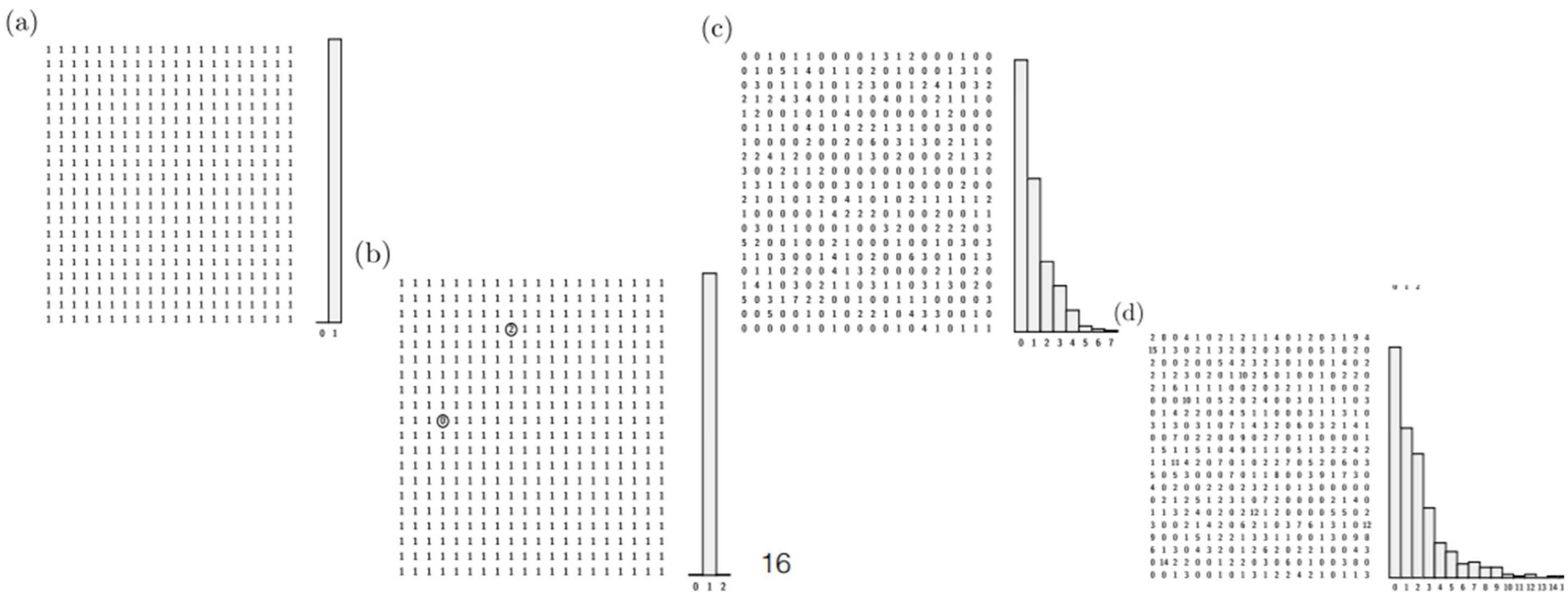
Exercise to see the Boltzmann distribution

- To illustrate the statistical nature of the Boltzmann distribution, let us do some coding in which quanta of energy are distributed in a lattice.
- Choose a lattice of 400 sites, arranged for convenience on a 20×20 grid.
- Each site initially contains a single energy quantum, as shown in (a).
- Now choose a site at random and remove the quantum from that site and place it on a second, randomly chosen site. The resulting distribution will be as in (b),
- Repeat the redistribution process many times and the resulting distribution will be as in (c). The histogram describing this looks very much like a Boltzmann exponential distribution.
- Role of temperature is played by the total number of energy quanta in play. If initial arrangement had been two quanta per site rather than one quantum per site, then after many iterations one would obtain the arrangement shown in (d).



P Y T H O N

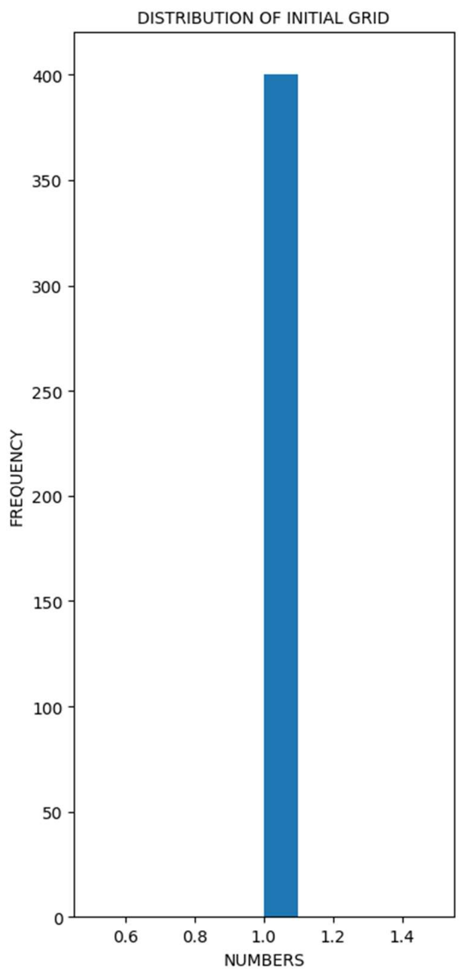
P R O G R A M M E

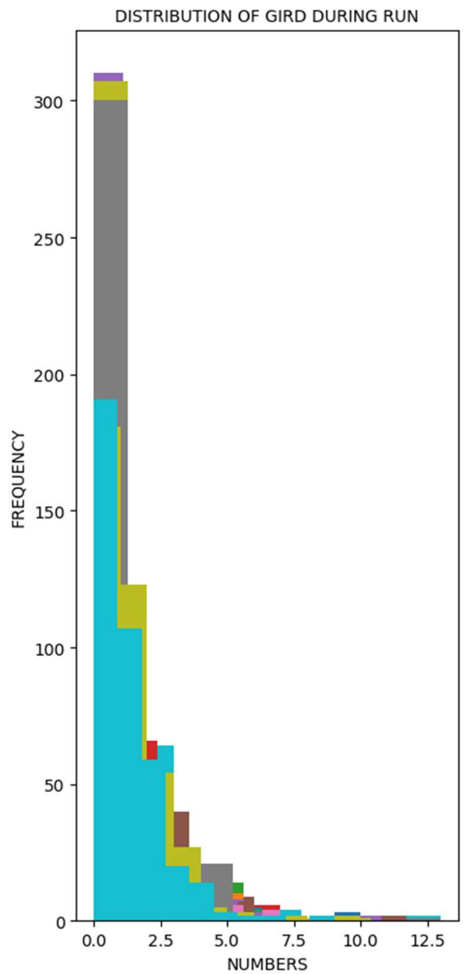
```python
import numpy as np

import matplotlib.pyplot as plt

print("\nEnter the size of array:")
n = int(input())
shape = (n, n)

# Initial quantum.
print("\nEnter the initial energy:")
ini_ene = int(input())

# Max number of iterations.
print("\nEnter the max iteration:")
imax = int(input())

# Intial configuration.
ini = np.full( shape, ini_ene)

# For updating the grid.
def update(ini):

  ini_new = np.copy(ini)
  i = np.random.randint(ini.shape[0])
  j = np.random.randint(ini.shape[1])

  if ini_new[i, j] > 0:
    ini_new[i, j] += -1
    m = np.random.randint(ini.shape[0])
    n = np.random.randint(ini.shape[1])
    ini_new[m, n] += 1

  return ini_new

f = plt.figure()
plt.ion()
```

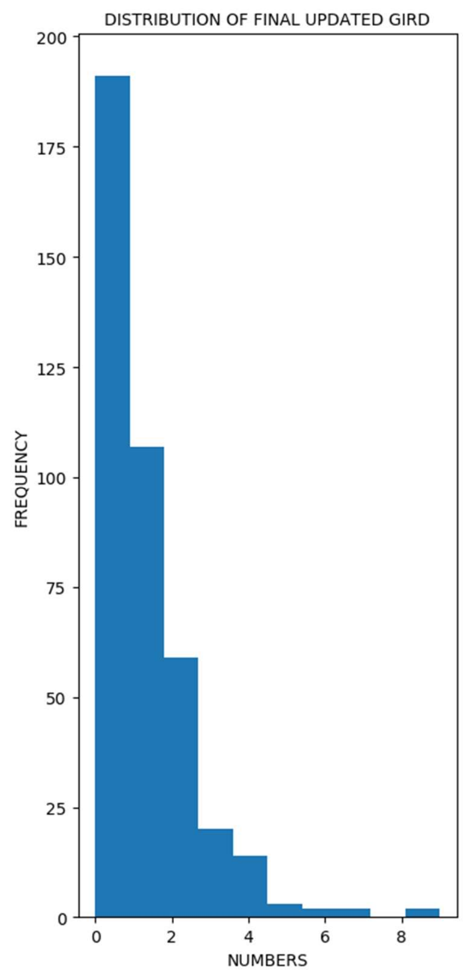
```python
# Plot for initial distribution.

ini_img=ini.flatten()
f.add_subplot(1,4, 2)
_ = plt.hist(ini_img)
_ = plt.title('DISTRIBUTION OF INITIAL GRID', fontsize=10)
_ = plt.xlabel('NUMBERS')
_ = plt.ylabel('FREQUENCY')

f.add_subplot(1,4, 1)
im = plt.imshow(ini ,cmap='gist_earth', vmin=0, vmax=400, interpolation='non
e')
a = plt.title('VISIUAL REPRESENTAION OF GRID', fontsize=10)
t = 1
while t<=imax:

  if t % 10 == 0:
    im.set_data(ini)
    plt.draw()
  ini = update(ini)

  if t%1000 == 0:
    m=ini.flatten()
    f.add_subplot(1,4, 3)
    a = plt.hist(m)
    a = plt.title('DISTRIBUTION OF GIRD DURING RUN', fontsize=10)
    a = plt.xlabel('NUMBERS')
    a = plt.ylabel('FREQUENCY')
  if t % imax == 0:
    m=ini.flatten()
    f.add_subplot(1,4, 4)
    b = plt.hist(m)
    b = plt.title('DISTRIBUTION OF FINAL UPDATED GIRD', fontsize=10)
    b = plt.xlabel('NUMBERS')
    b = plt.ylabel('FREQUENCY')
  plt.pause(.001)
  t += 1
plt.show(block=True)
```

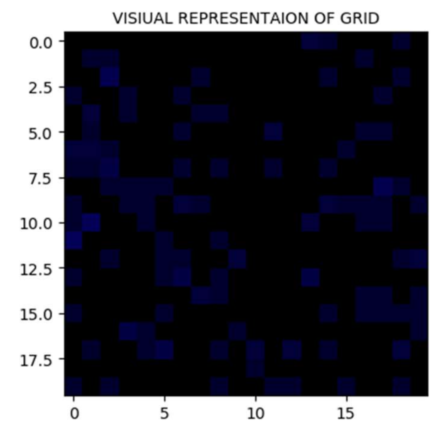
The programme was executed for the grid of shape 20x20 with the initial quantum of 1 for each site and for 50000 iterations.

```
Enter the size of array:
20

Enter the initial energy:
1

Enter the max iteration:
50000
```









NOTE: Here frequency is the occurrence of each digit in the grid