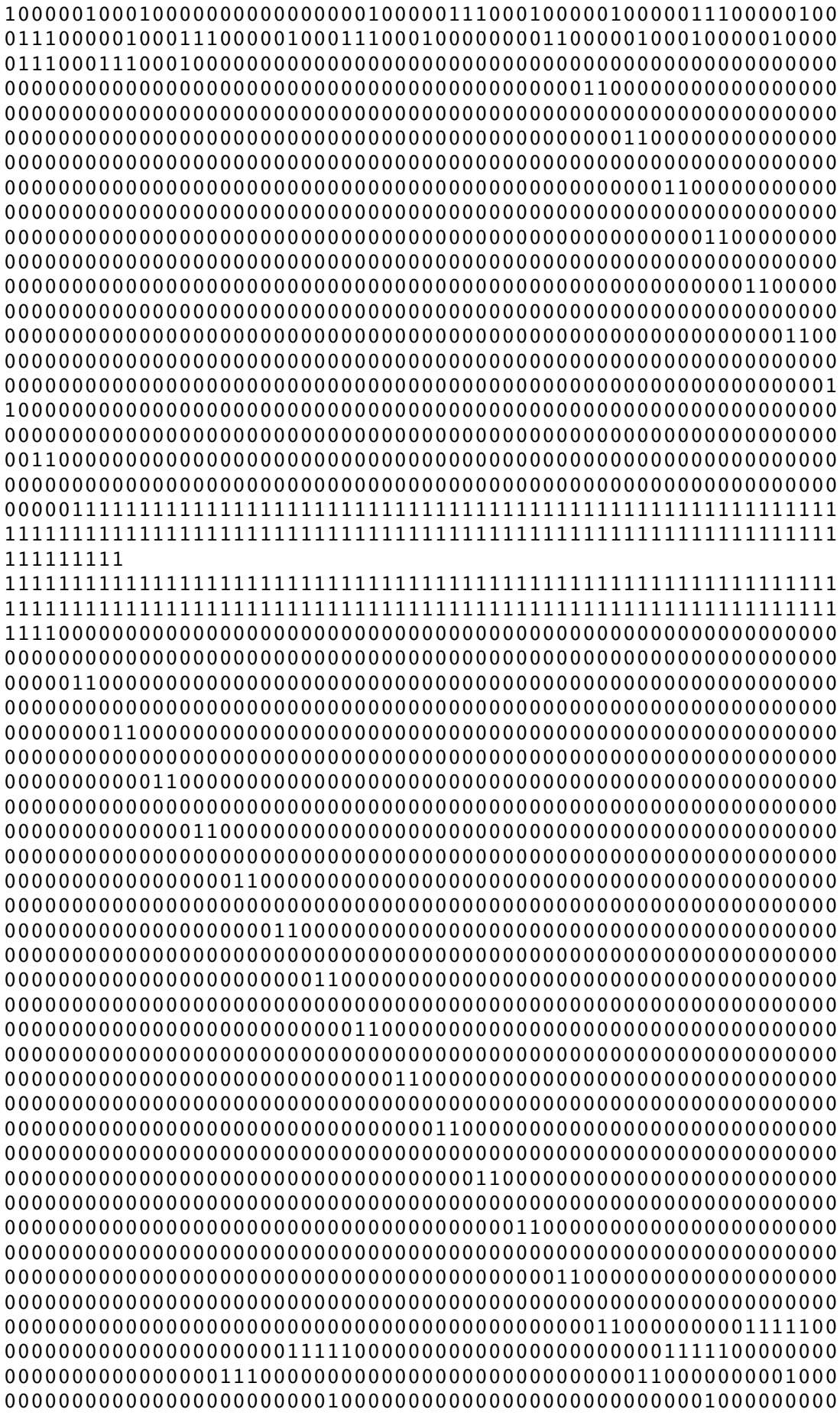


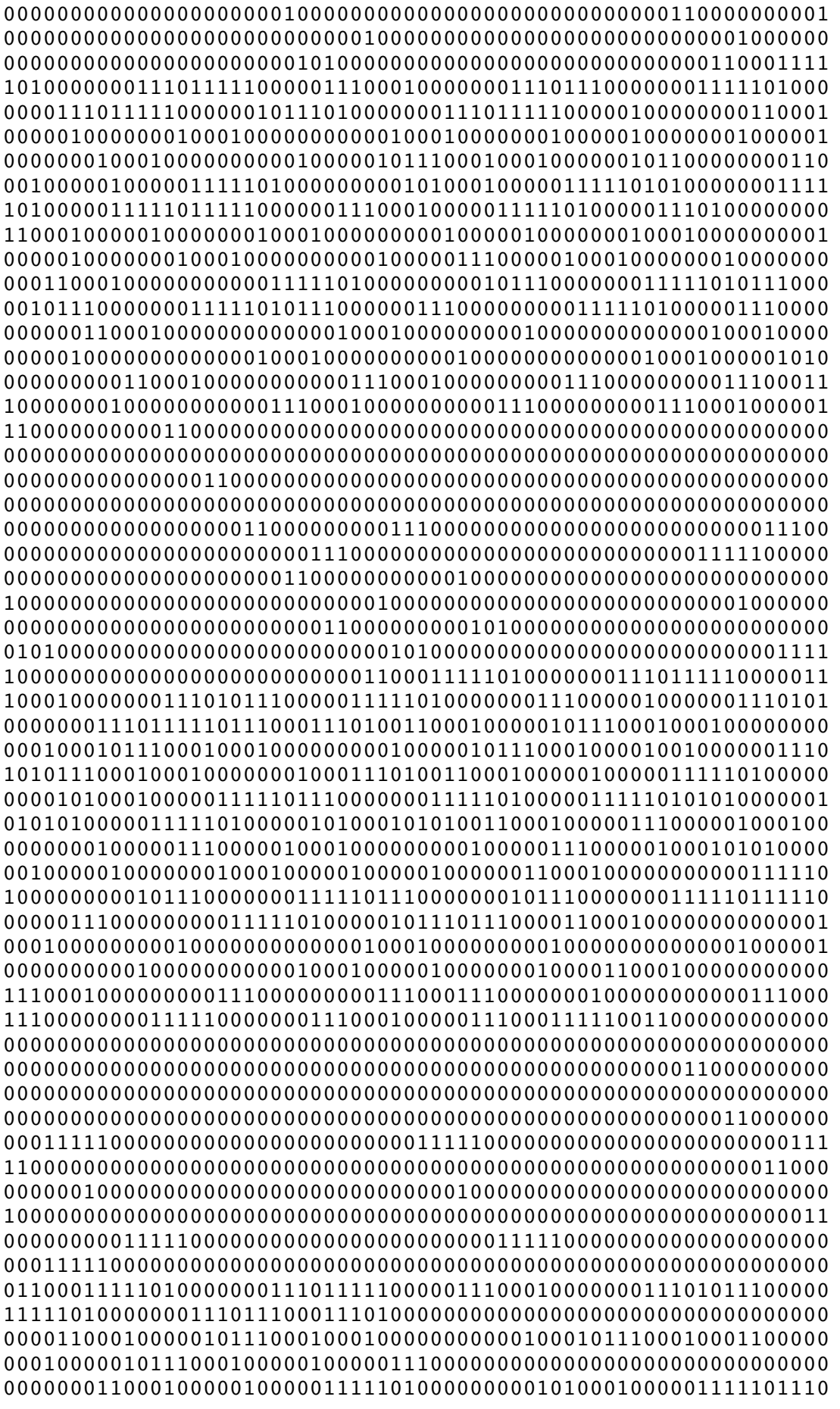


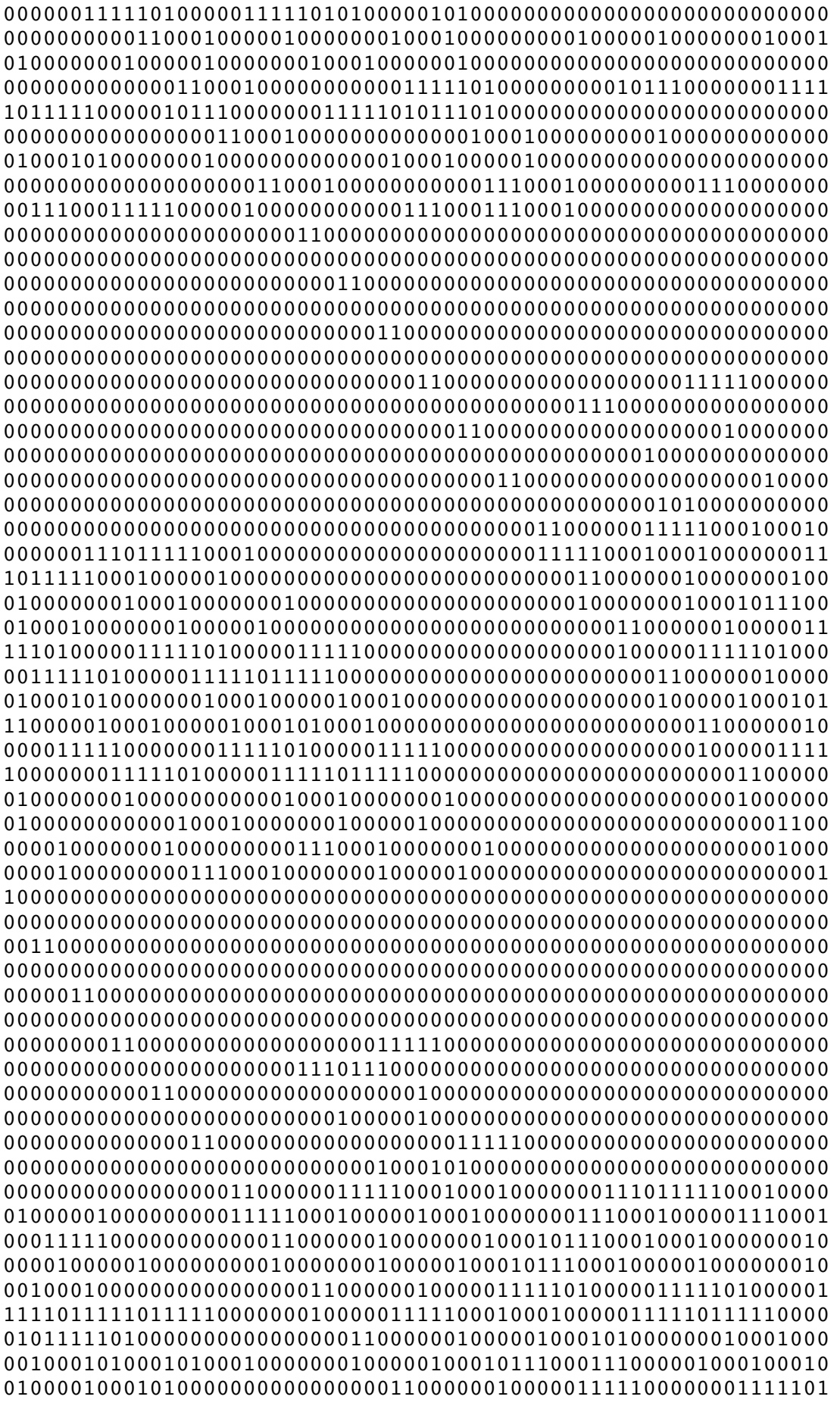


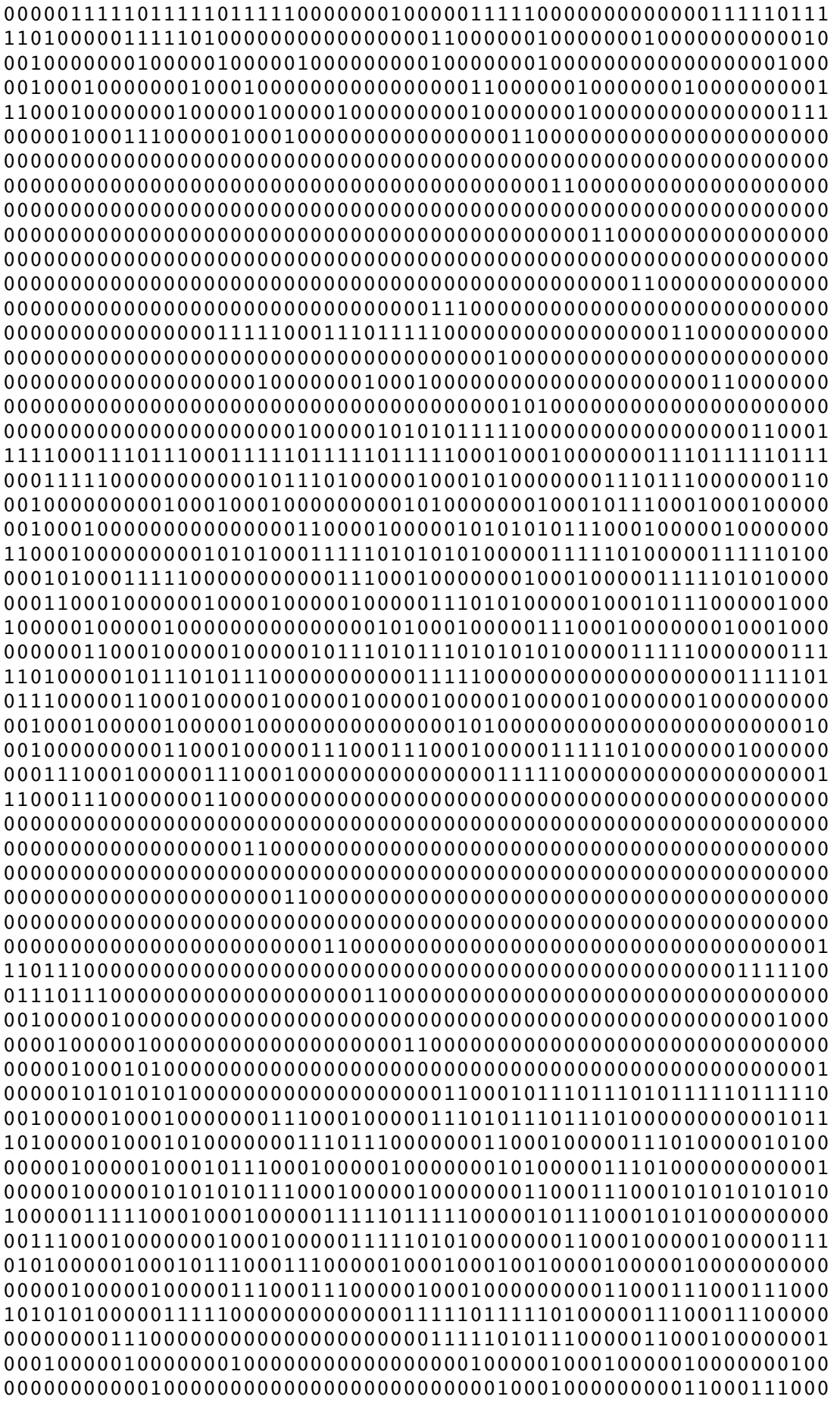


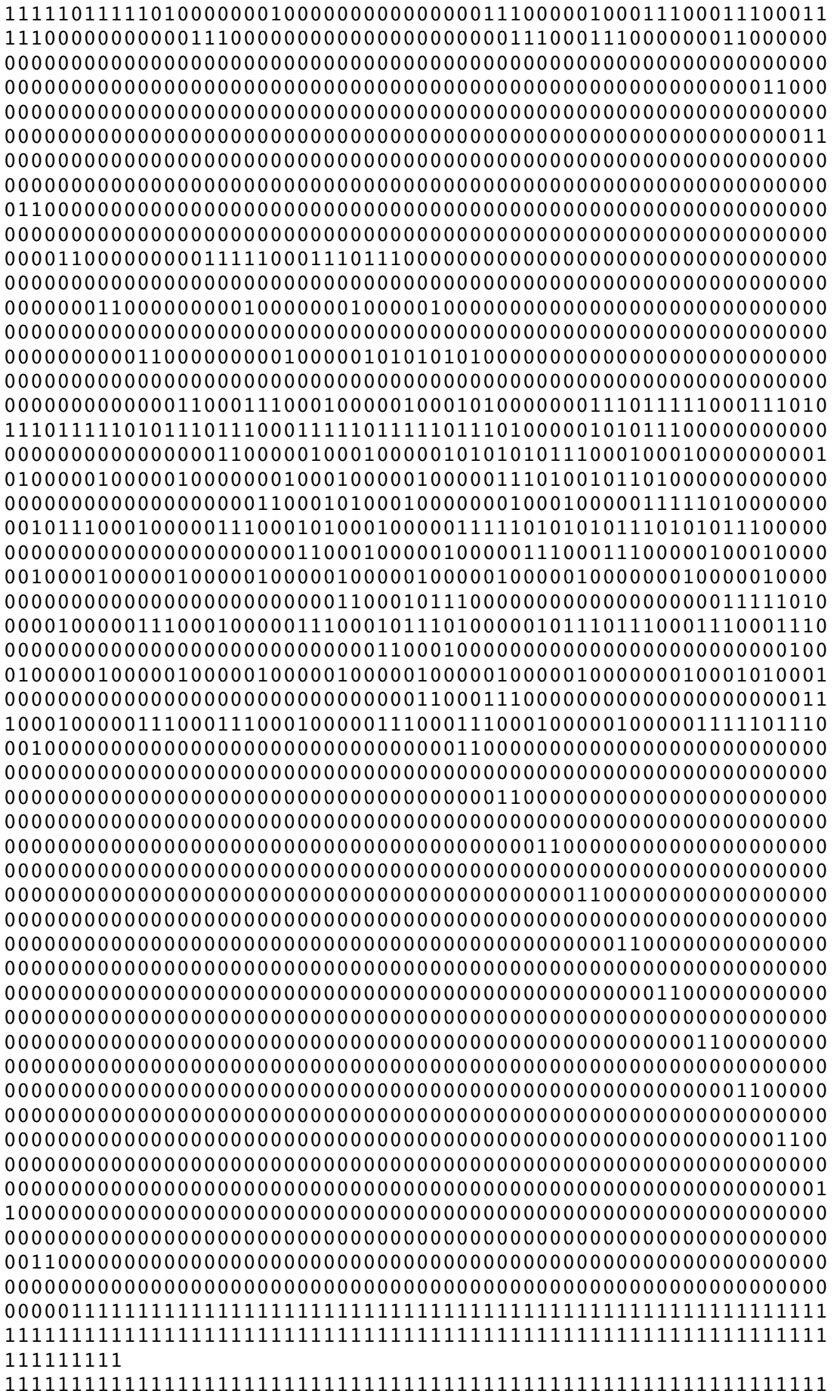


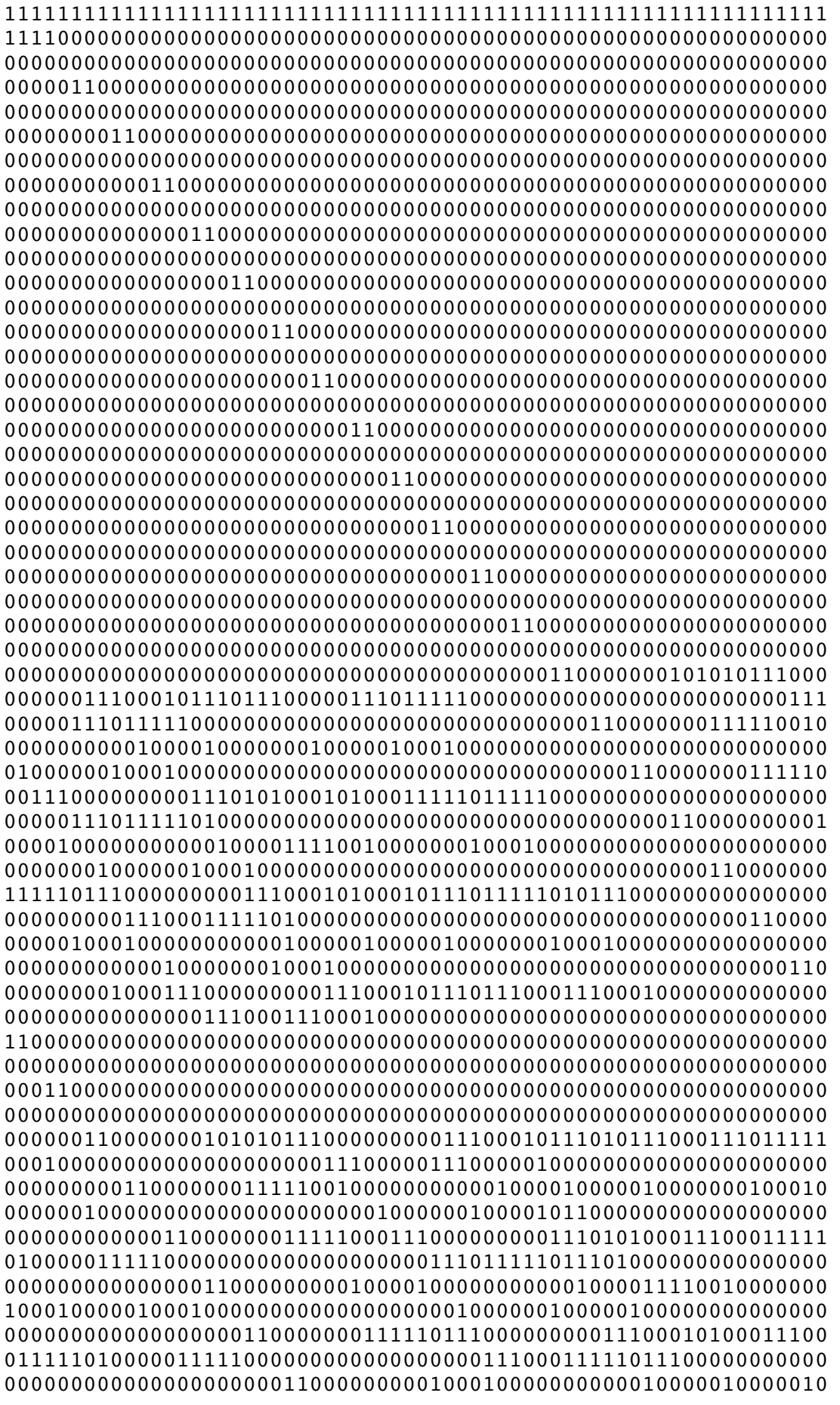


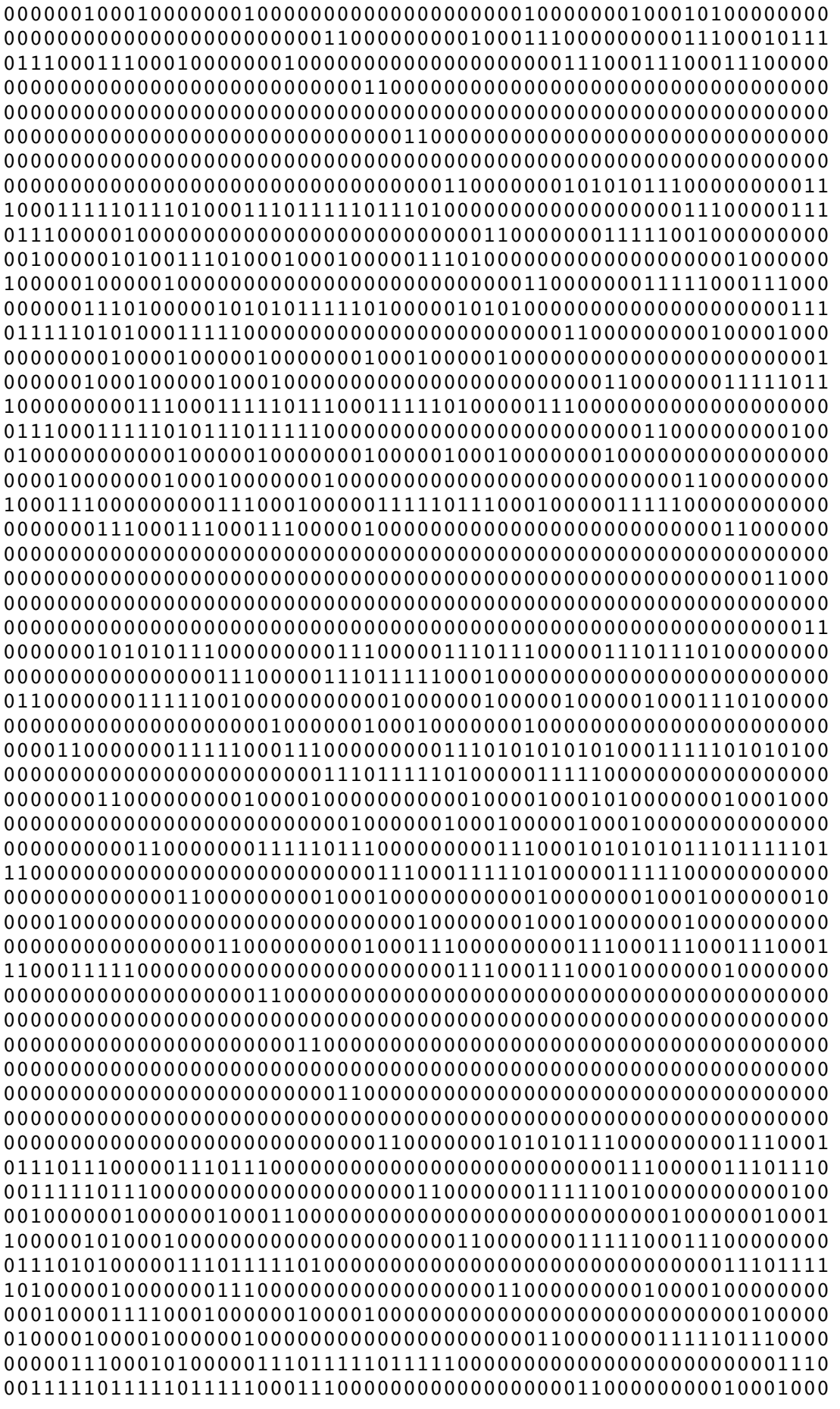


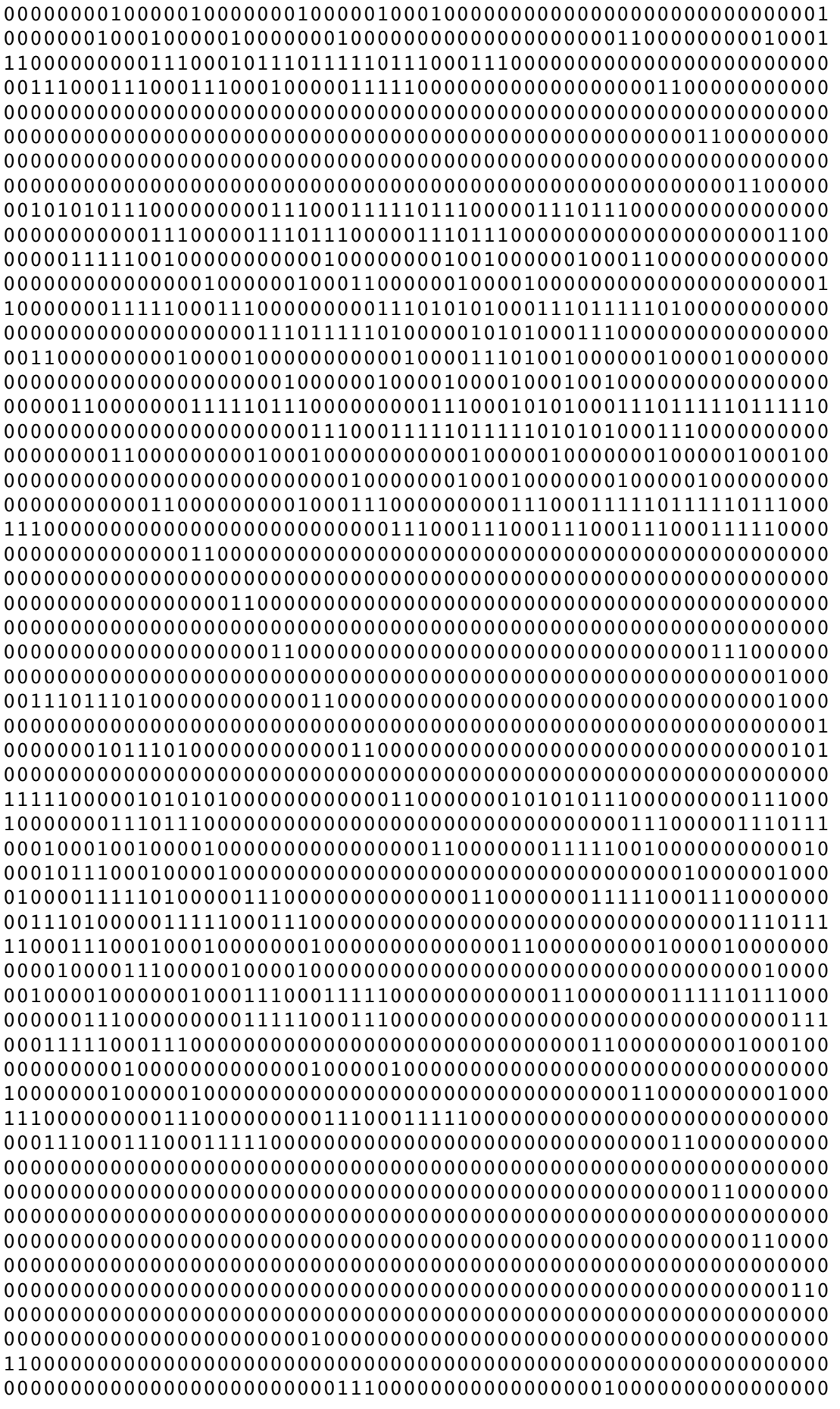


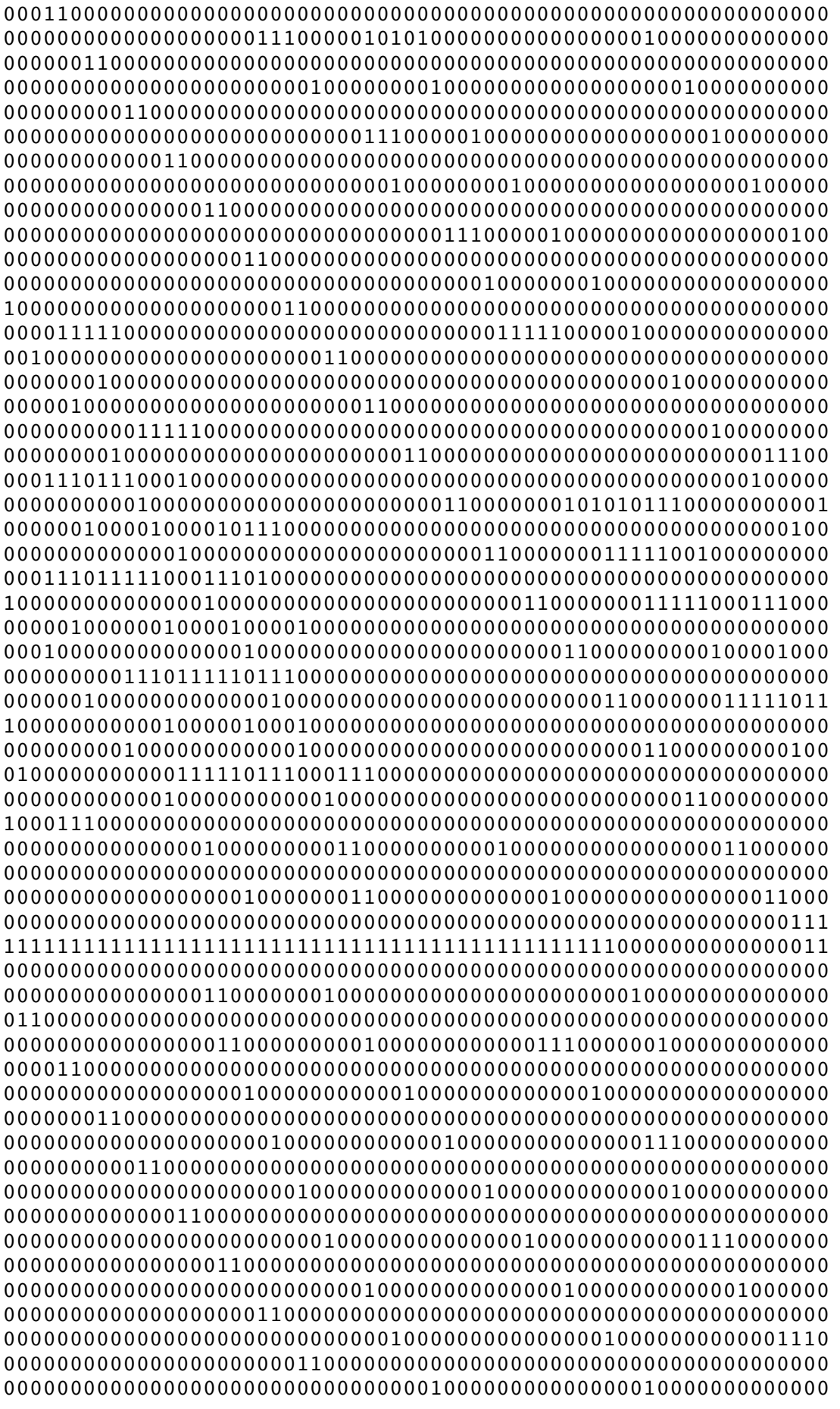


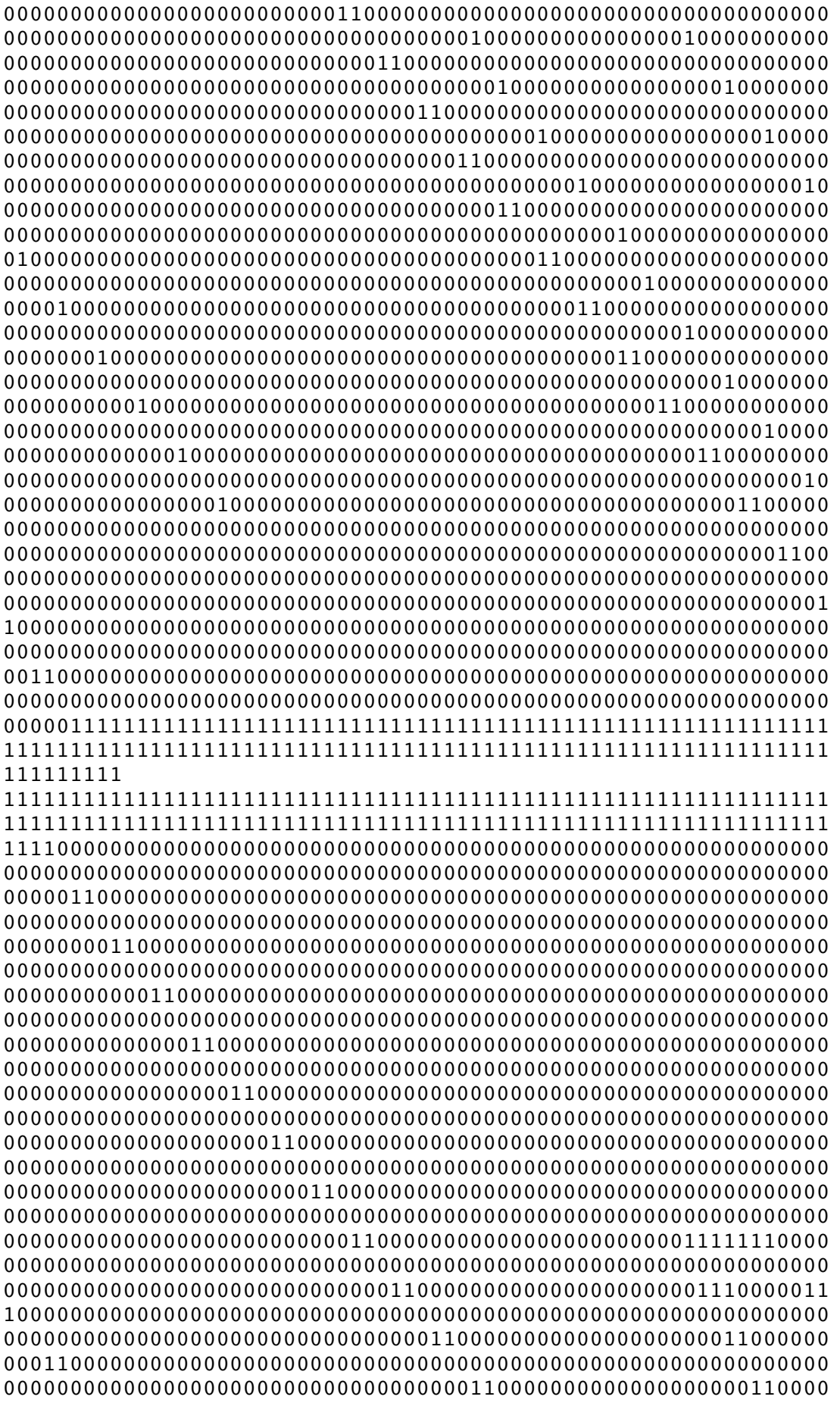


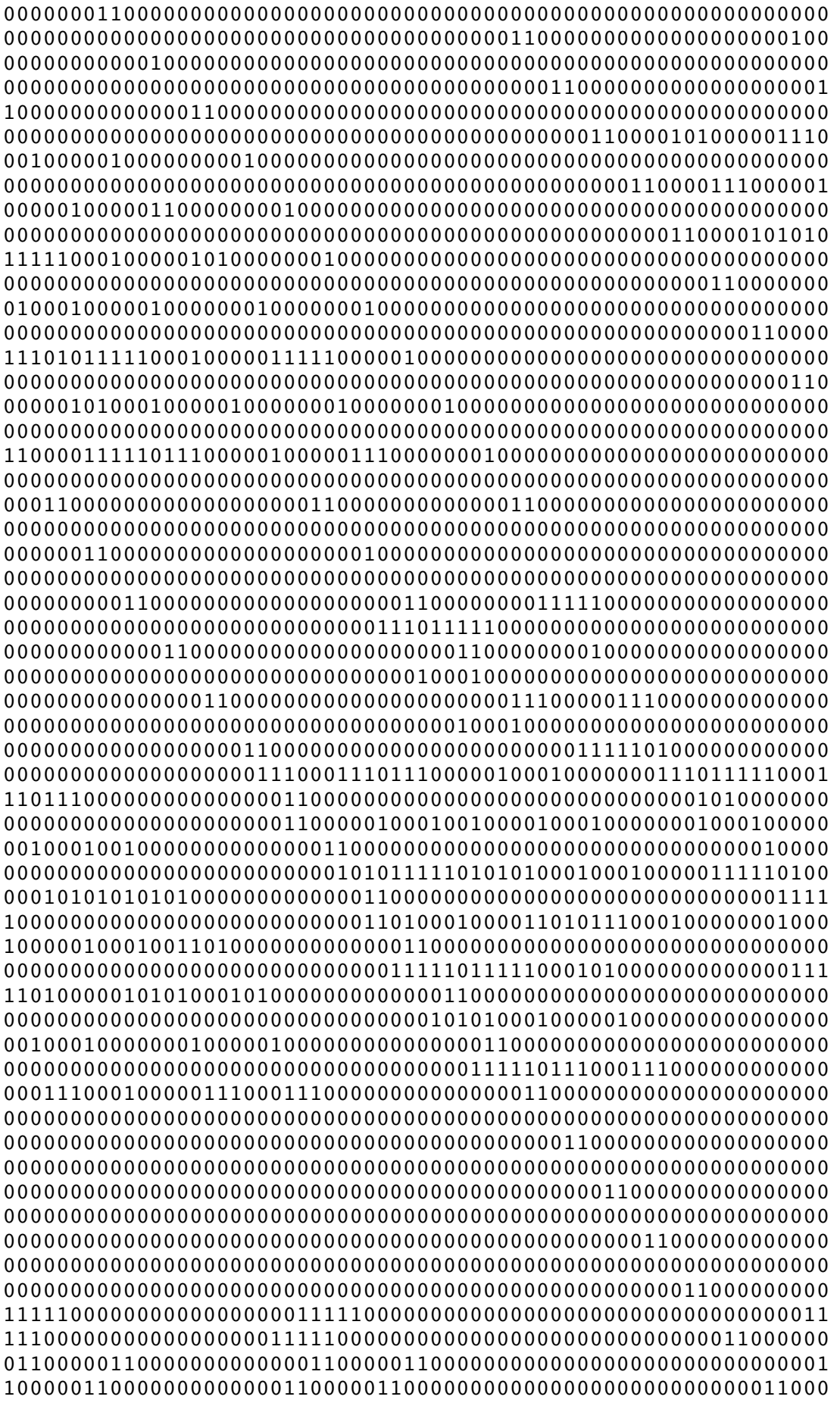


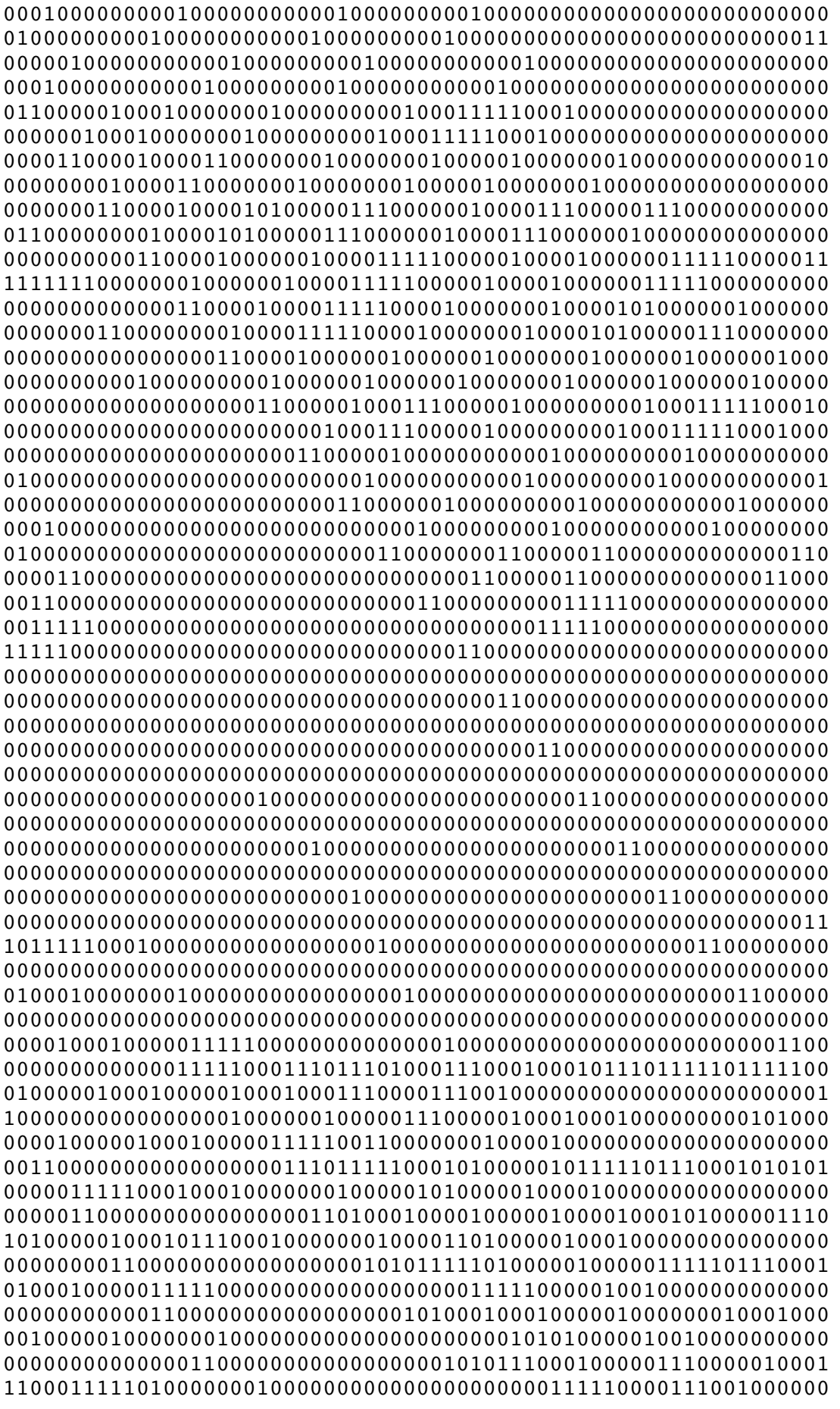


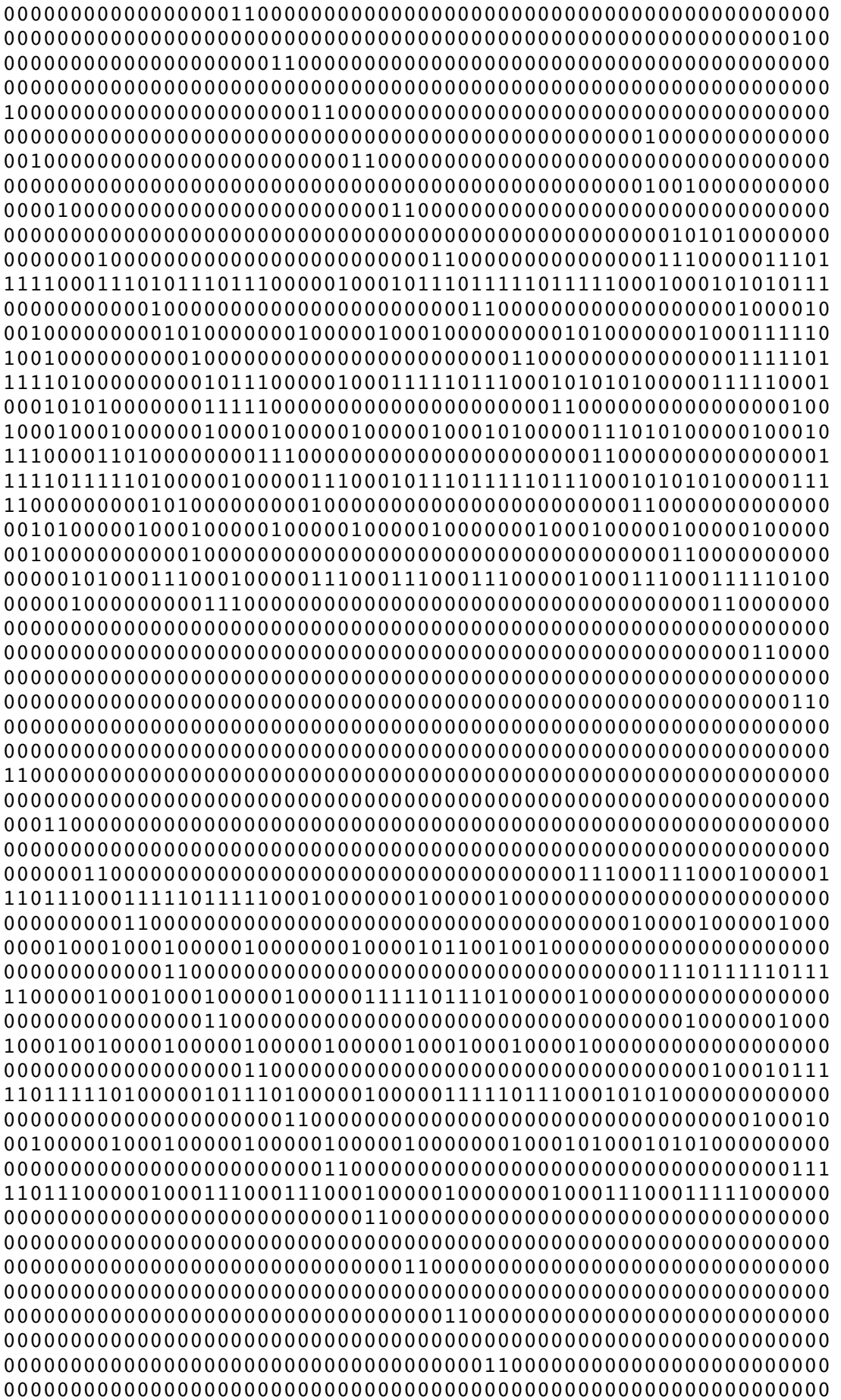


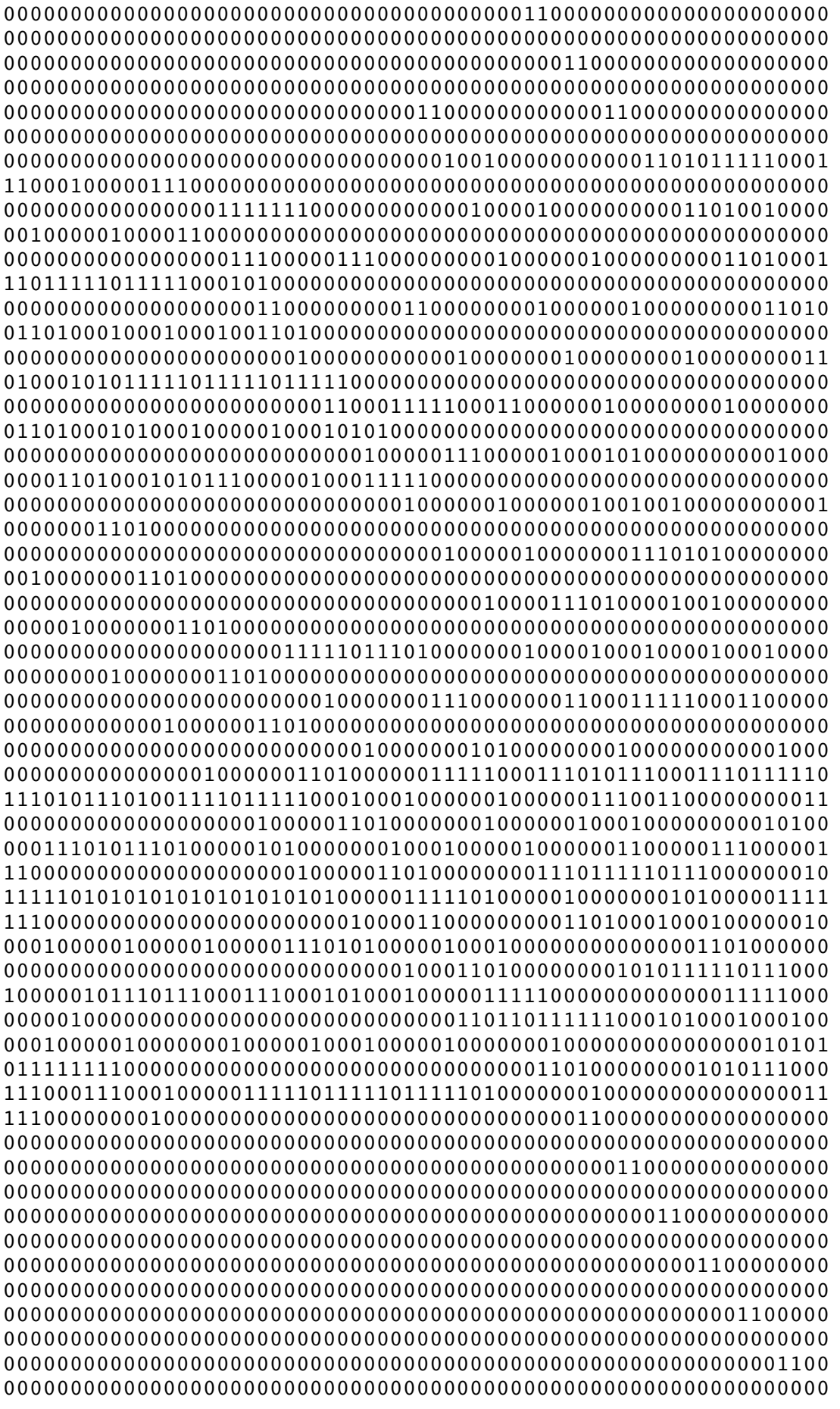


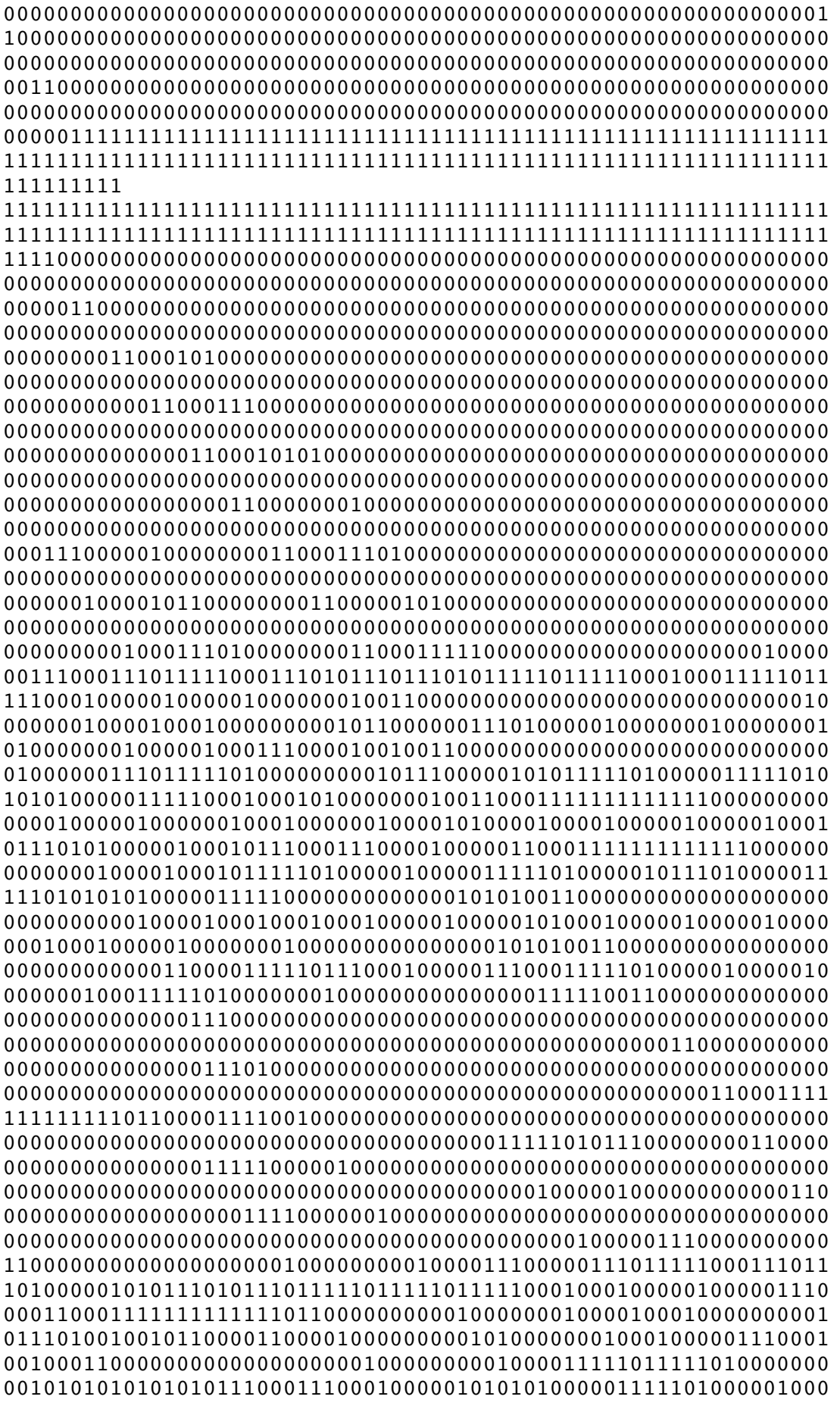


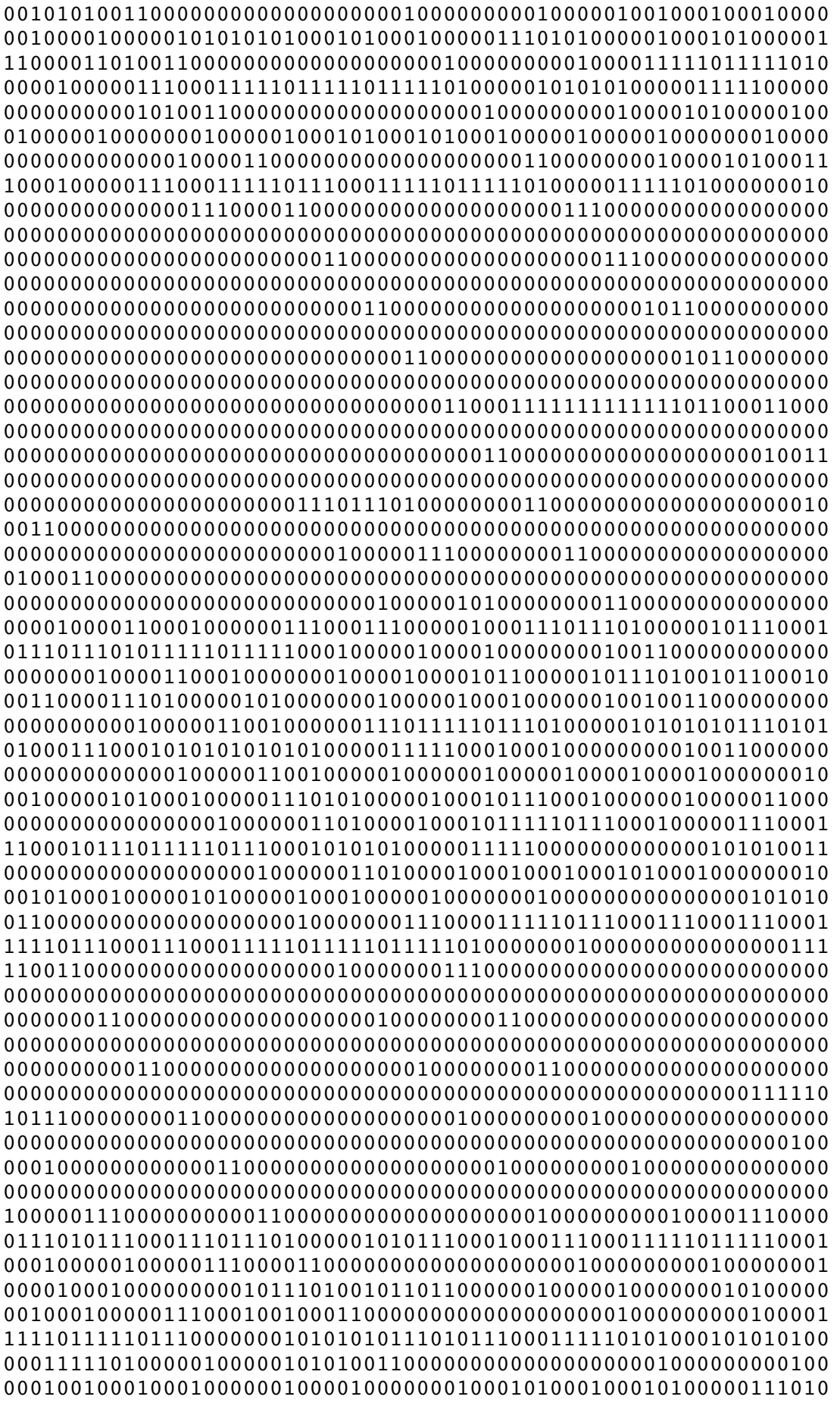


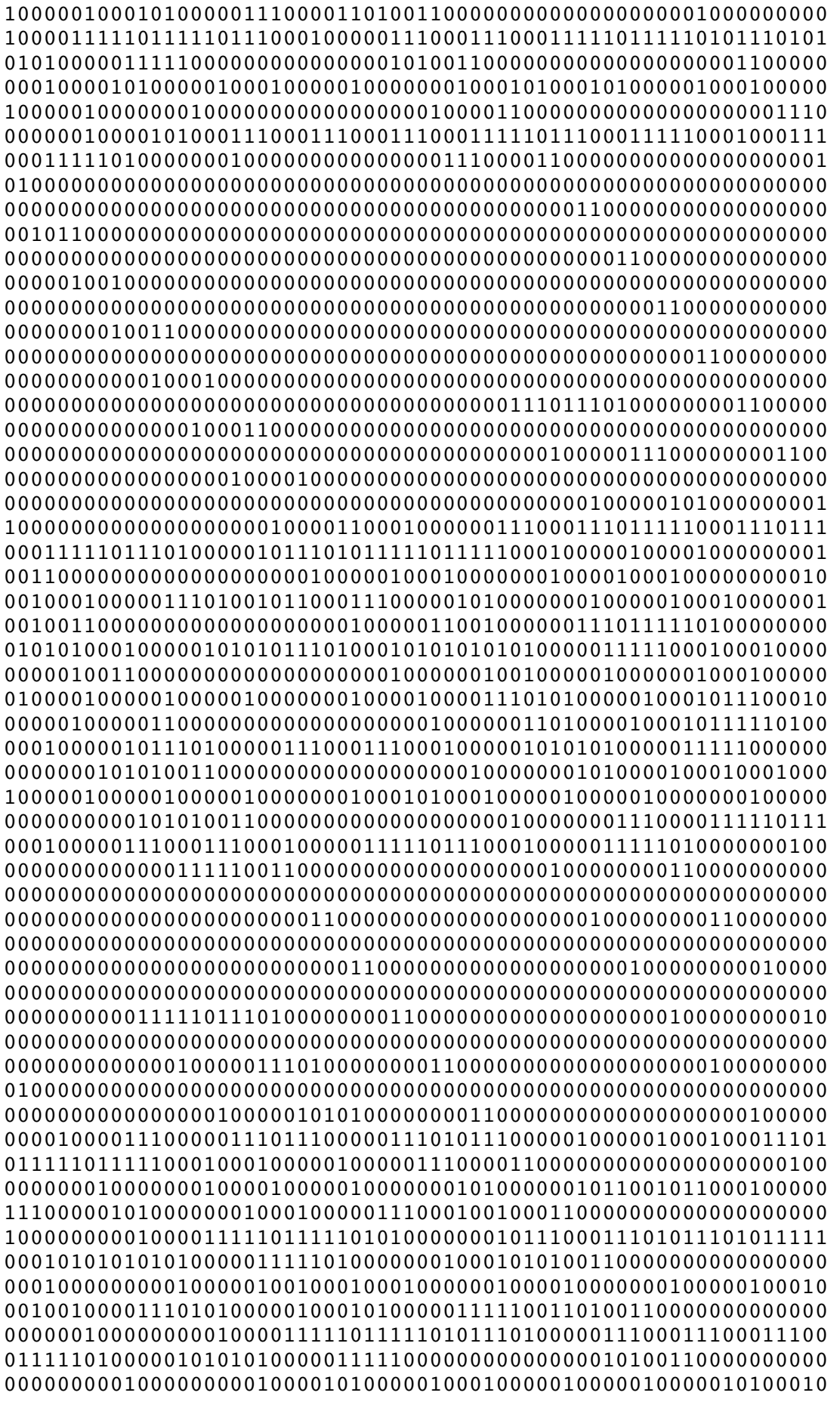


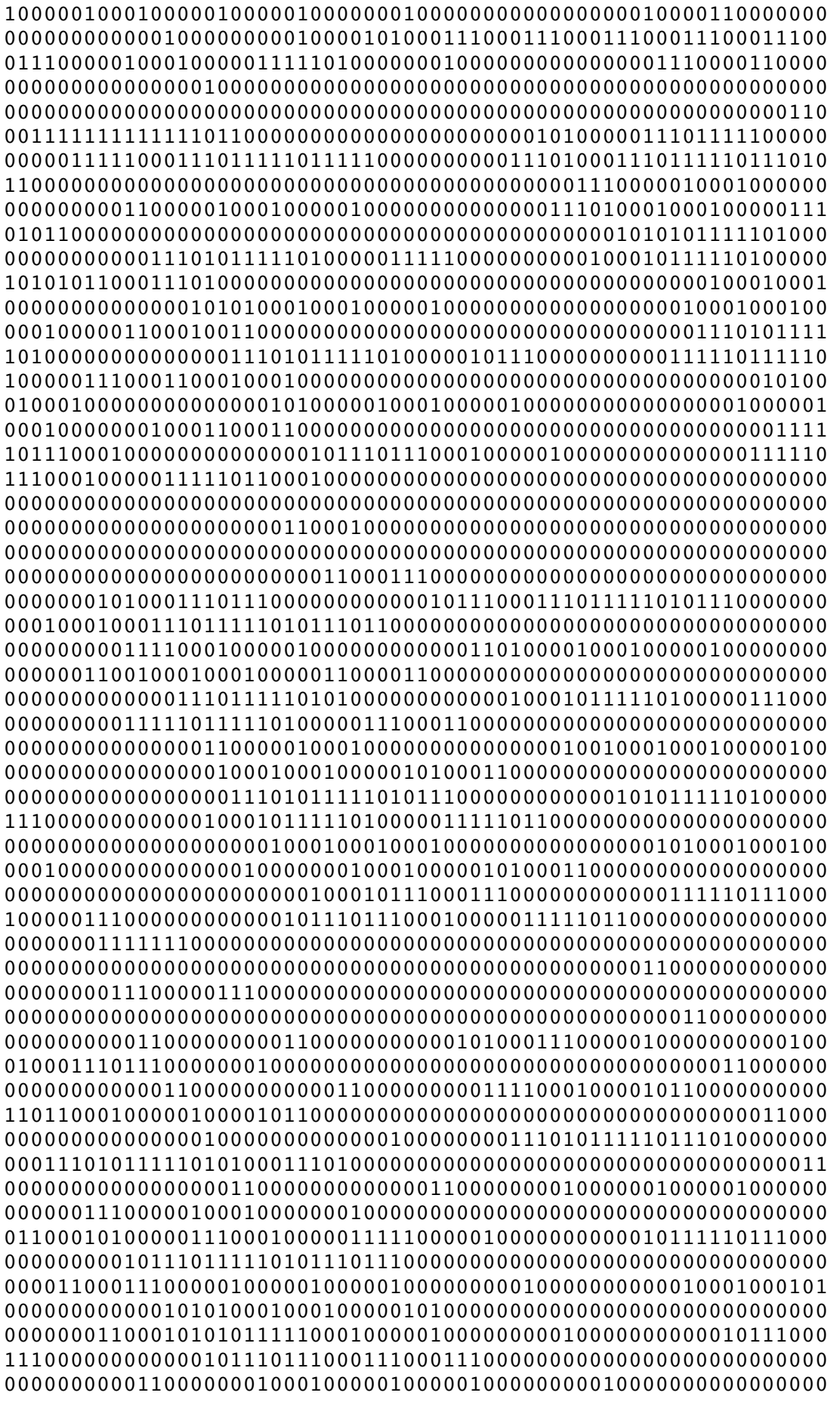


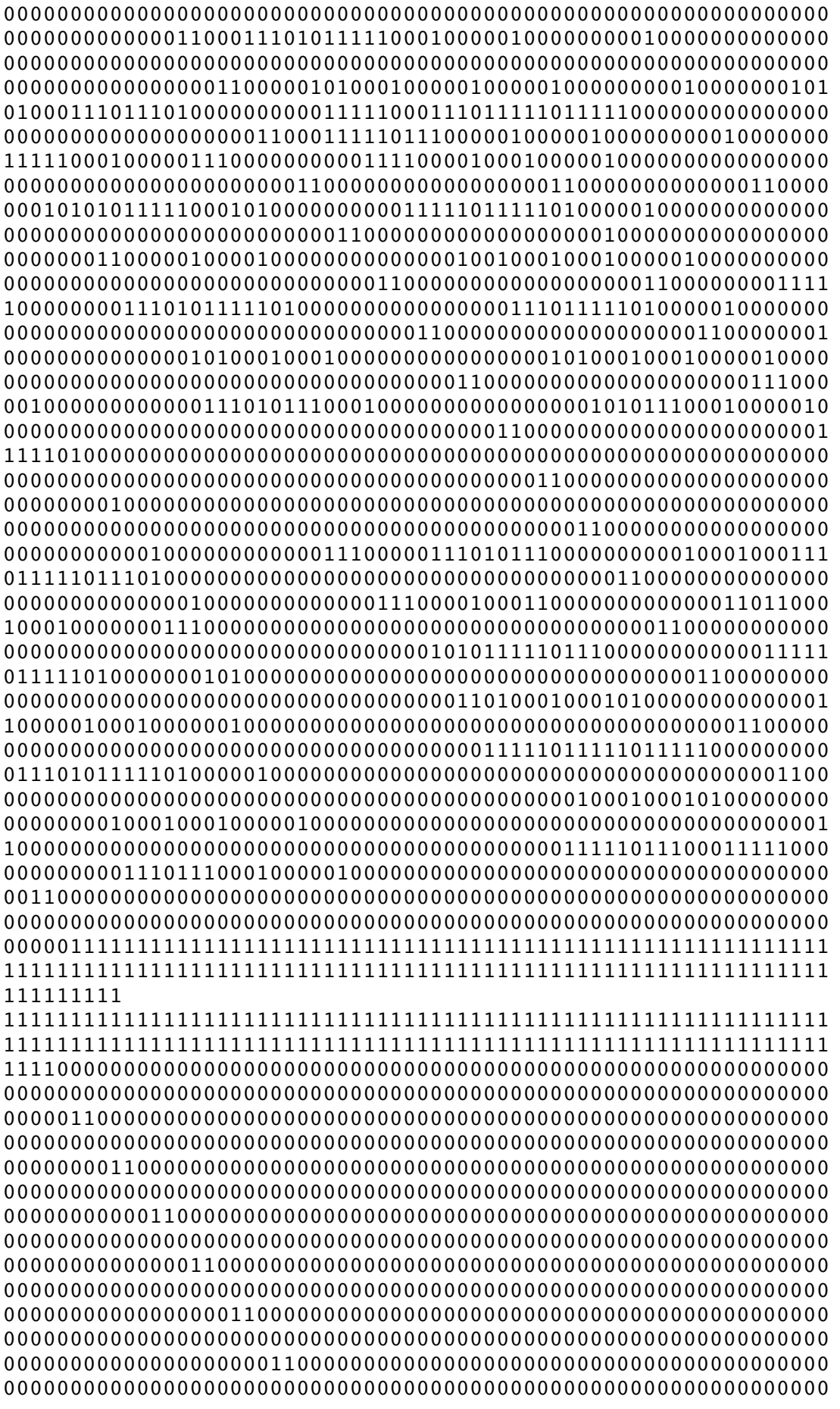


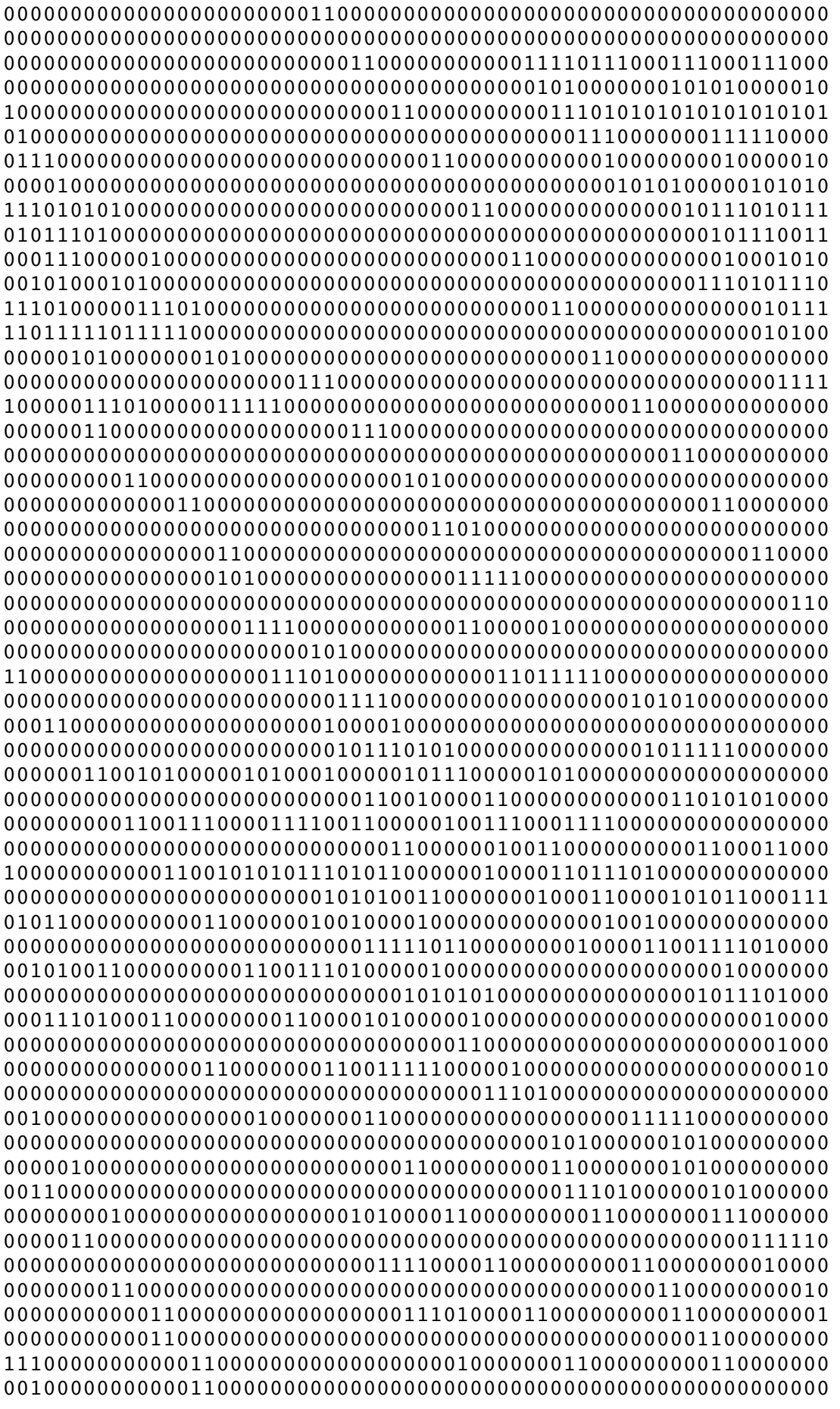


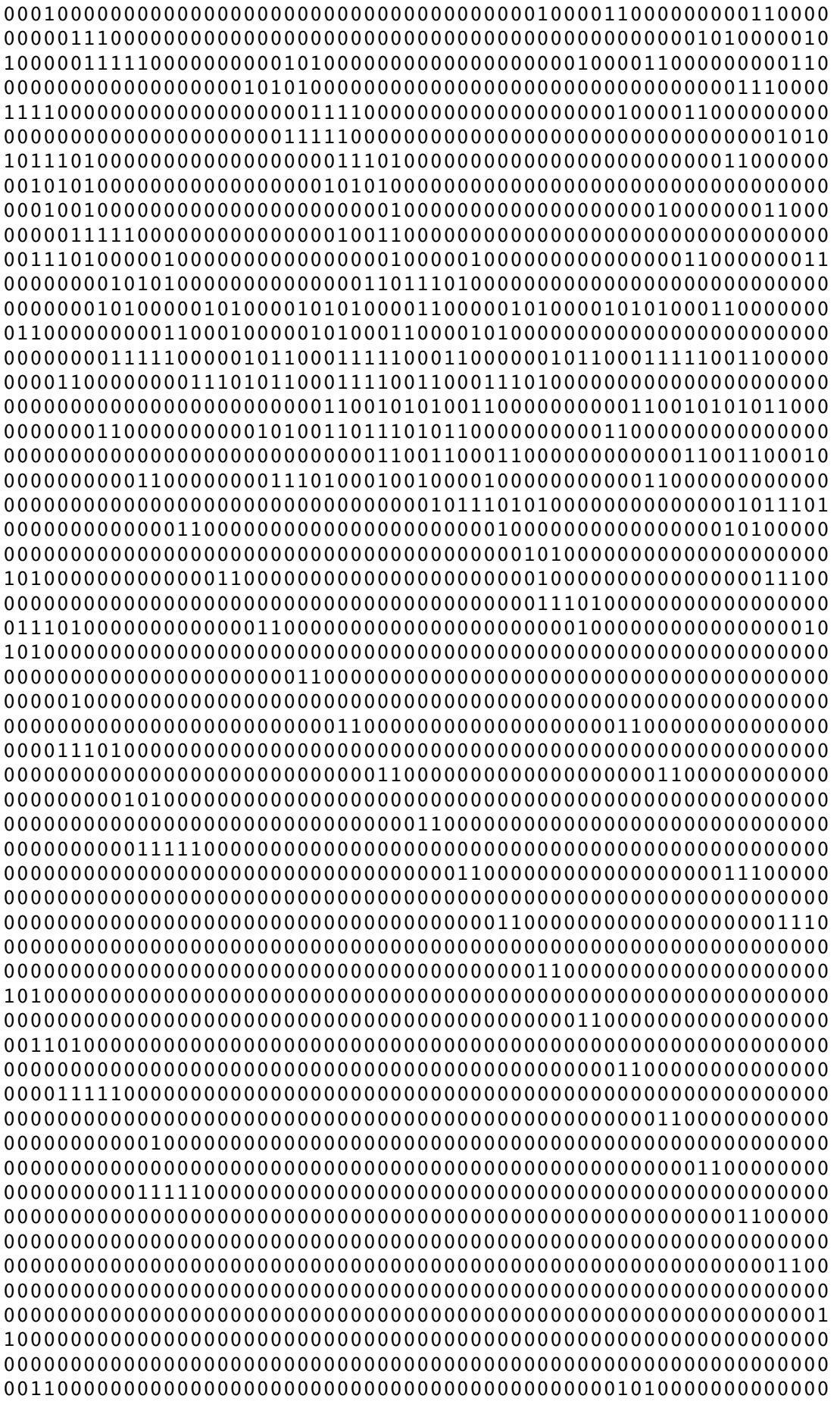


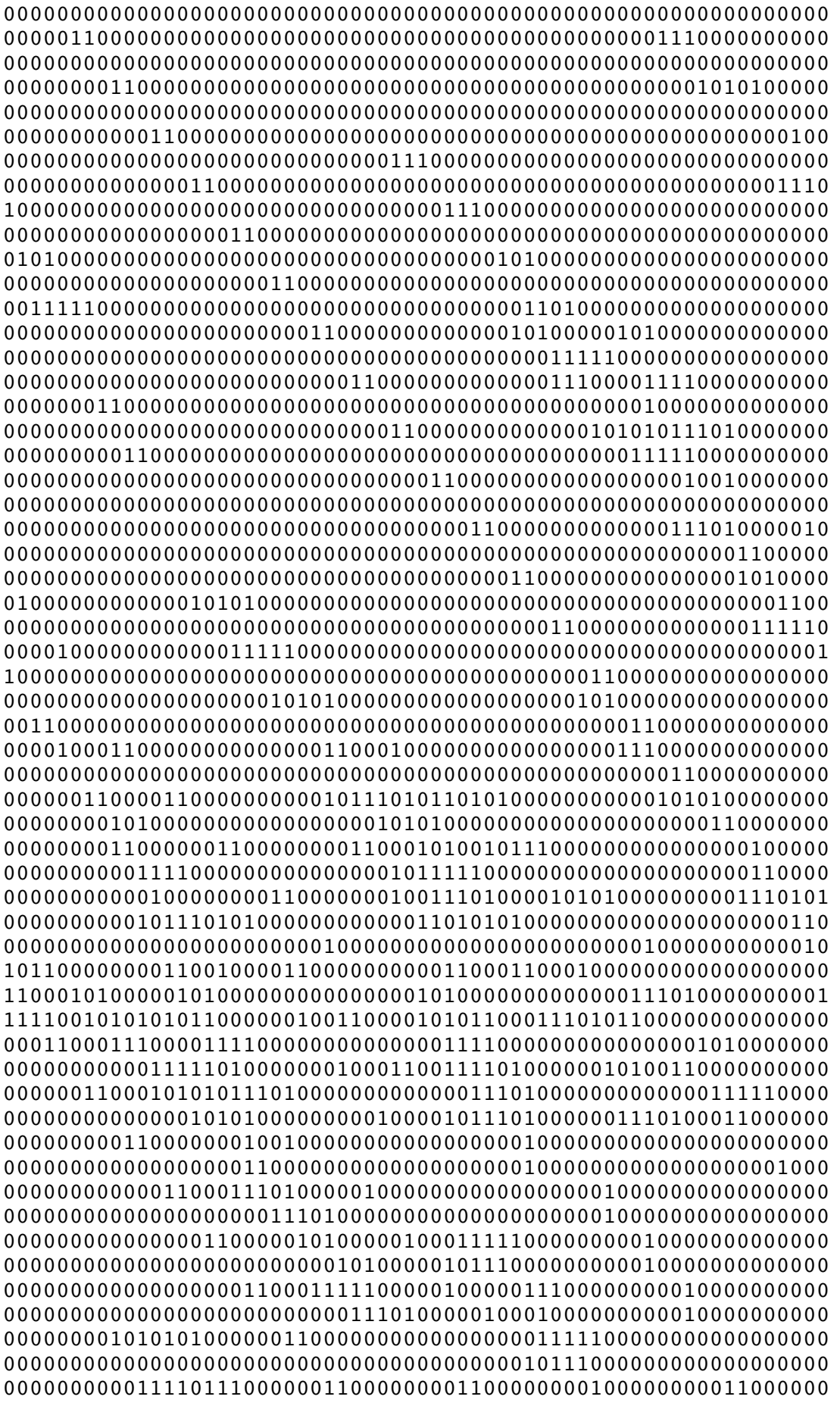


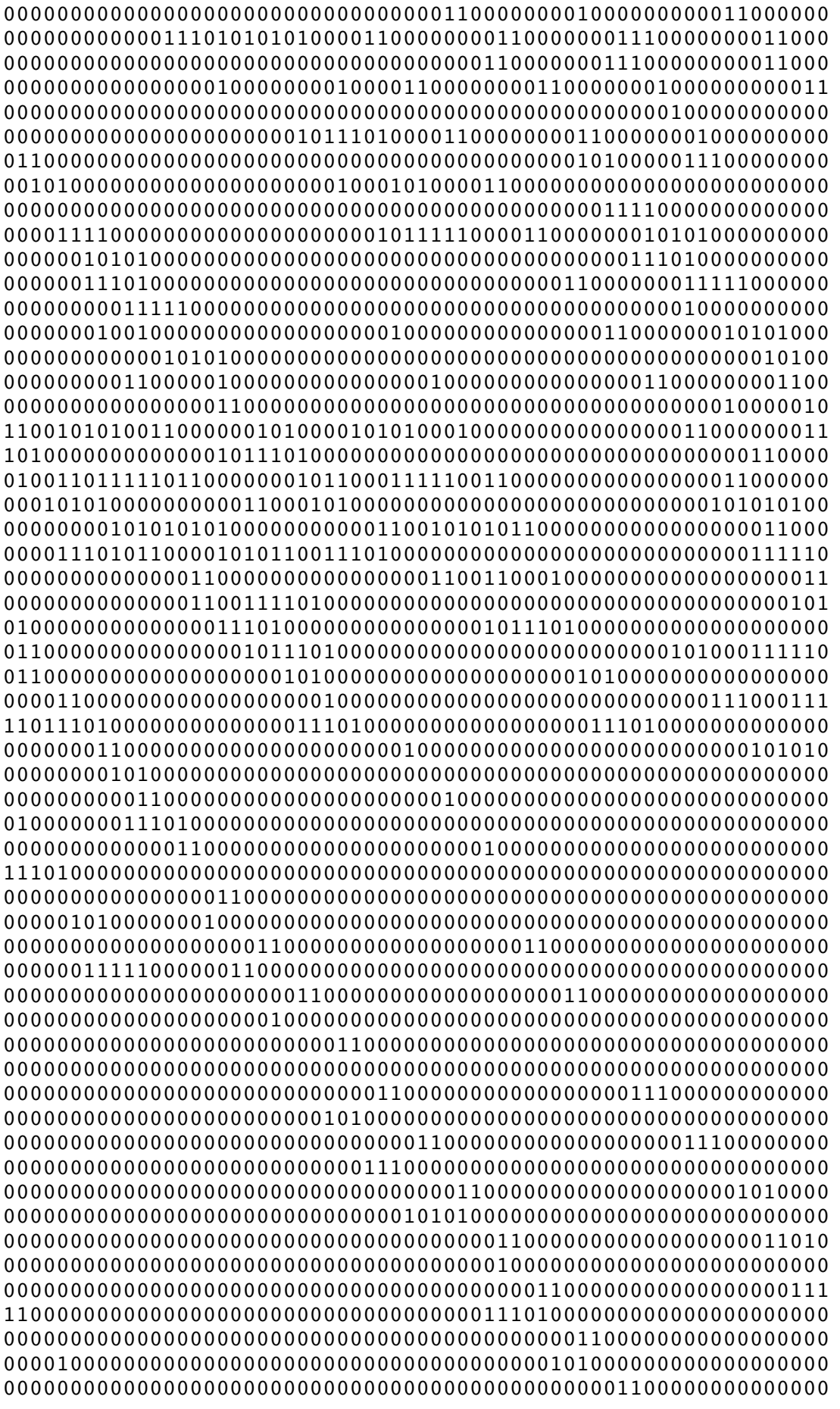


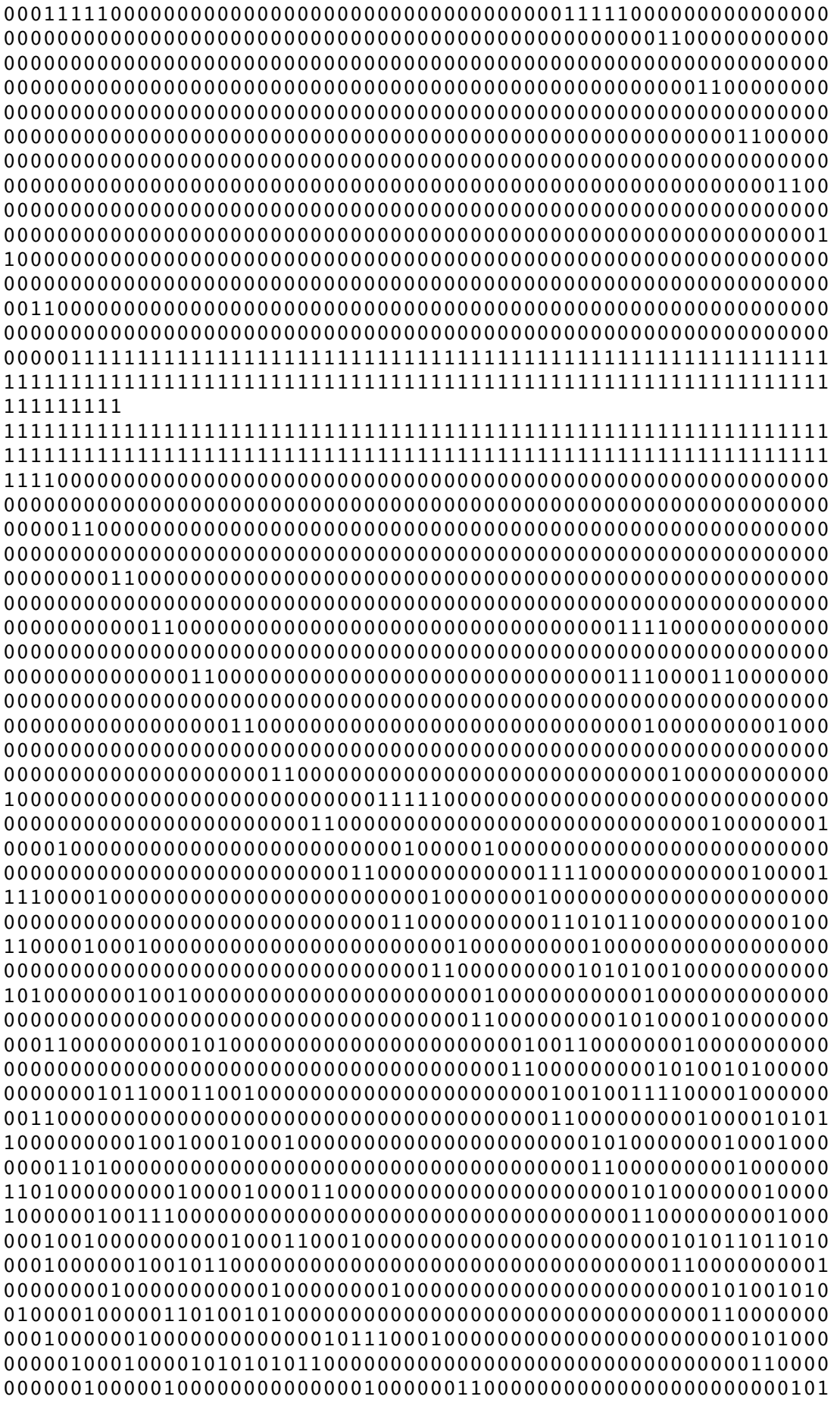


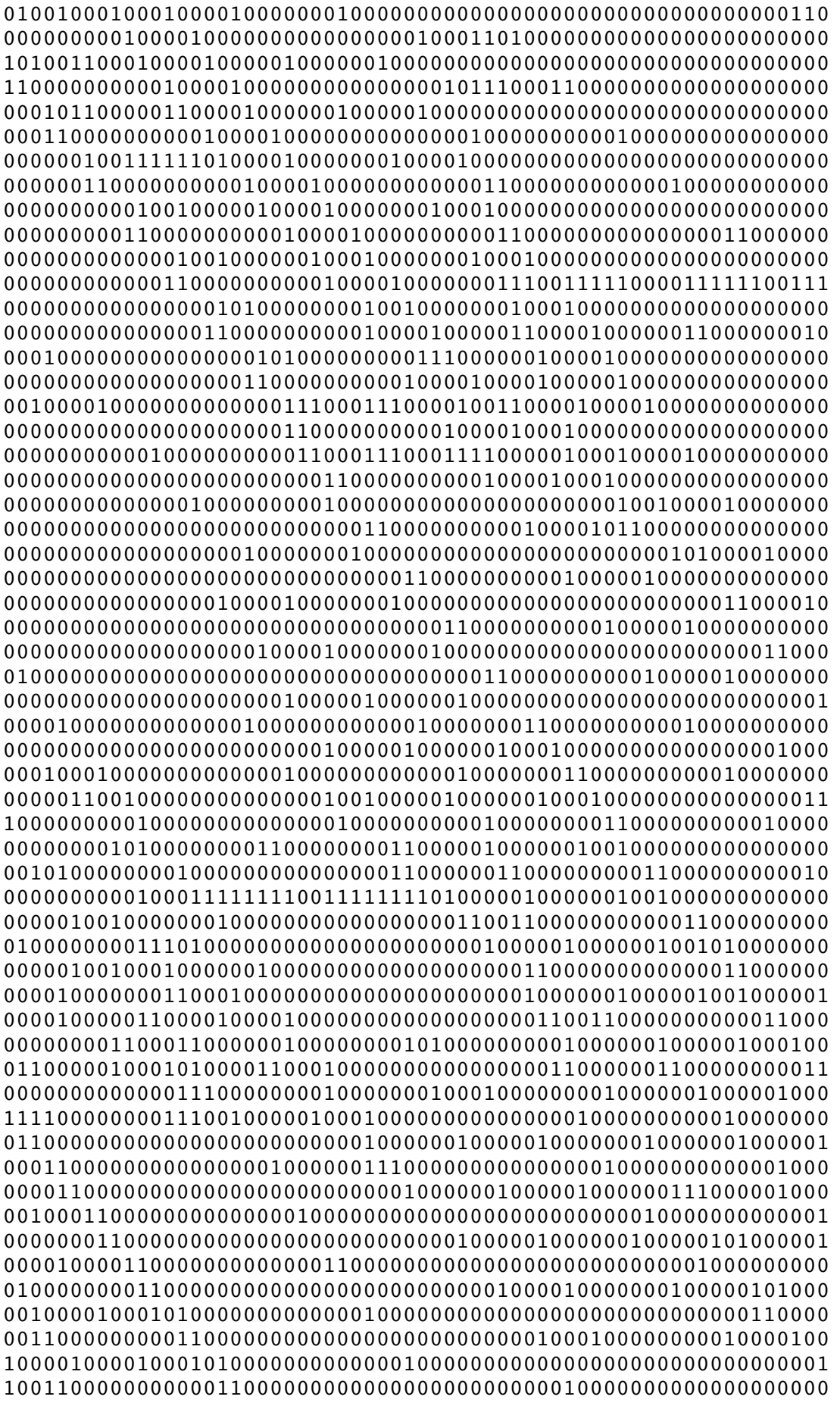


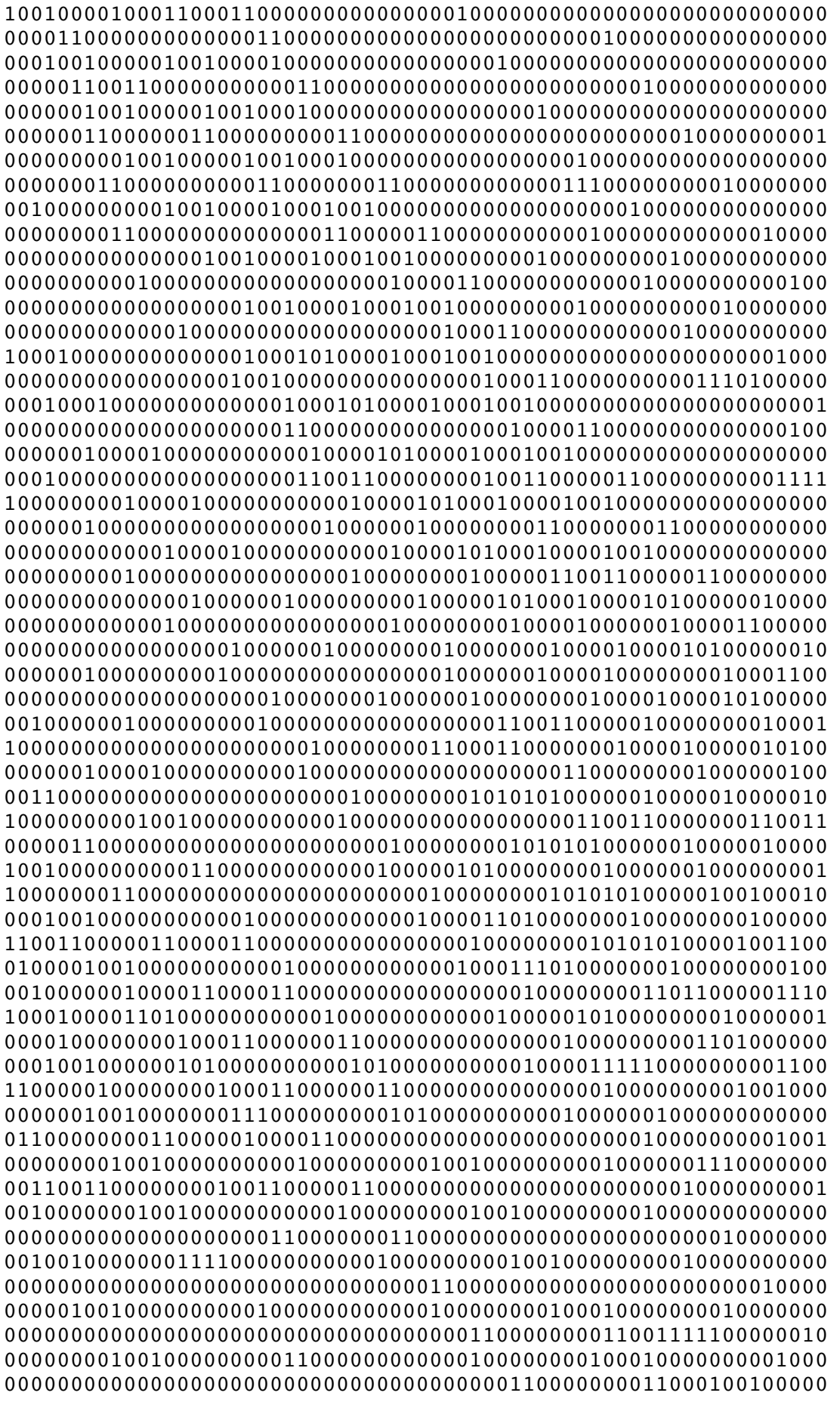


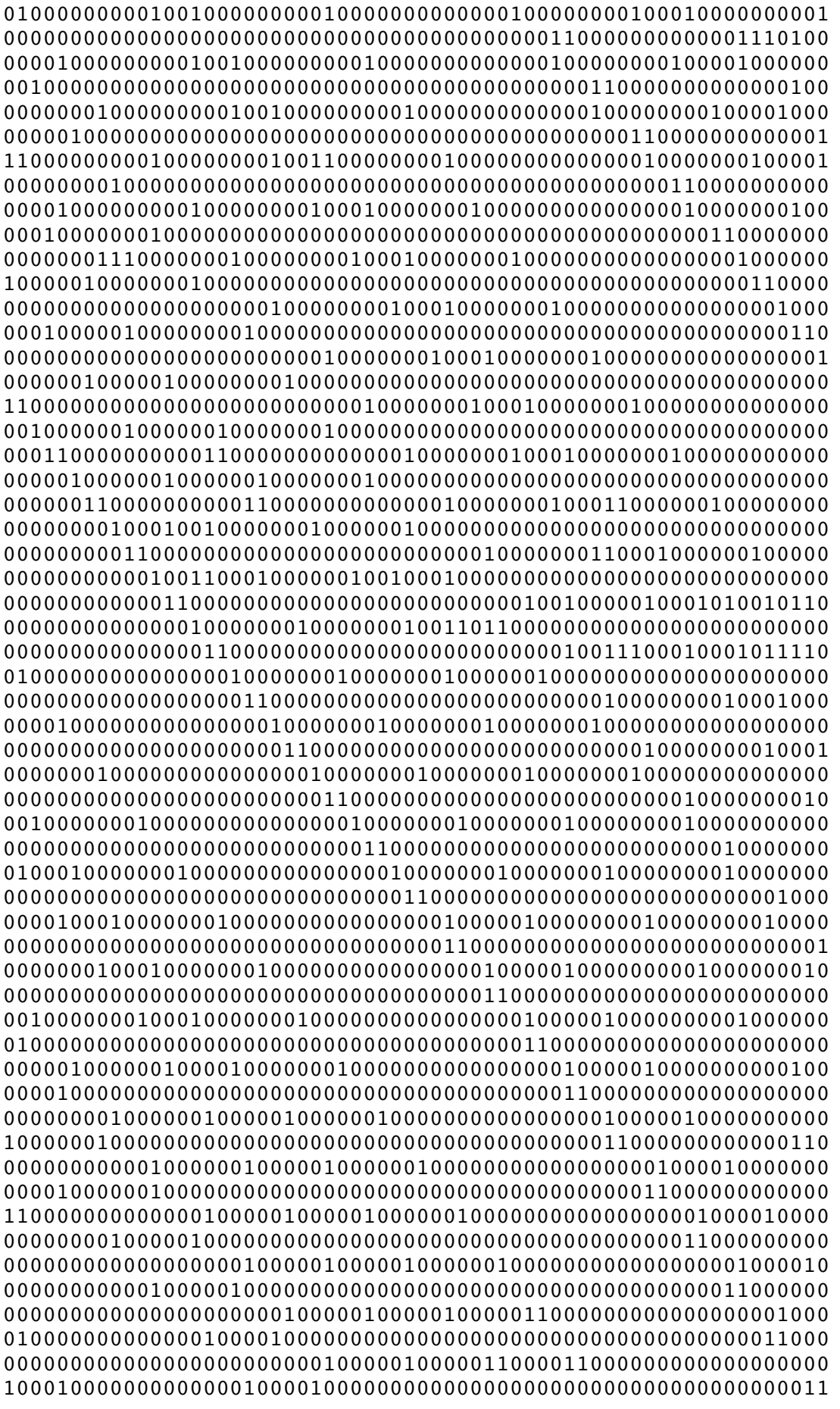


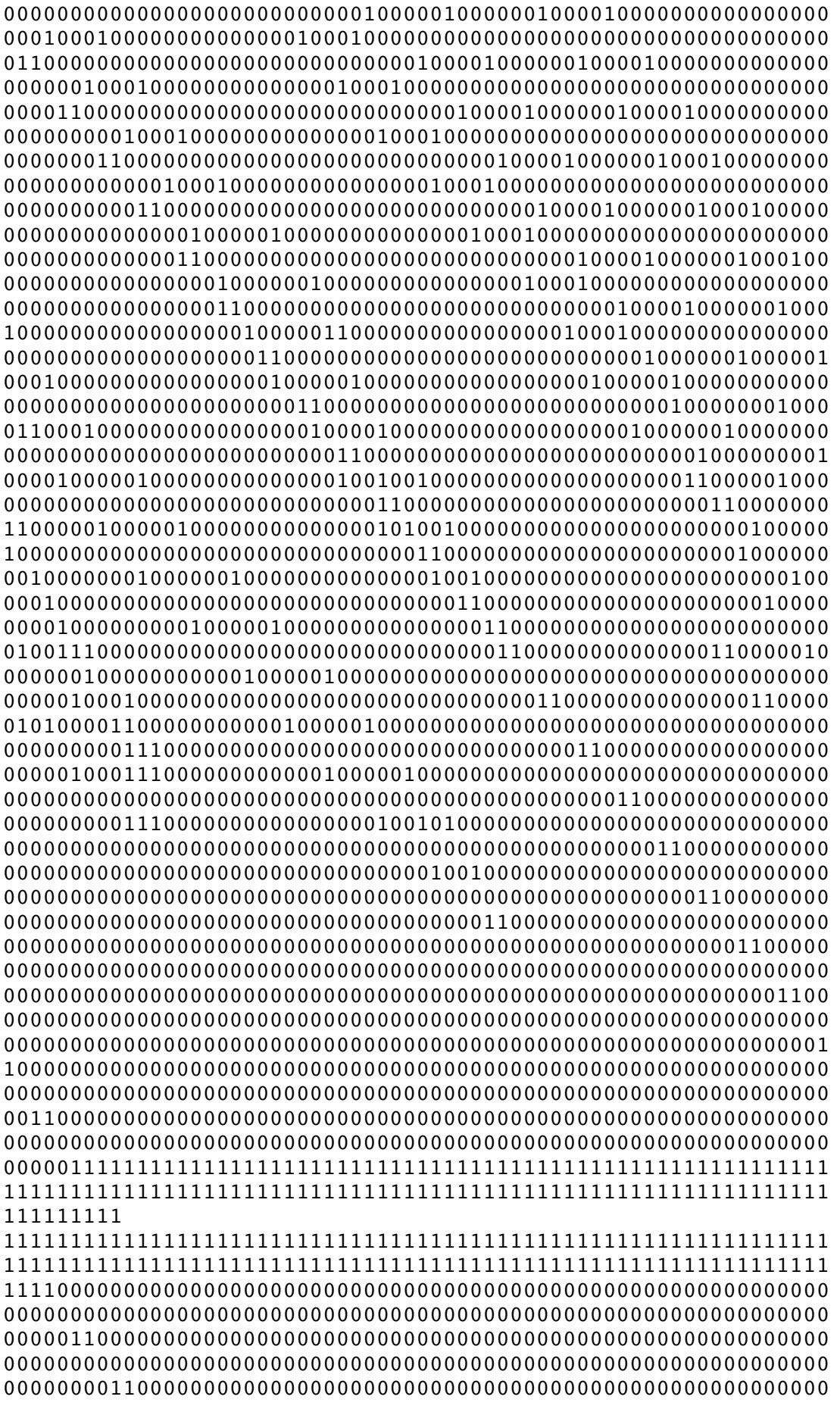


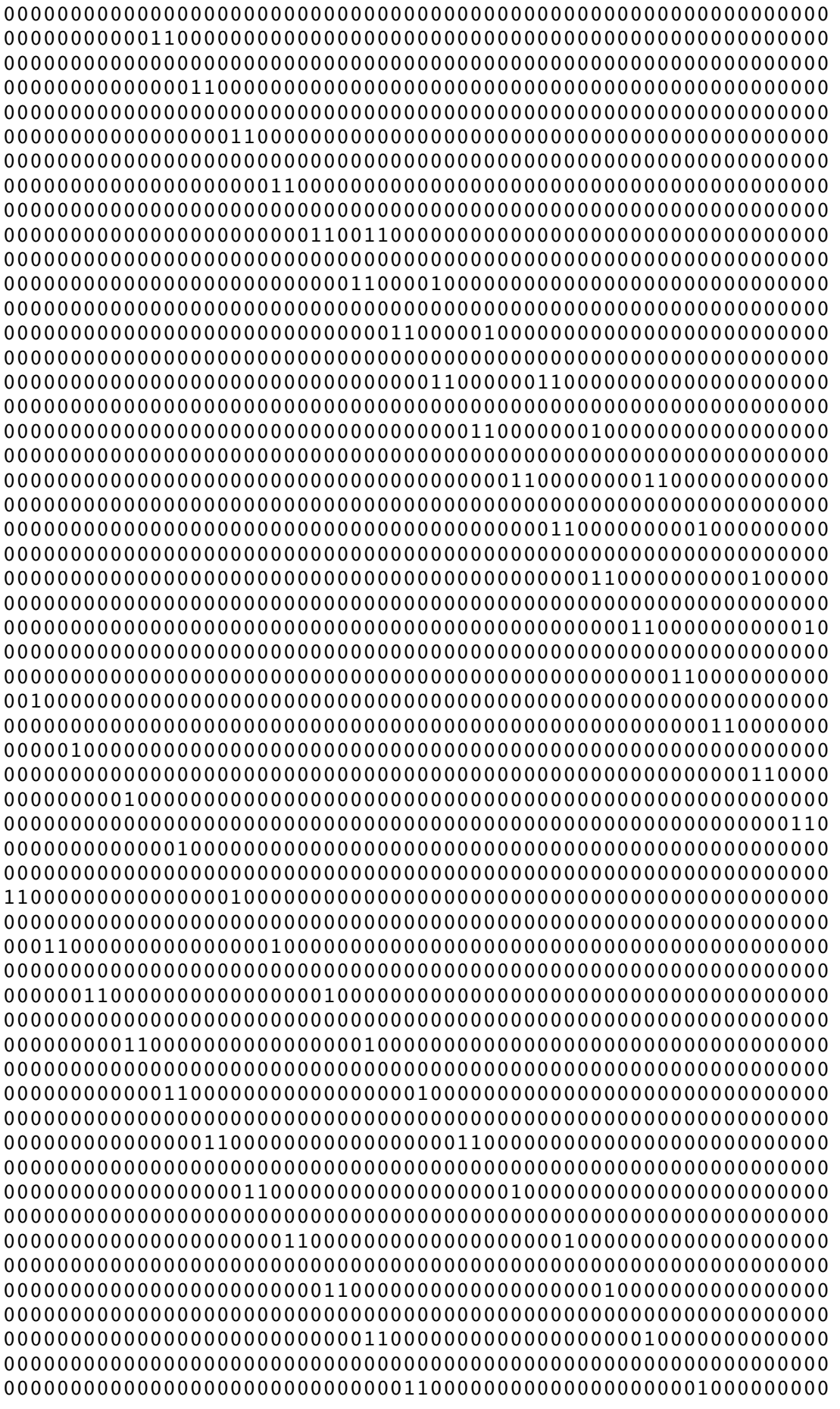


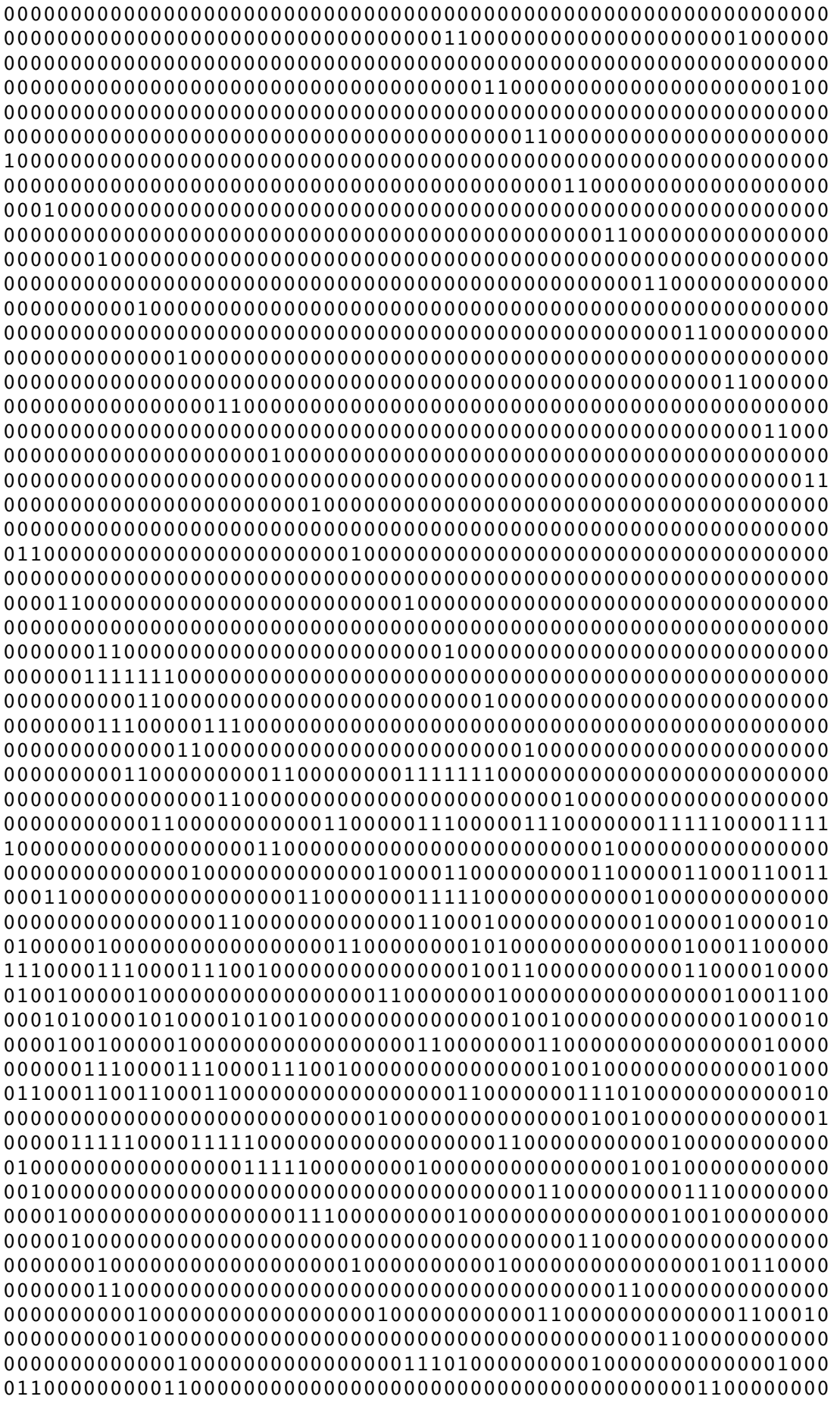


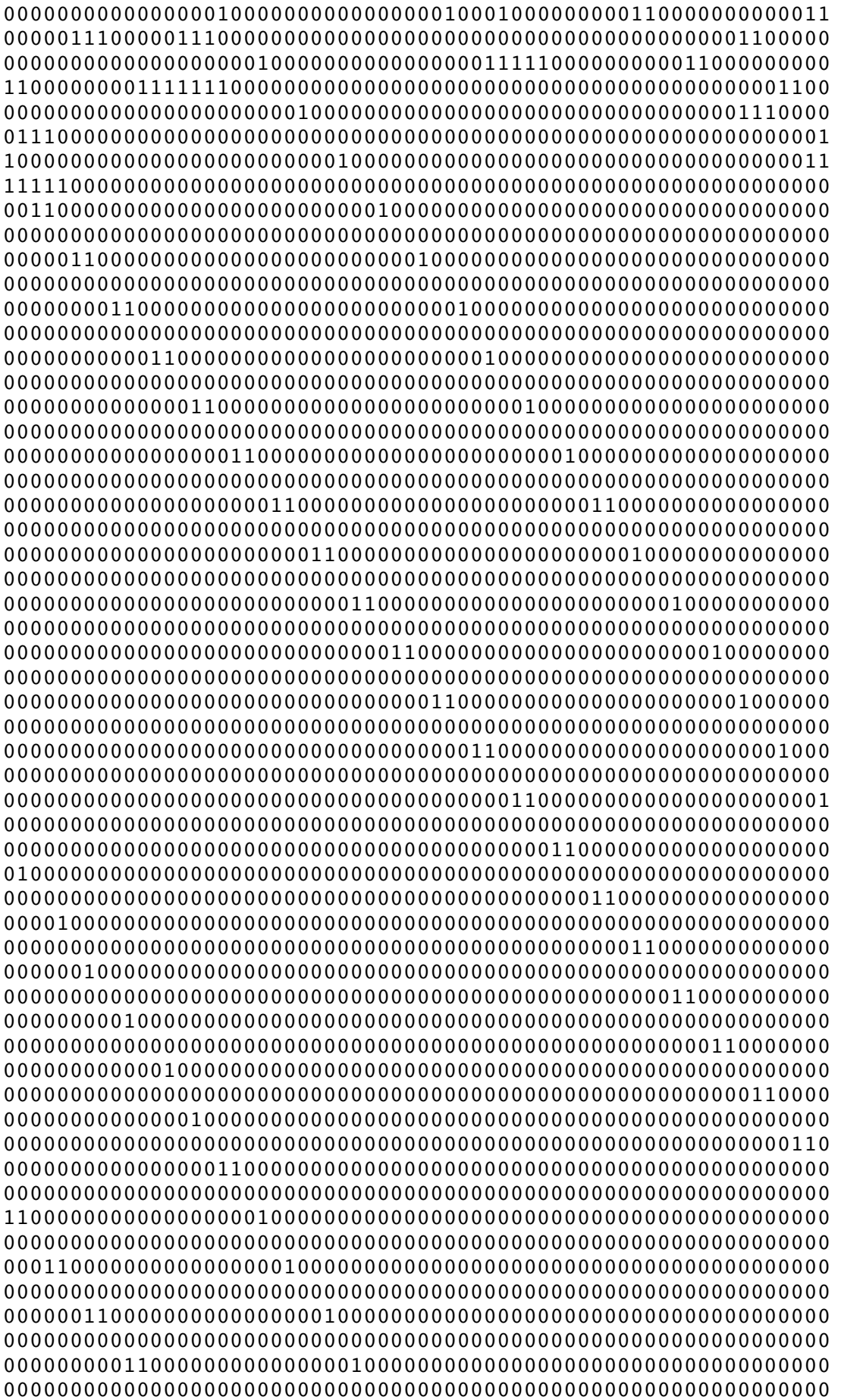


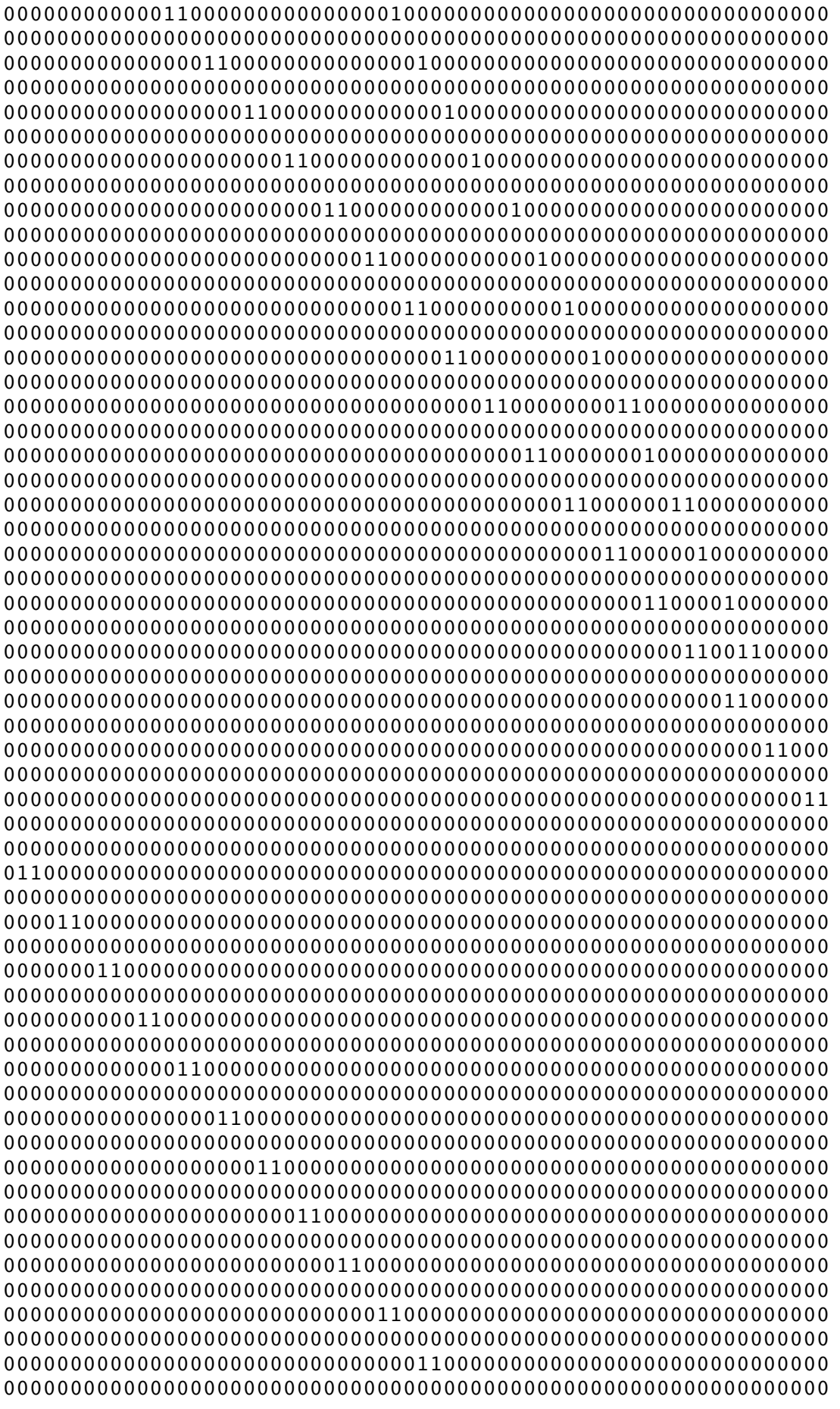


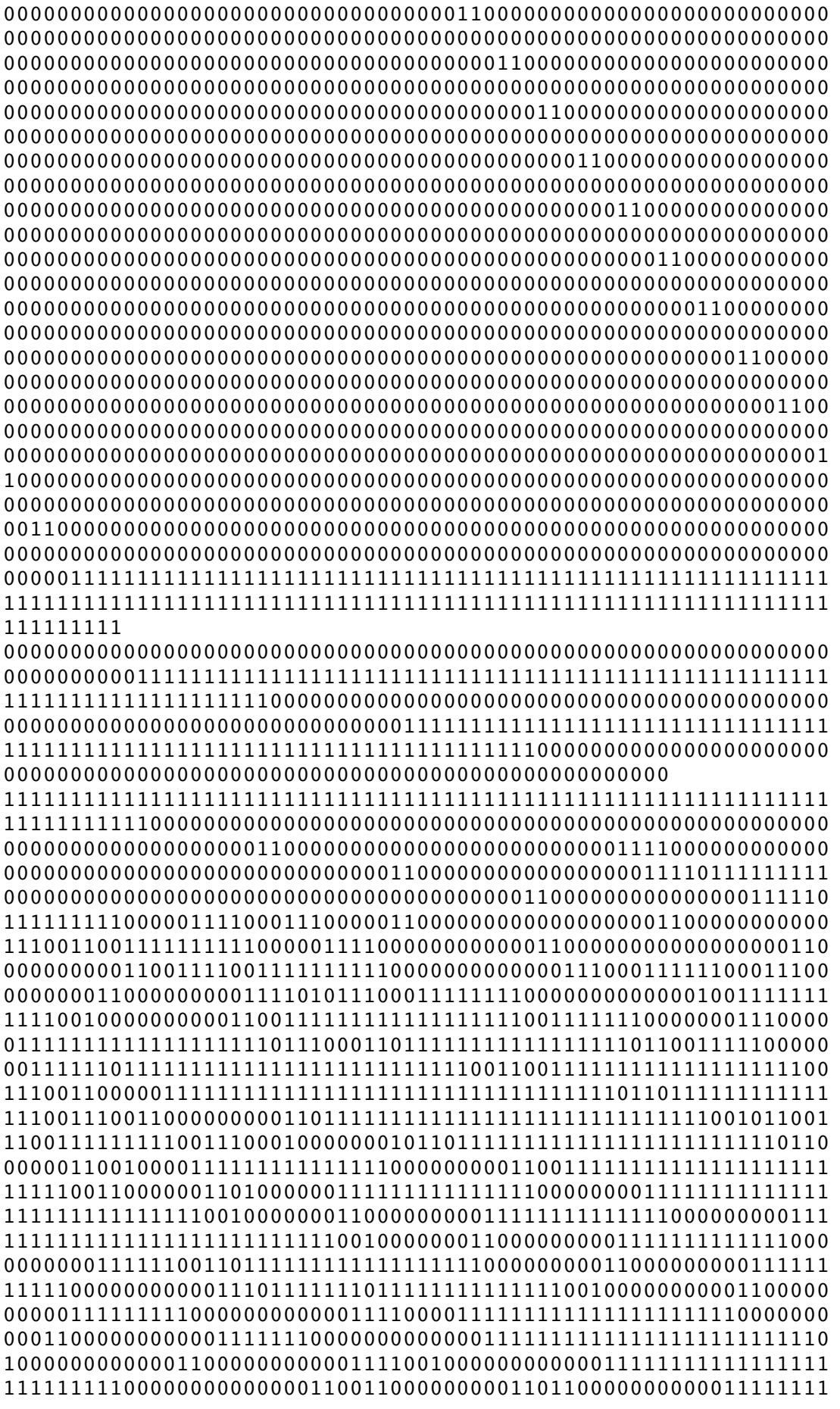


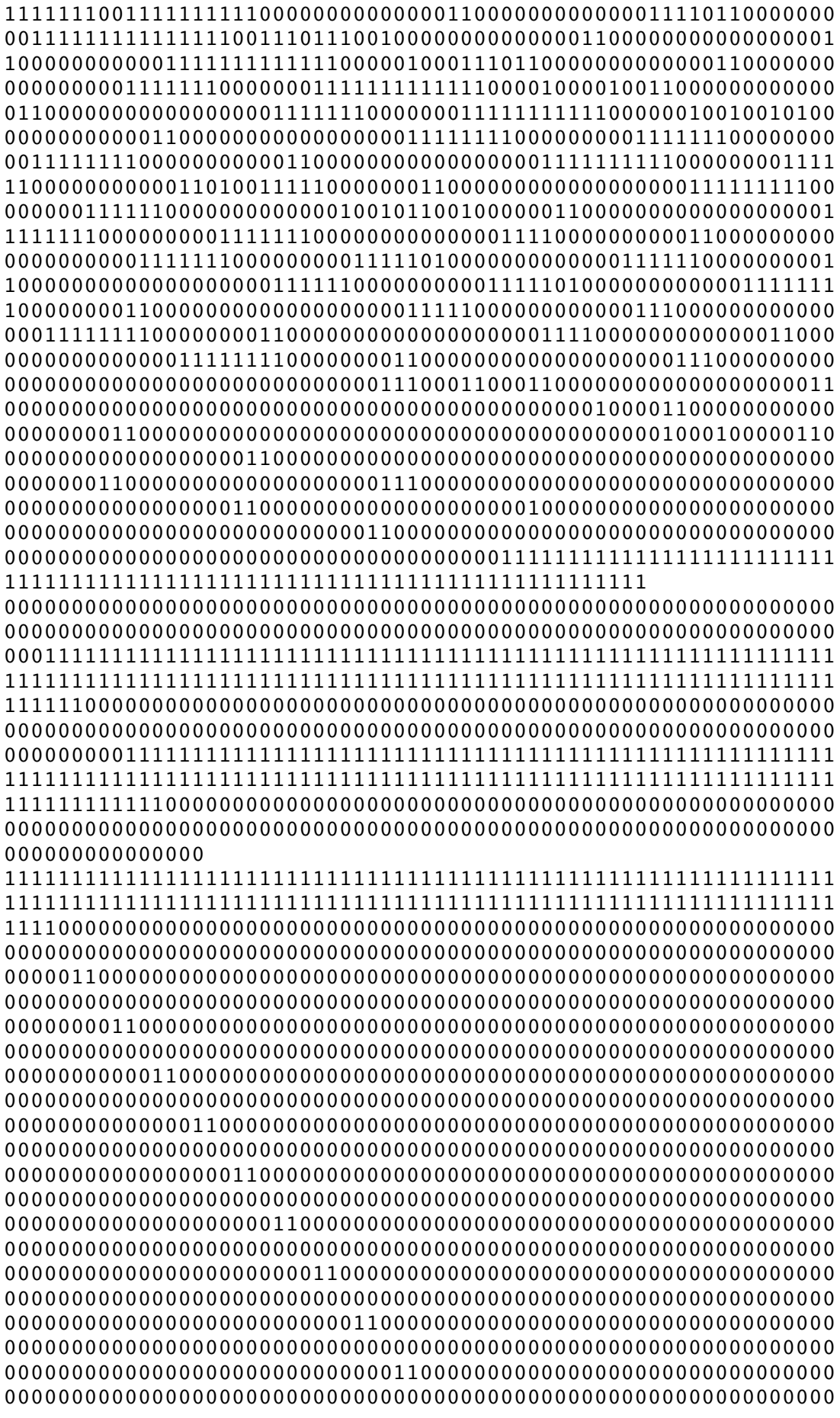


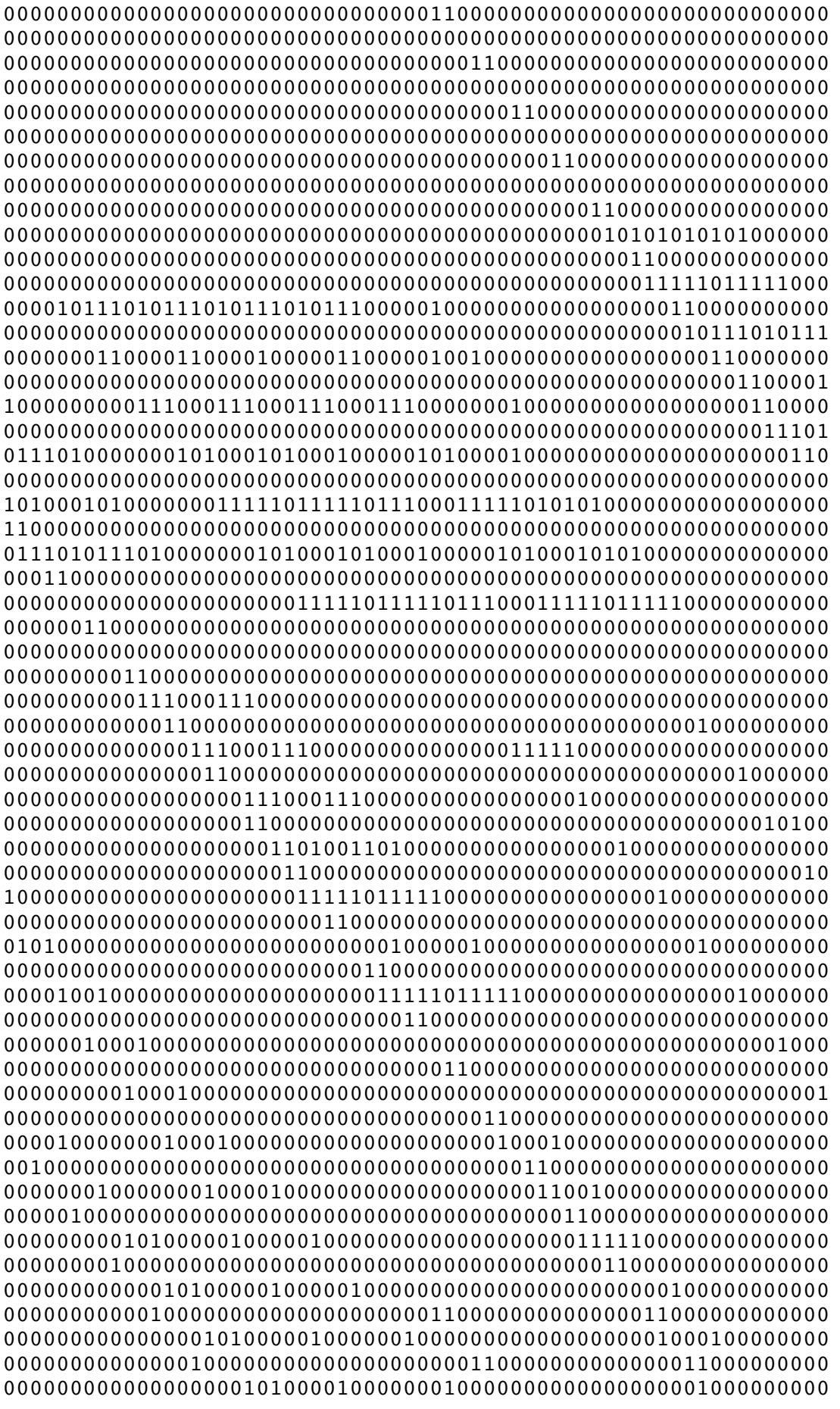


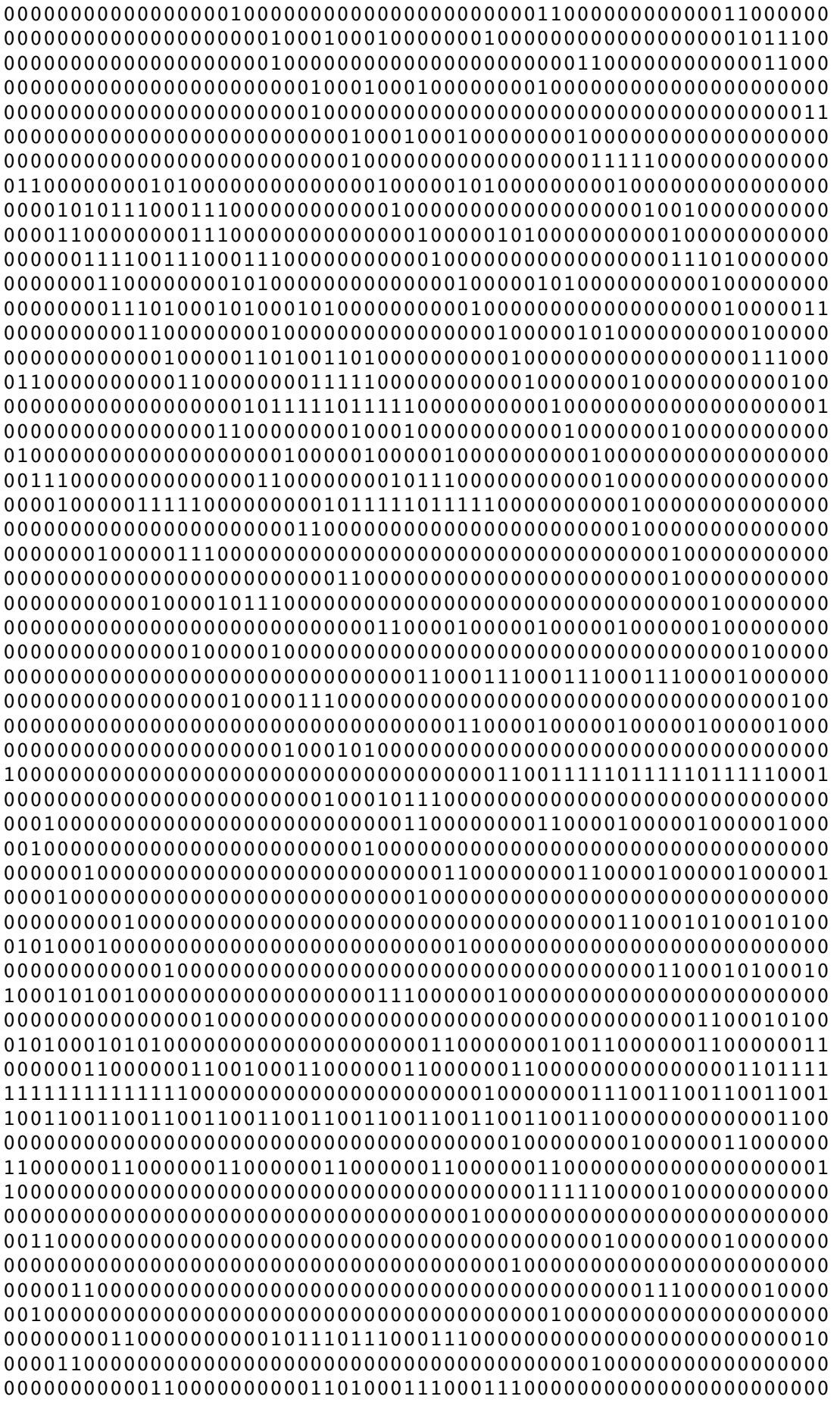


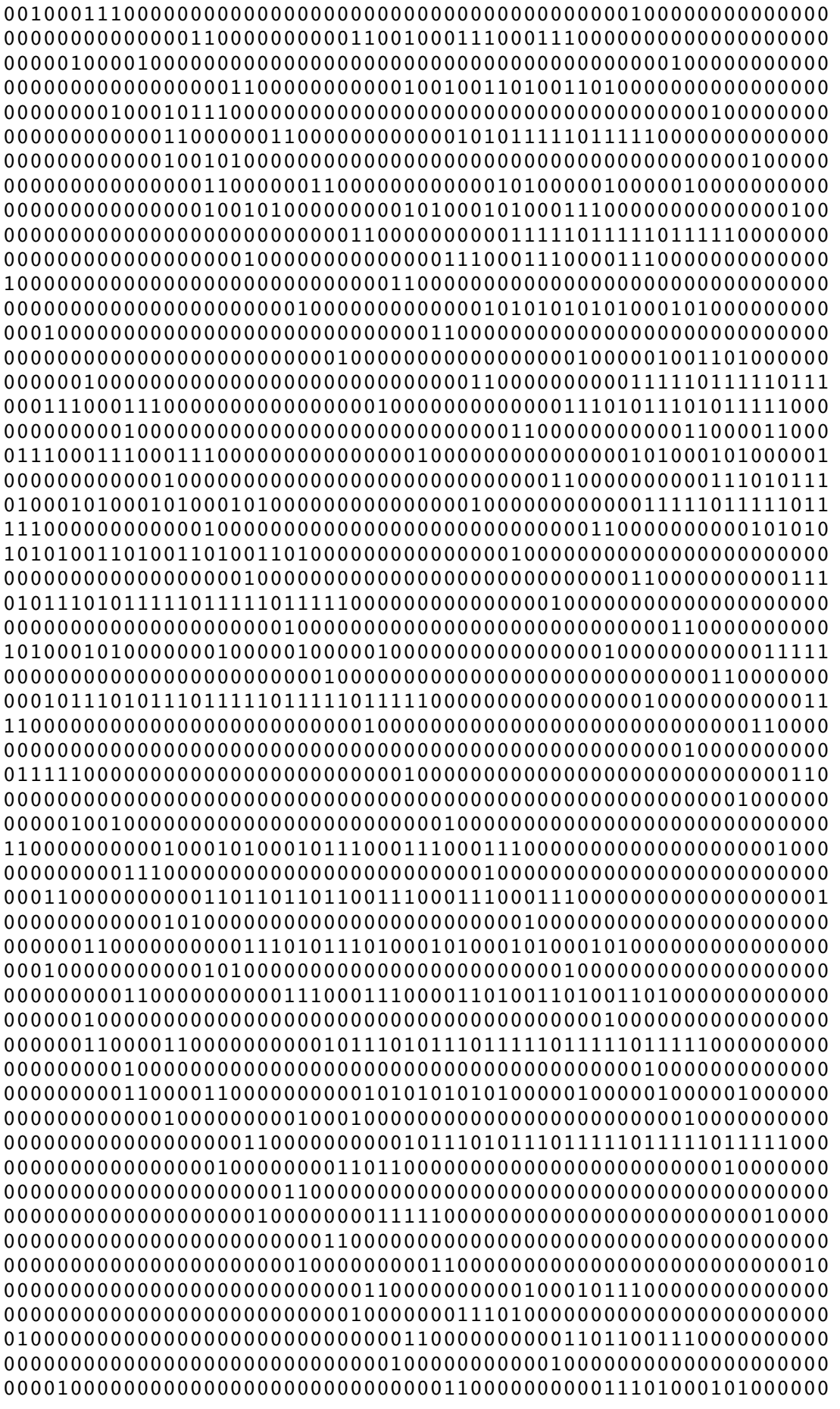


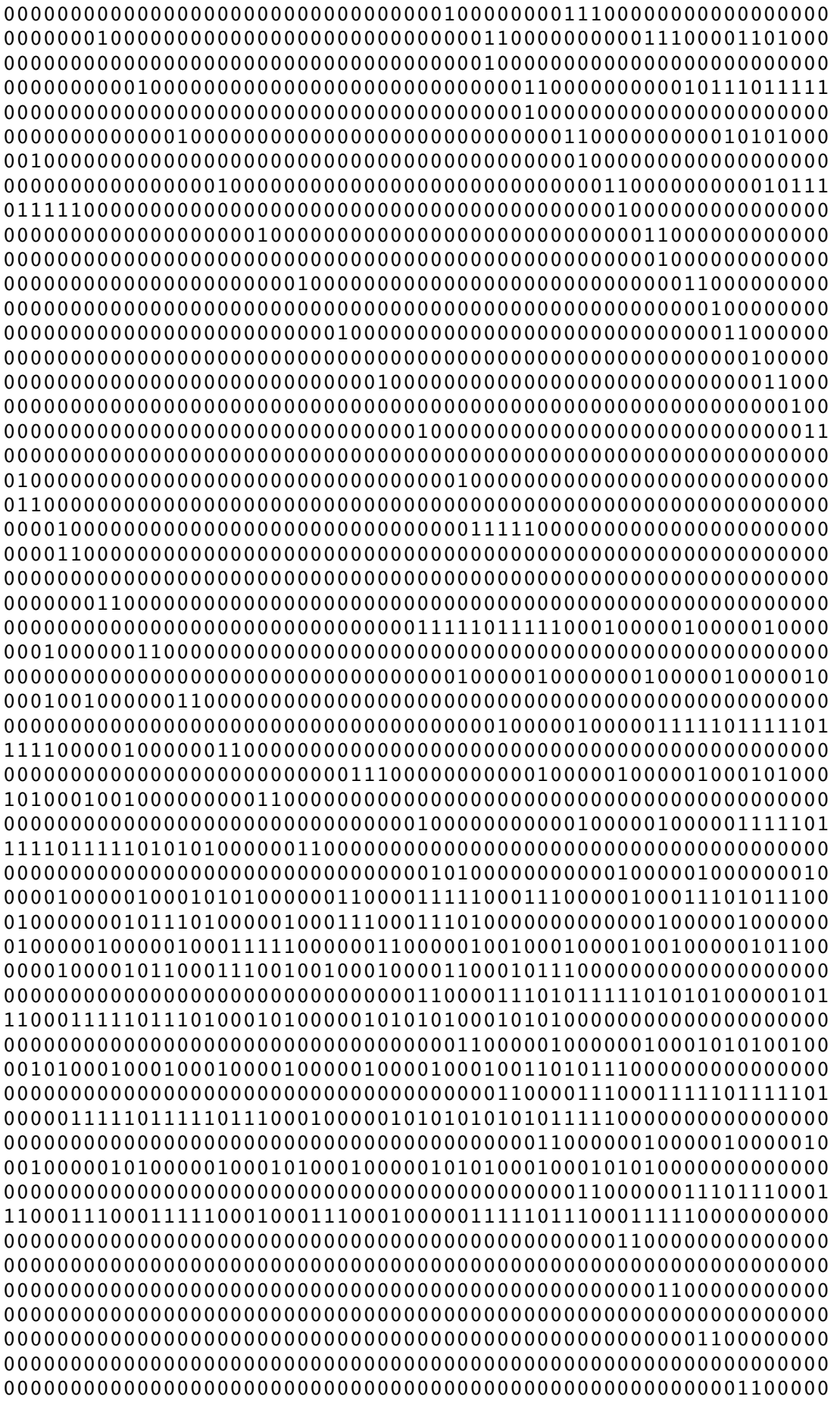


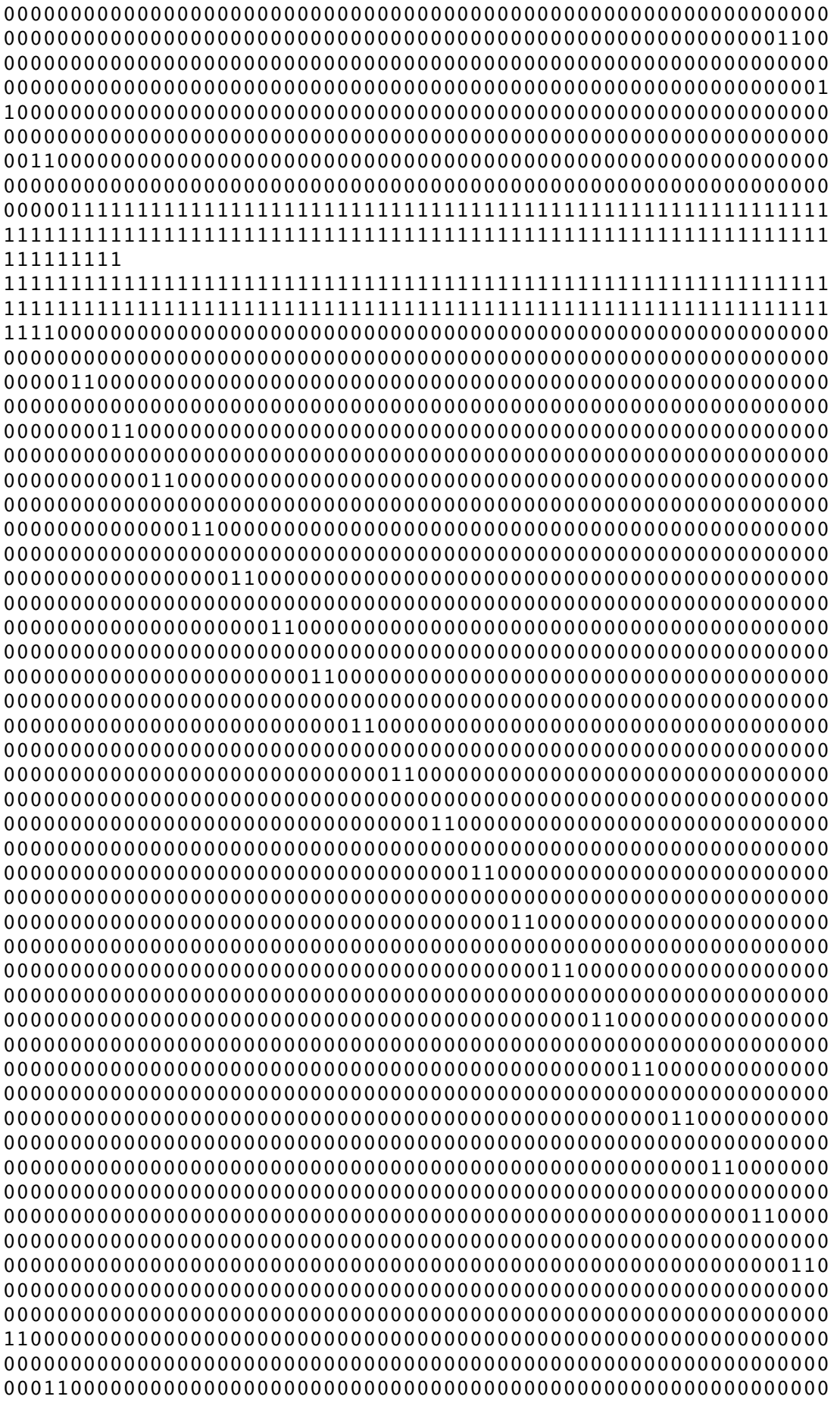


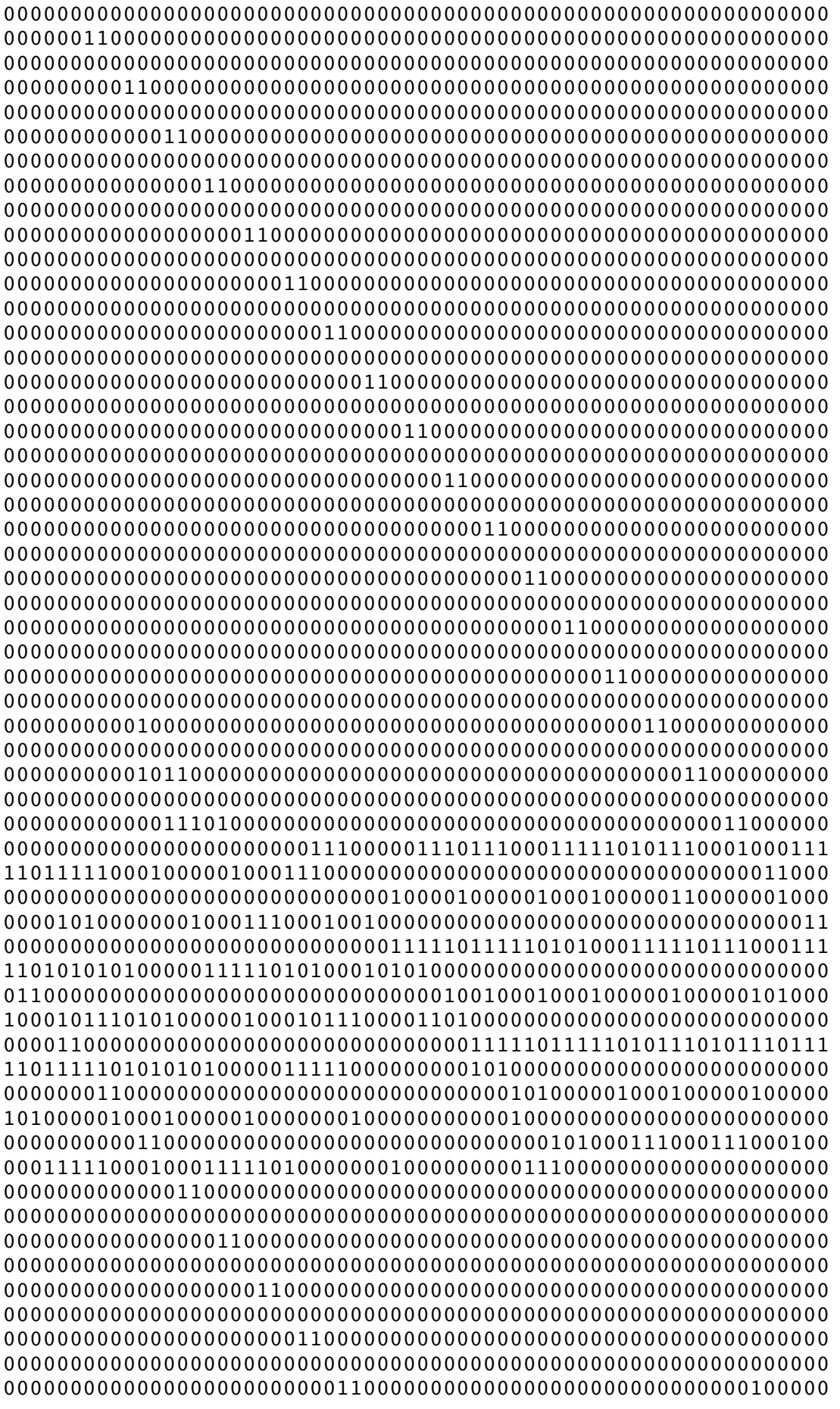


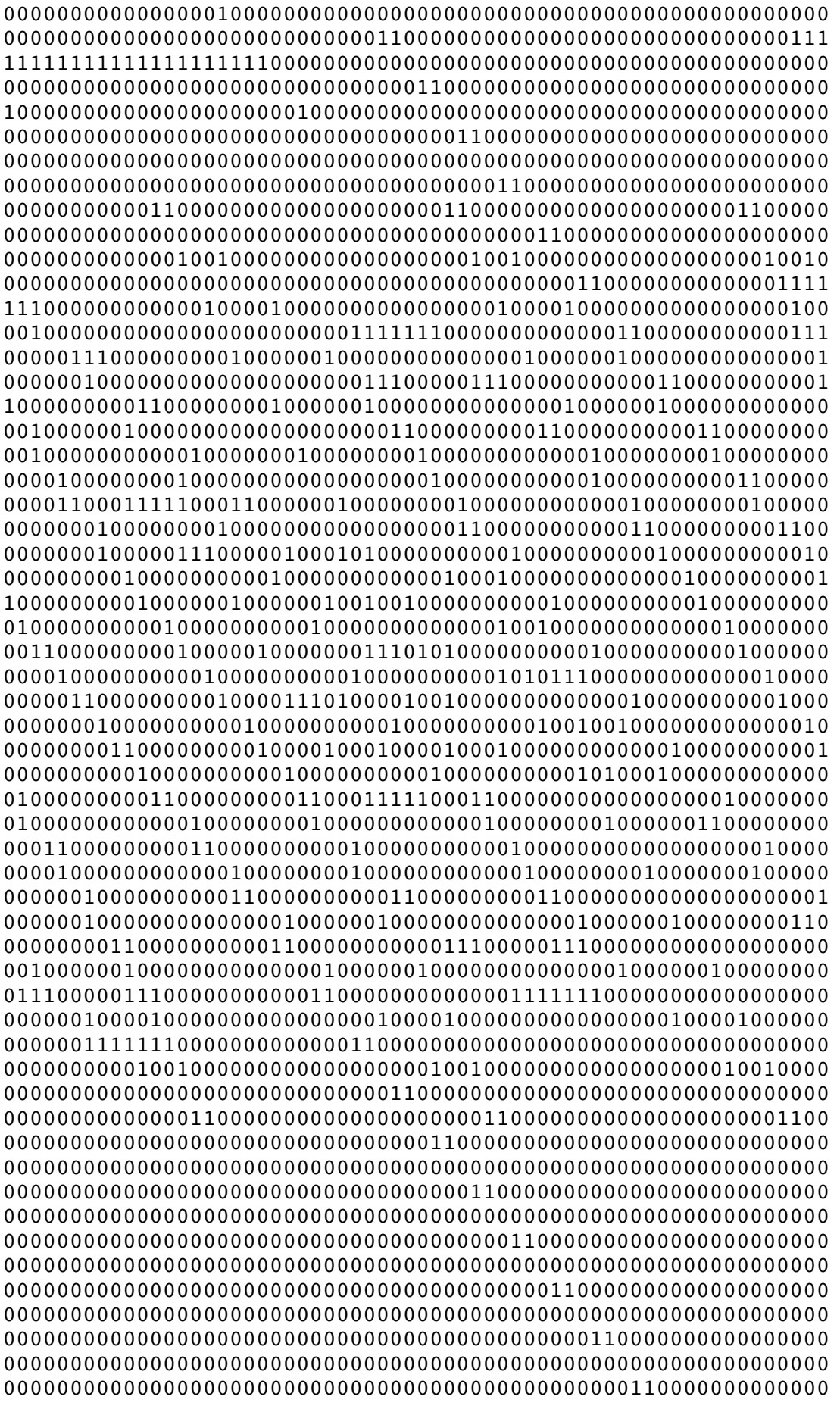


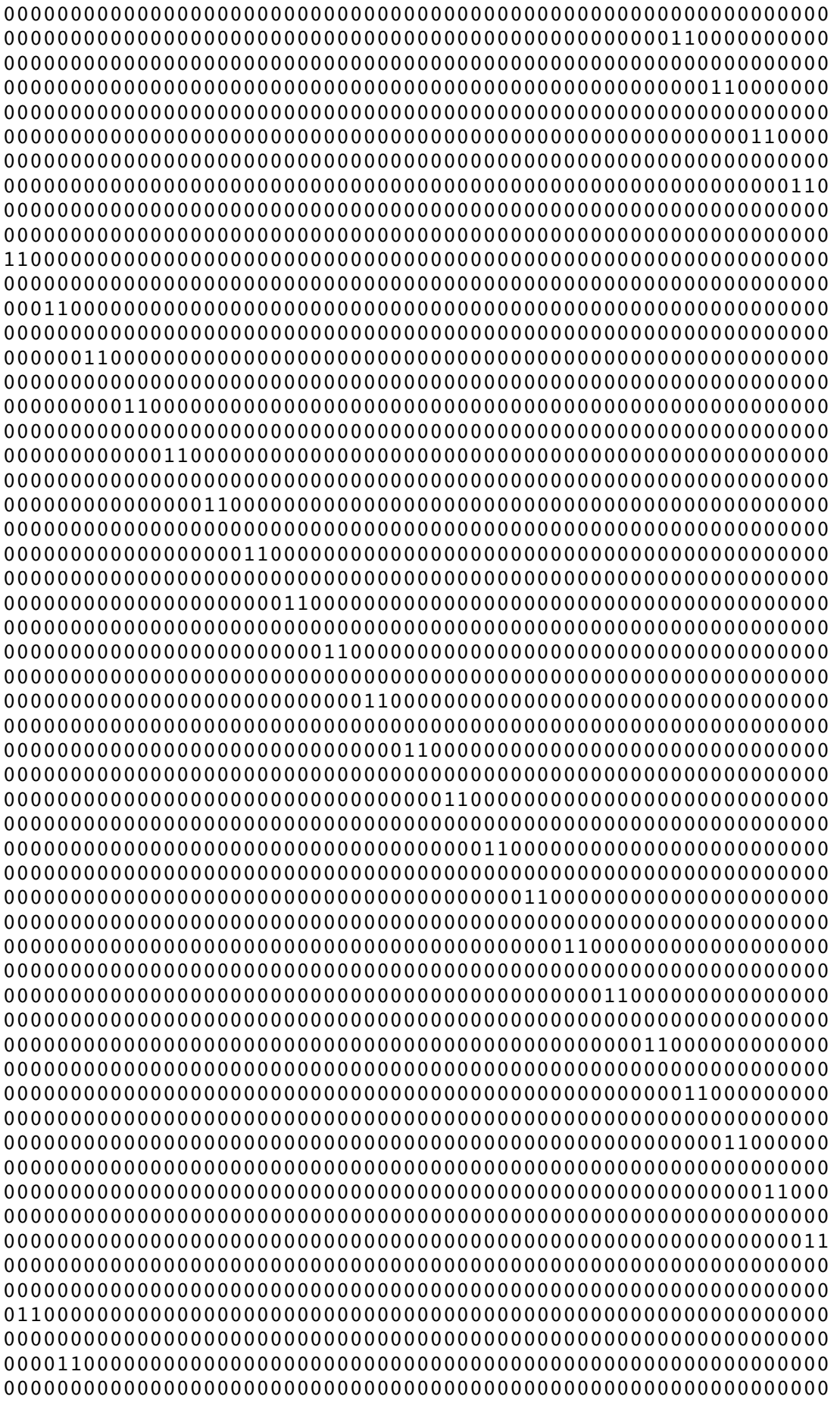


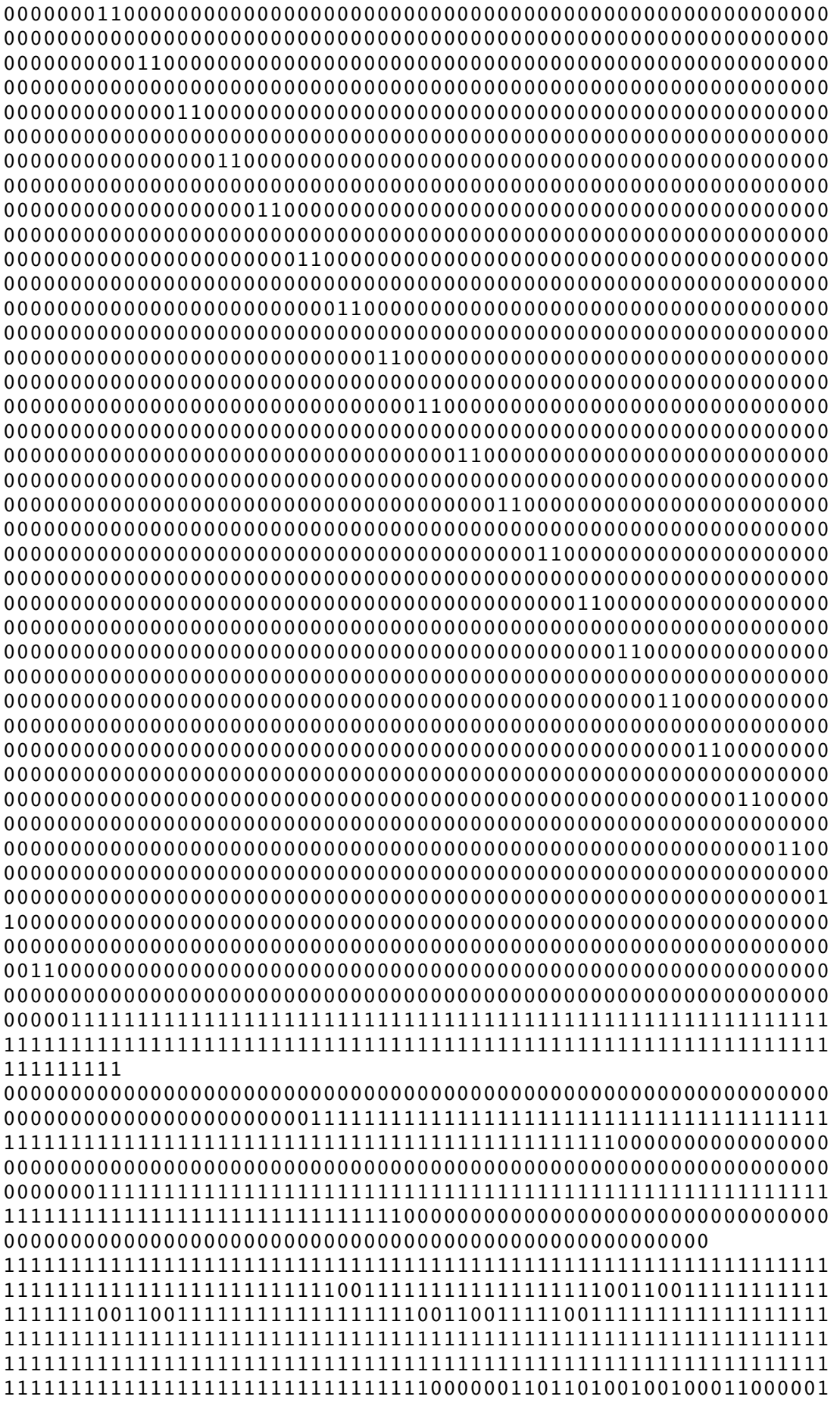


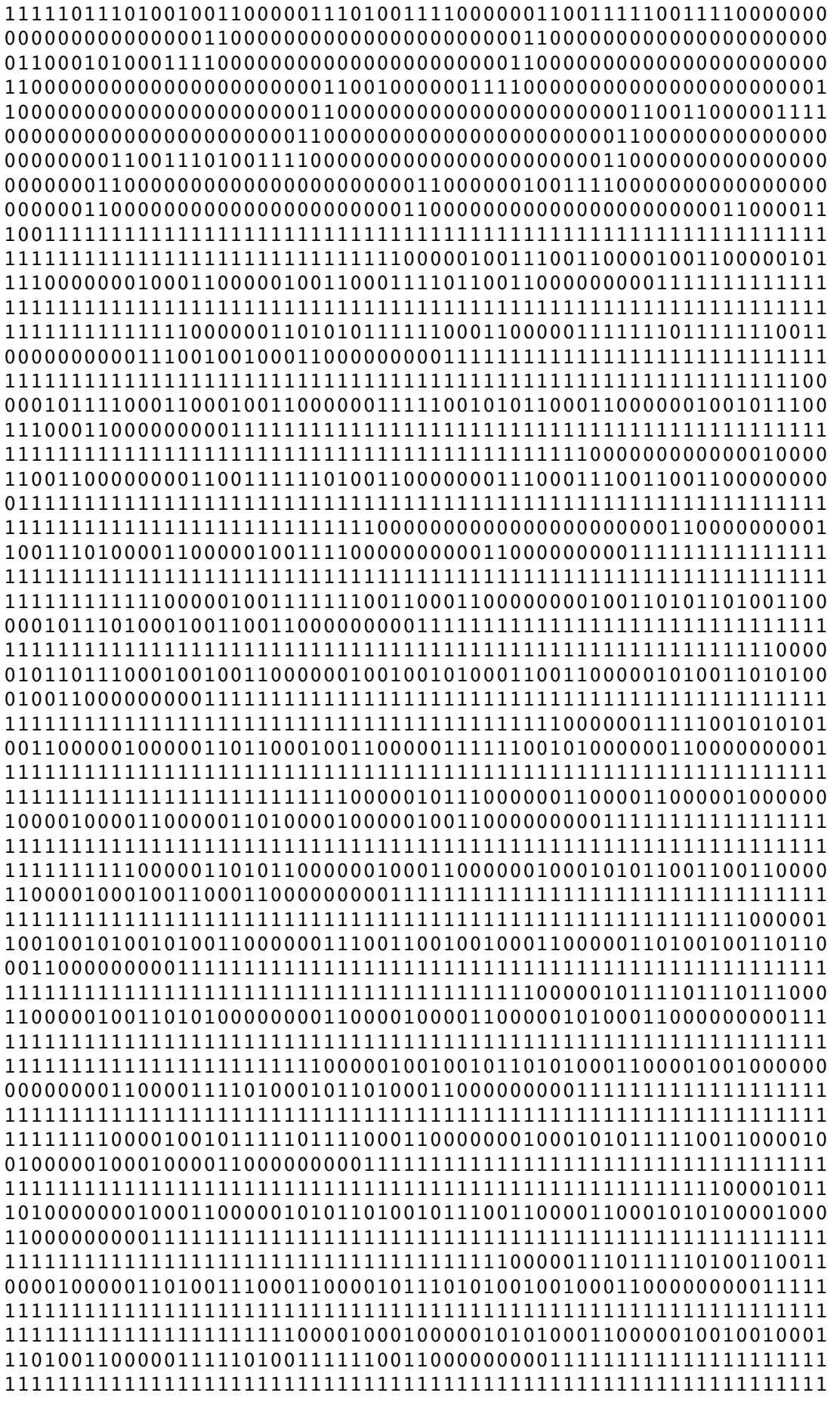


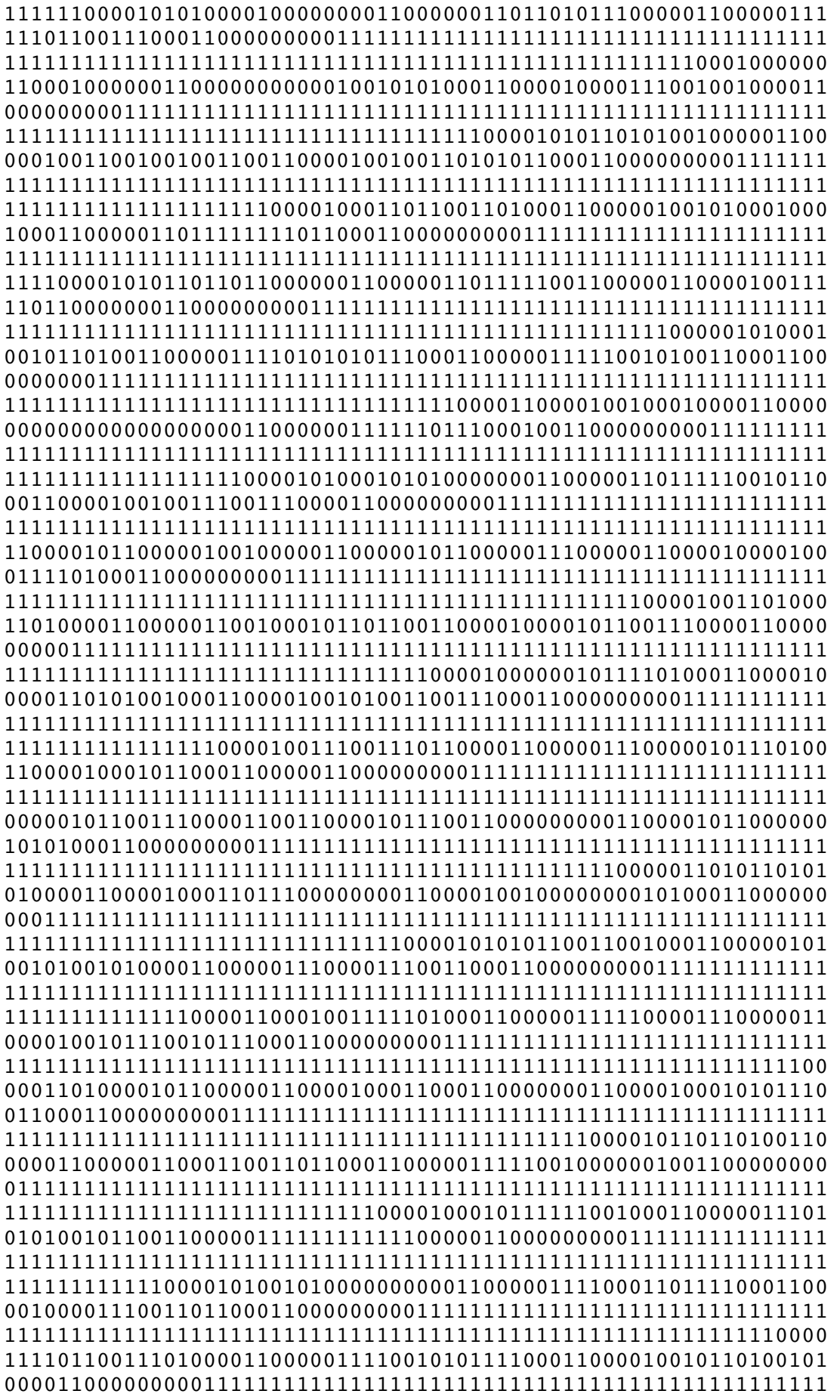


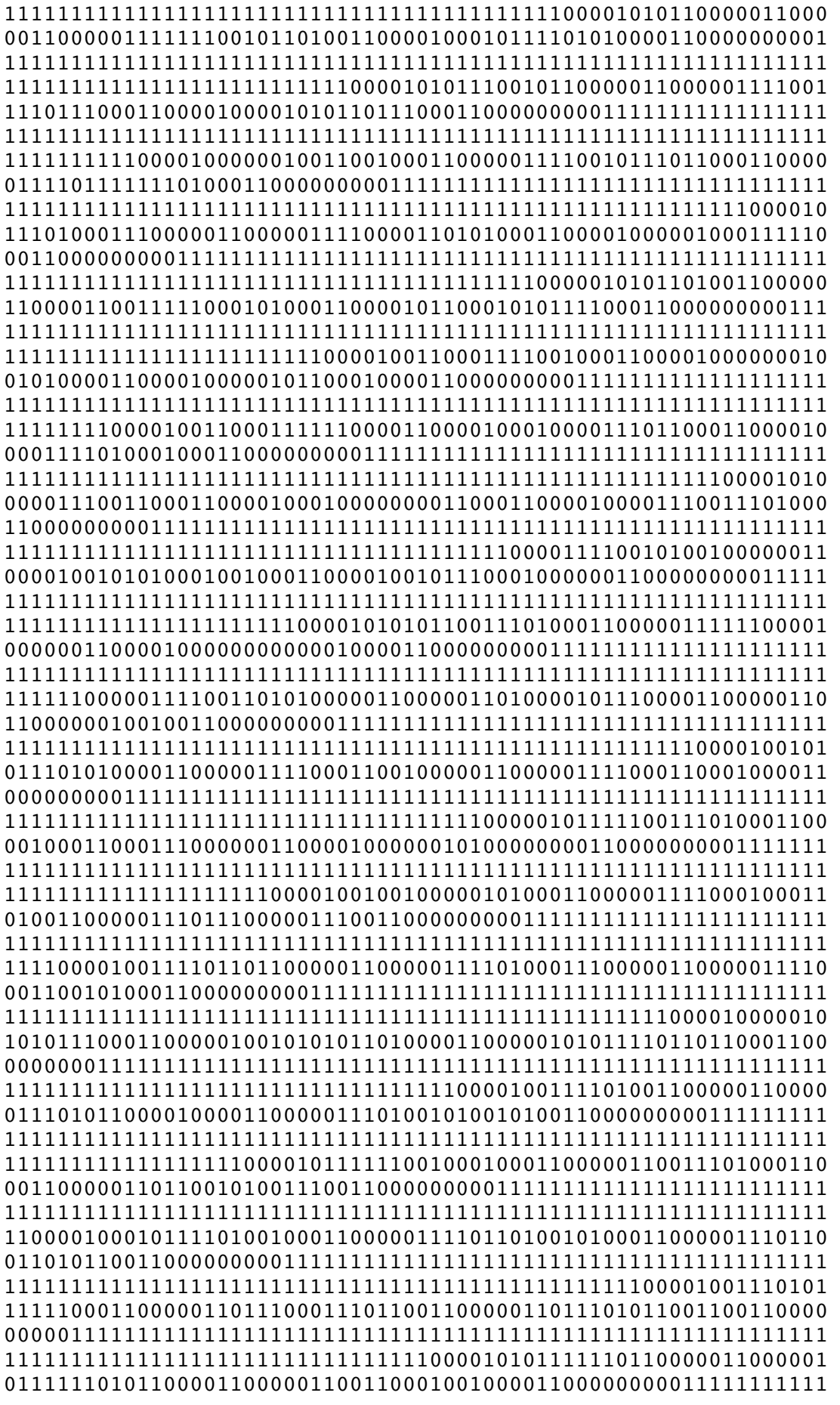


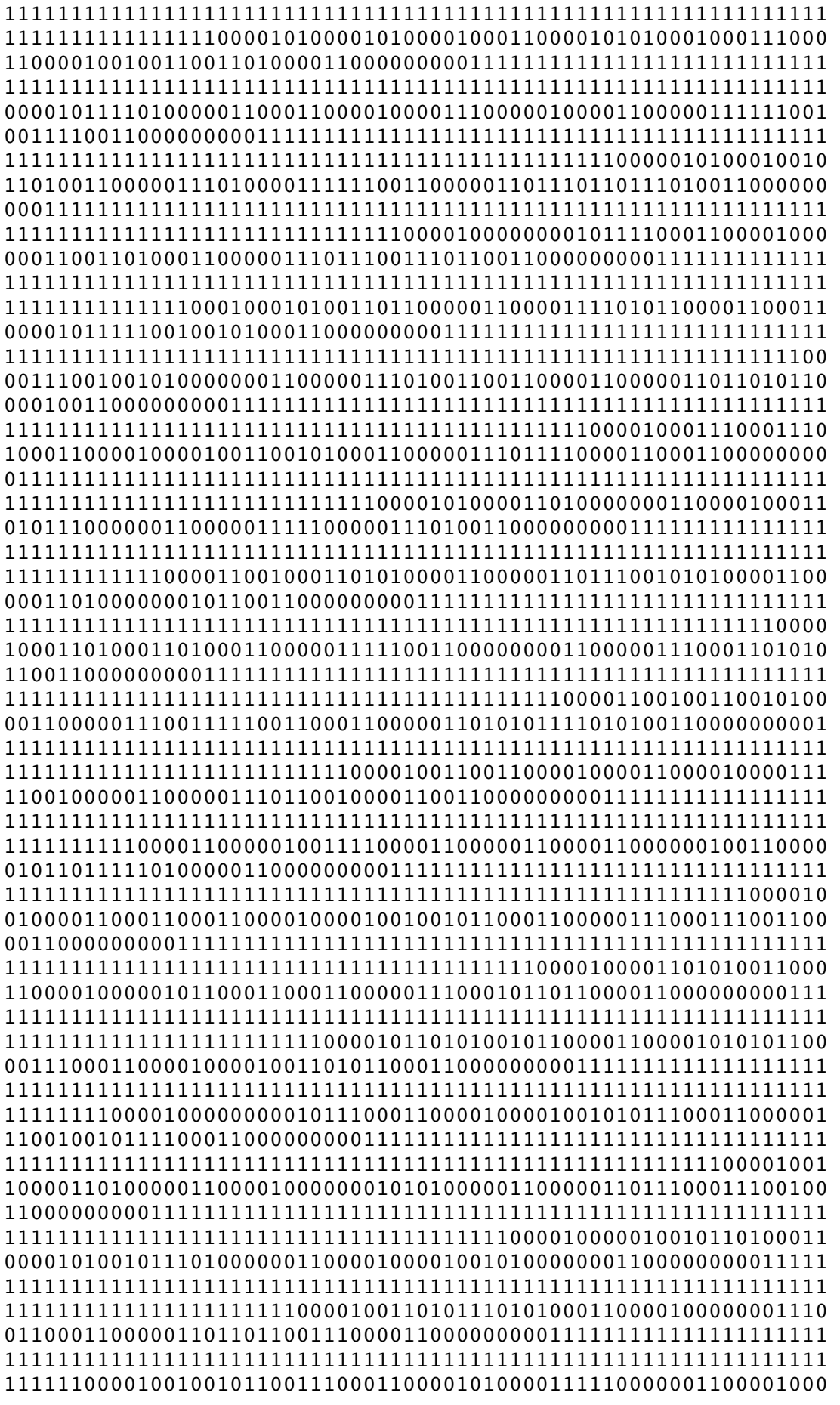


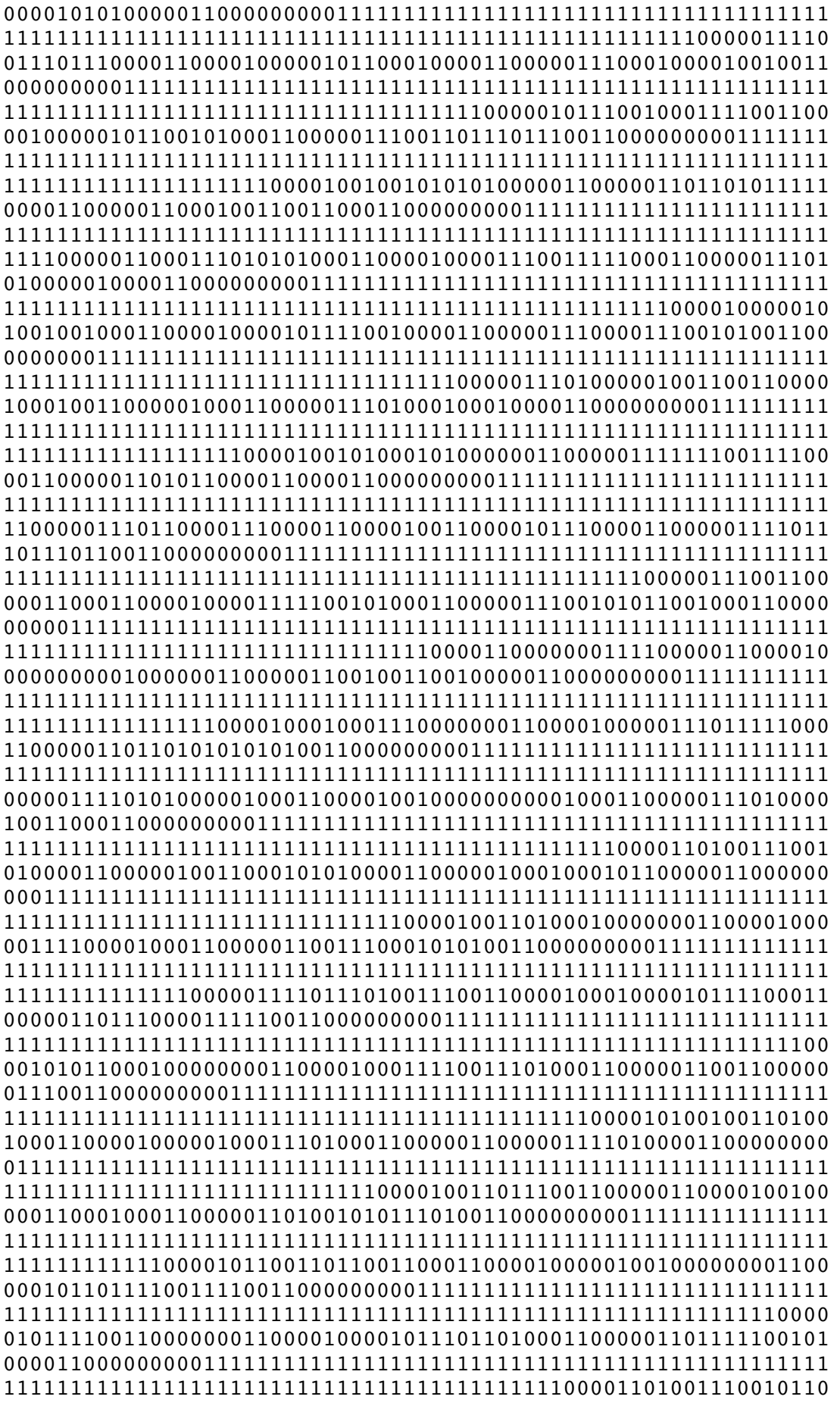


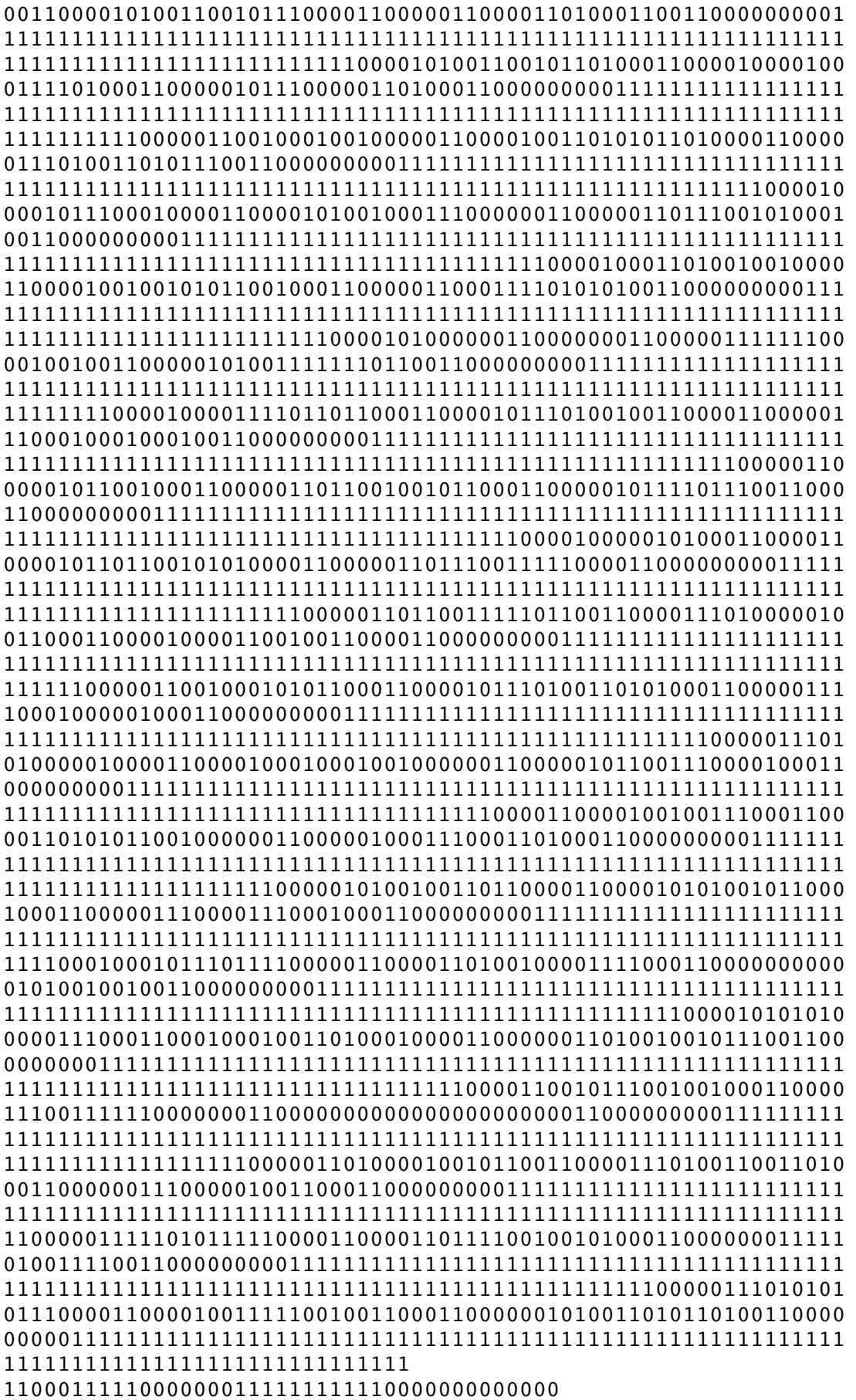












Acknowledgements

This work was developed in collaboration with the authors of *Jiang et al.* [2022]. We sincerely acknowledge Yvan Dutil and Stéphane Dumas, who pioneered Evpatoria Transmission Messages (ETMs) in 1999 and 2003. Part of our BITG Message were designed based on the ETMs concept. We also thank Michael Chorost for archiving the ETMs on Smithsonian's public website (https://www.smithsonianmag.com/science-nature/how-couple-guys-built-most-ambitious-alien-outreach-project-ever-180960473/). This work was supported by the Jet Propulsion Laboratory, California Institute of Technology, under contract with NASA. We also thank the supports from the FAST and SETI programs.

Reference

Jiang, J.H.; Li, H.; Chong, M.; Jin, Q.; Rosen, P.E.; Jiang, X.; Fahy, K.A.; Taylor, S.F.; Kong, Z.; Hah, J.; Zhu, Z.-H. A Beacon in the Galaxy: Updated Arecibo Message for Potential FAST and SETI Projects *Galaxies*, 10, 55, 2022.