In [3]:

```python
import numpy as np
import matplotlib.pyplot as plt


betas = [0.5, 0.6, 0.7, 0.8, 0.83, 0.84, 0.85, 0.9]
# betas = [1]
size = 64


def dist(x, row, col, label, beta):
    energy = np.zeros(4)
    if row + 1 <= 63:
        energy[0] = int(x[row + 1, col] == label)
    if row - 1 >= 0:
        energy[1] = int(x[row - 1, col] == label)
    if col + 1 <= 63:
        energy[2] = int(x[row, col + 1] == label)
    if col - 1 >= 0:
        energy[3] = int(x[row, col - 1] == label)
    energy = np.sum(energy)
    return np.exp(beta * energy)

```

In [*]:

```
for beta in betas:
    n_sweeps = 0
    x_1 = np.ones((size, size), dtype=np.int16)
    x_2 = np.zeros((size, size), dtype=np.int16)
    sum_1_arr = np.sum(x_1)
    sum_2_arr = np.sum(x_2)

    while True:
        row_i = np.random.permutation(size)
        col_i = np.random.permutation(size)
        for row in row_i:
            for col in col_i:
                denom = dist(x_1, row, col, 1, beta) + dist(x_1, row, col, 0, beta)
                prob_x1 = dist(x_1, row, col, 1, beta) / denom
                denom_2 = dist(x_2, row, col, 1, beta) + dist(x_2, row, col, 0, beta)
                prob_x2 = dist(x_2, row, col, 1, beta) / denom_2

                rand = np.random.uniform()
                x_1[row, col] = 1 if prob_x1 > rand else 0
                x_2[row, col] = 1 if prob_x2 > rand else 0

        sum_1 = np.sum(x_1)
        sum_2 = np.sum(x_2)
        sum_1_arr = np.append(sum_1_arr, sum_1)
        sum_2_arr = np.append(sum_2_arr, sum_2)
        n_sweeps += 1
        if sum_1 == sum_2:
            break

    print('beta: %s, n_sweeps: %d' % (beta, n_sweeps))

    fig = plt.figure()
    plt.plot(range(n_sweeps + 1), sum_1_arr, color='green', linewidth=0.9)
    plt.plot(range(n_sweeps + 1), sum_2_arr, color='darkblue', linewidth=0.9)
    plt.xlabel('Sweeps')
    plt.ylabel('Sum of Image')
    plt.grid()
    plt.title('Total Magnetization of Ising Model with beta = %s' % beta)
    plt.legend(['White Chain (Upper Bound)', 'Black Chain (Lower Bound)'], loc='upper right
    fig.savefig('./exact-sampling-imgs/chains-beta=' + str(beta) + '.png')

    fig = plt.figure()
    plt.imshow(x_1, cmap='gray')
    plt.title('Ising Sample at Coalesence with beta = %s' % beta)
    fig.savefig('./exact-sampling-imgs/sample-beta=' + str(beta) + '.png')

# do afterwards
# fig = plt.figure()
# n_sweeps = np.array([25, 53, 69, 458, 372, 887, 883, 15330])
# plt.plot(betas[0:8], n_sweeps)
# plt.xlabel('Beta')
# plt.ylabel('Sweeps')
# plt.grid()
# plt.title('Coalesence Time Tau for Beta Values')
# fig.savefig('./exact-sampling-imgs/tau-over-beta.png')

```

```
beta: 0.5, n_sweeps: 19
beta: 0.6, n_sweeps: 55
beta: 0.7, n_sweeps: 84
```