# Multimedia
# Lecture 9

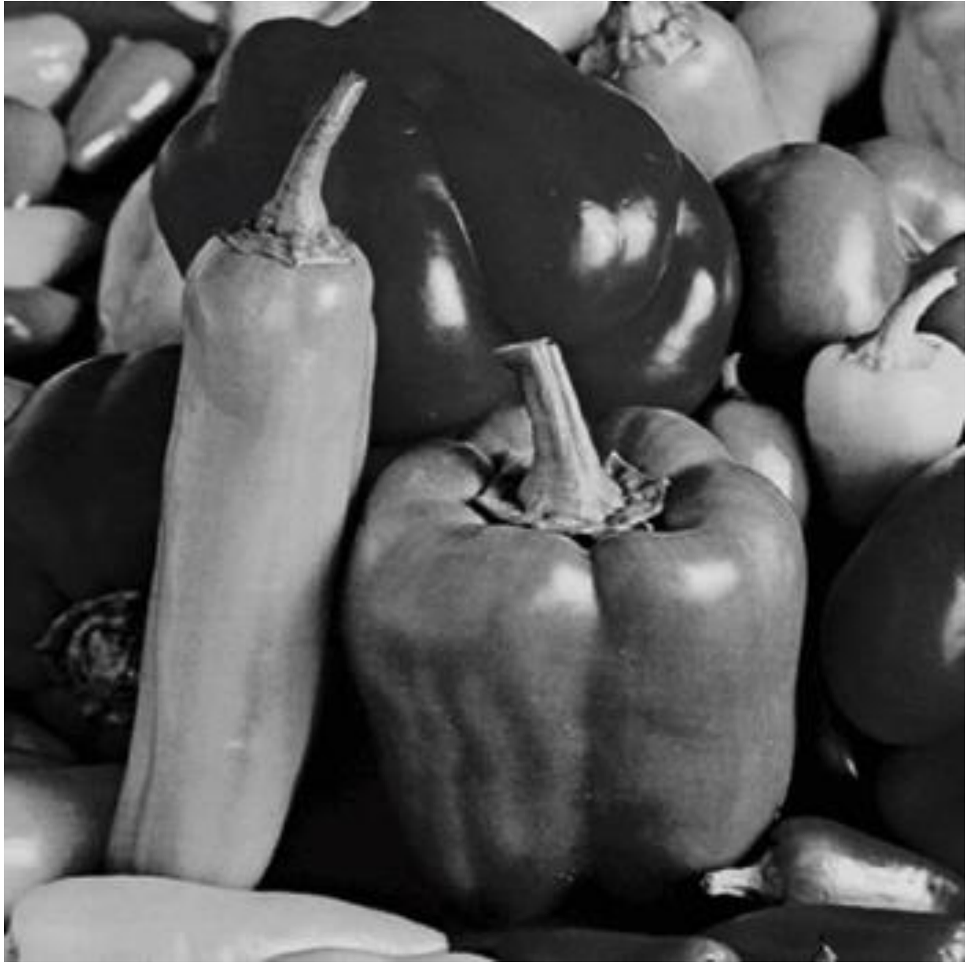
**Dr. Mona M.Soliman**

Faculty of Computers and Artificial Intelligence

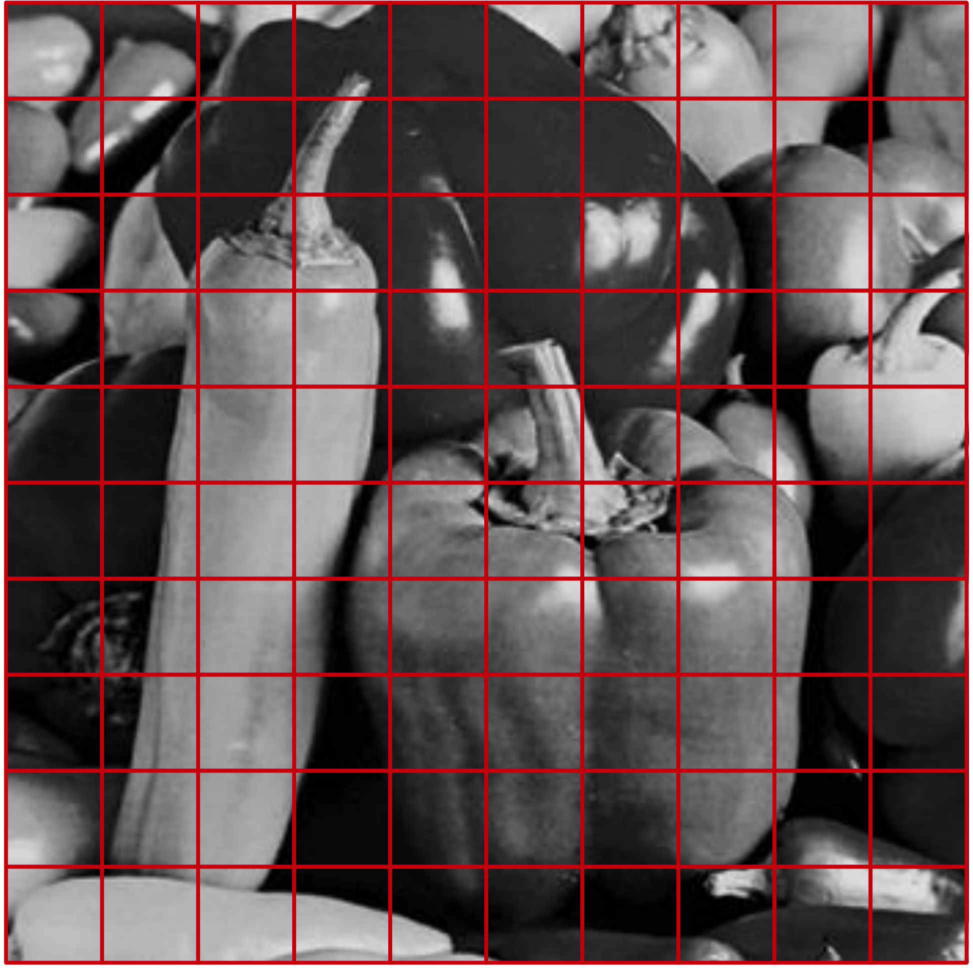
Cairo University

Fall 2022

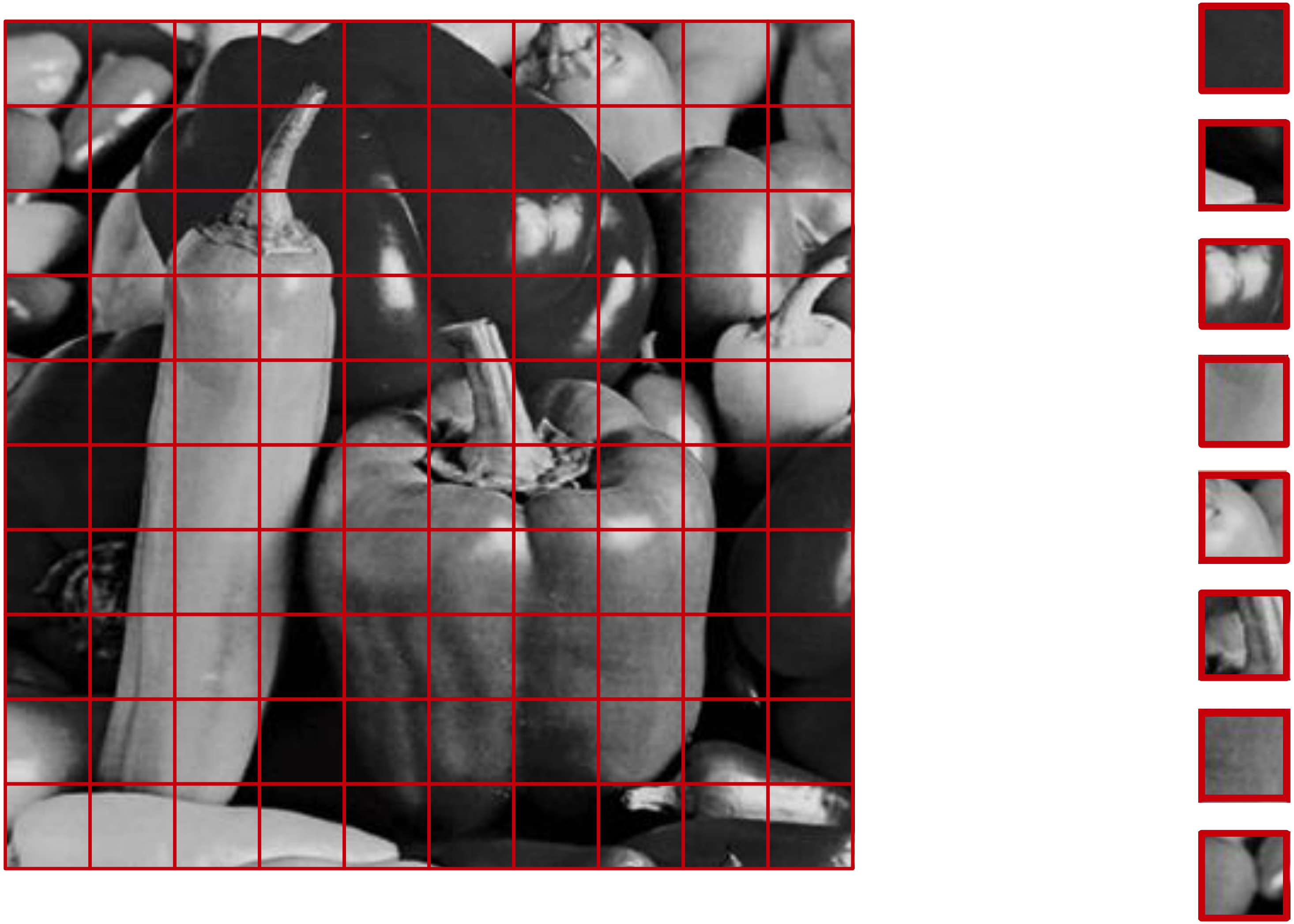
# Original Image



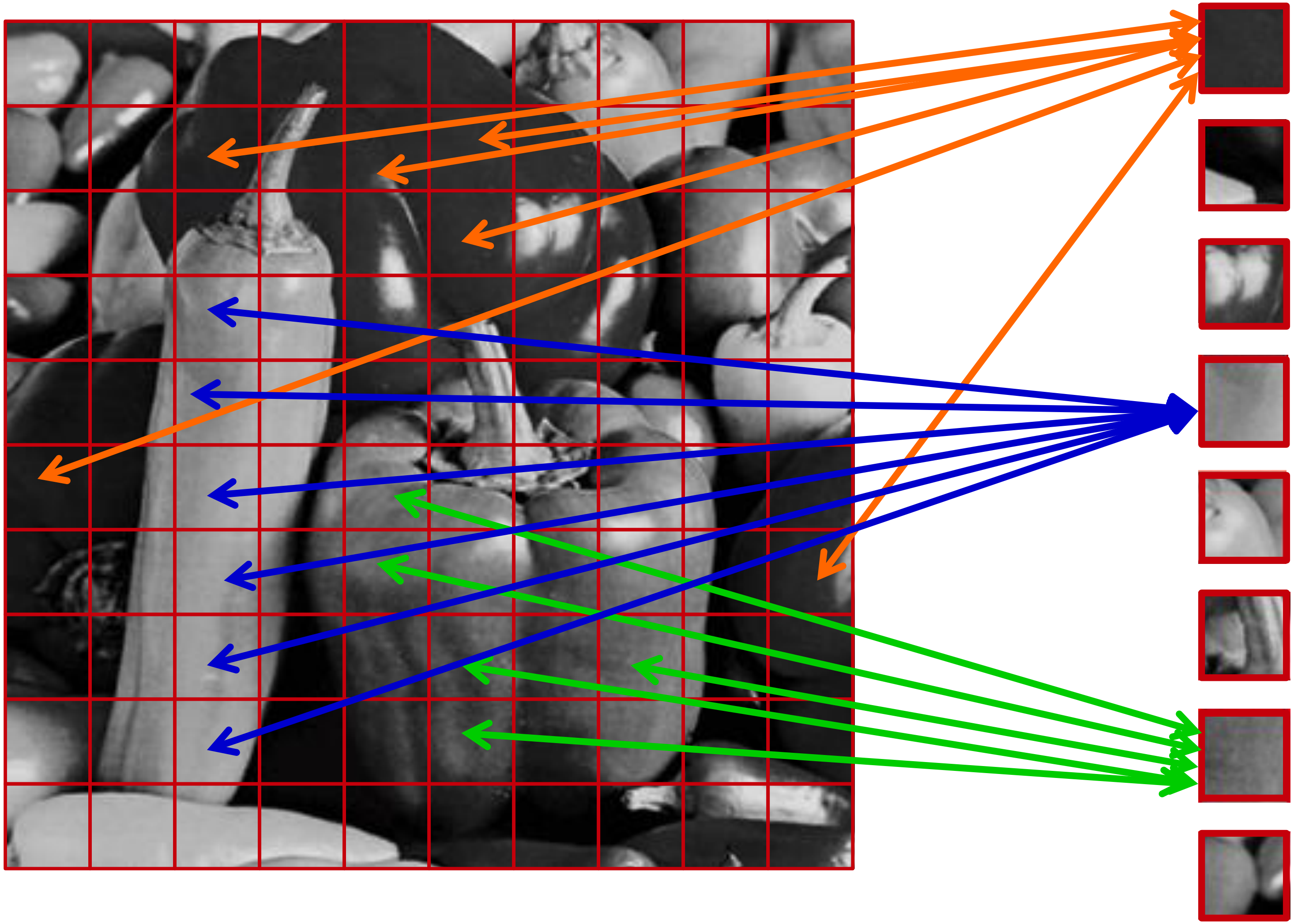
# Divide Image into Blocks (Vectors)



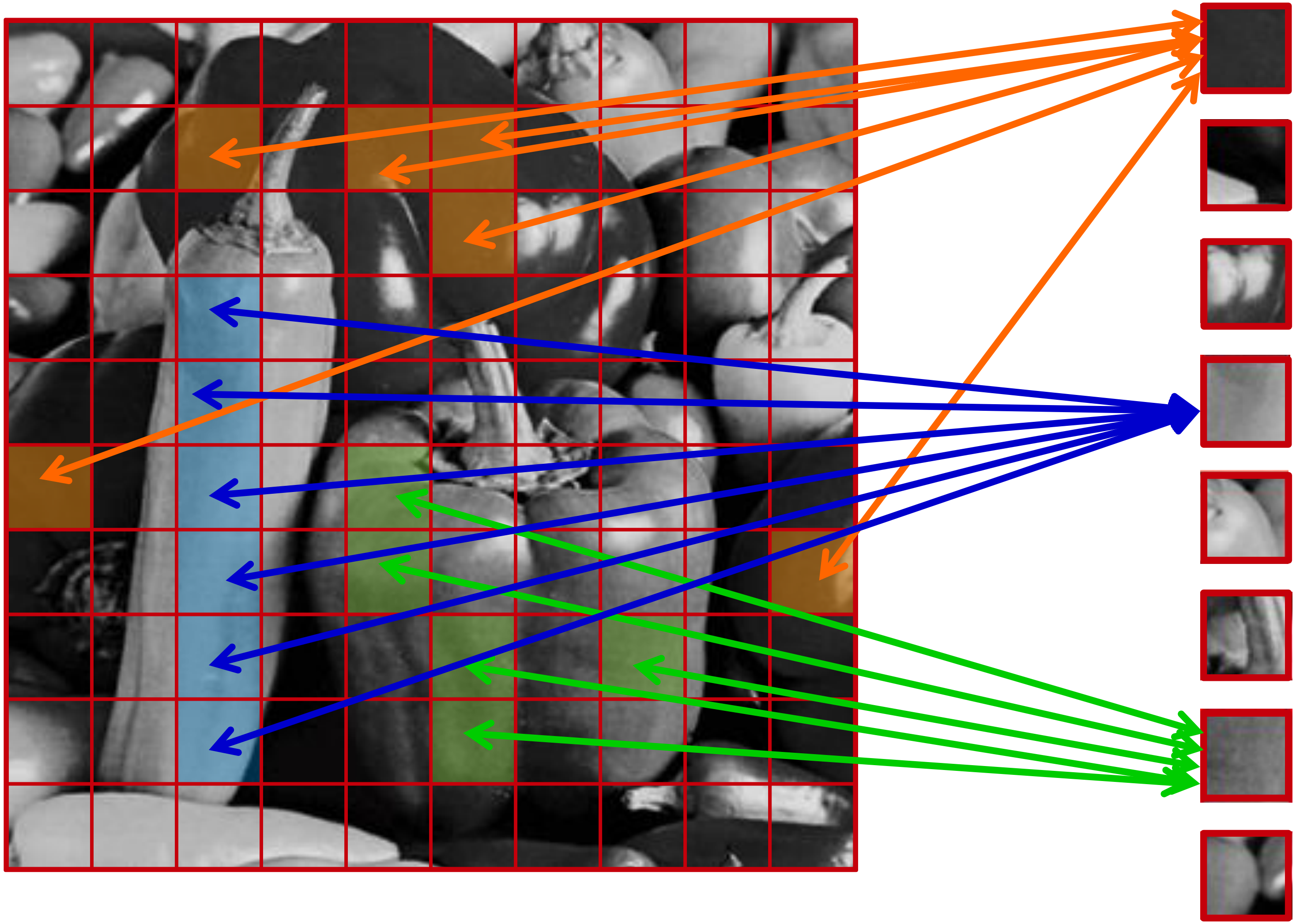
# Generate Best “K” Vectors that can be used to Re-Construct Original Image



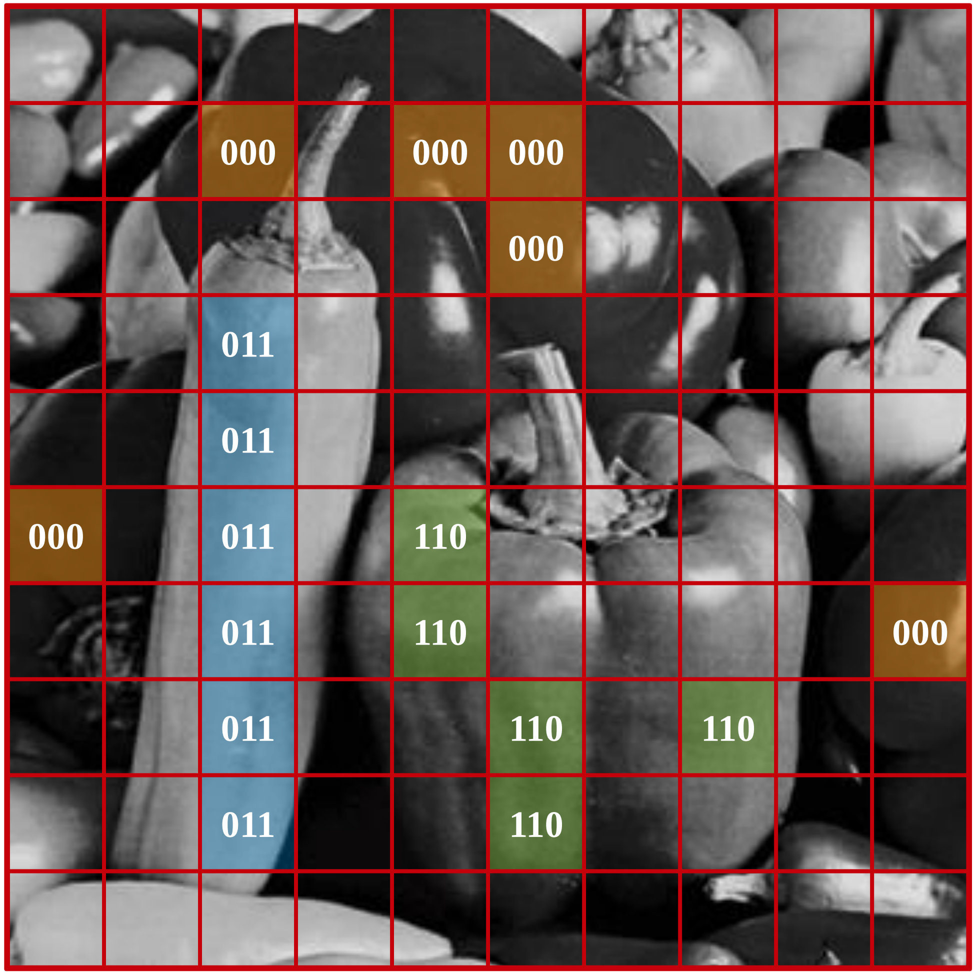
# For Each Block in the Image, Select the Nearest Vector (Using Euclidean Distance)

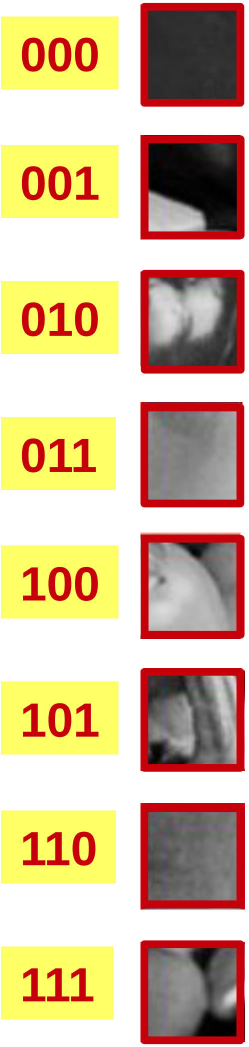


# Label each Block in the image with INDEX of Nearest Vector (in the Codebook)



# Label each Block in the image with INDEX of Nearest Vector (in the Codebook)



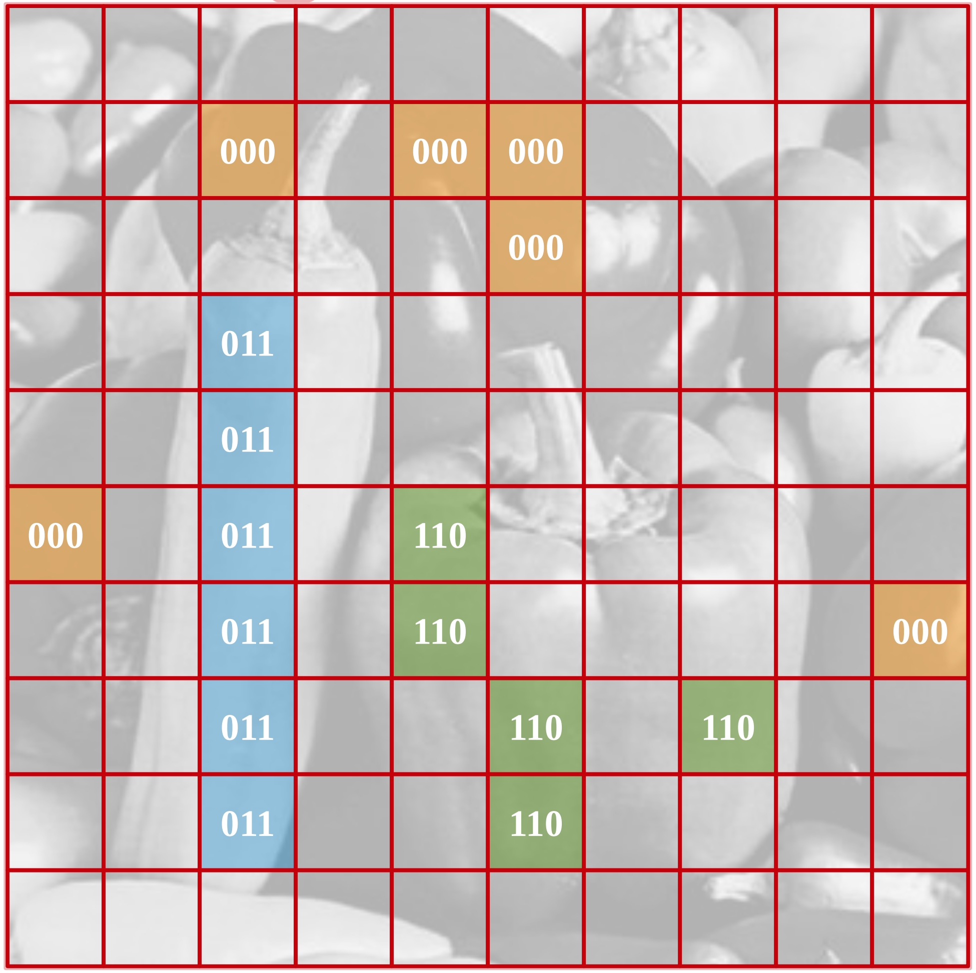


# Compression Ratio

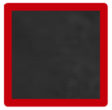
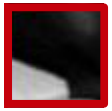
In order to Re-Construct the Image,  it is required to have:

- **All Labels** (one label for each BLOCK in the Image)
- **The Codebook itself** which consists of K Vectors, each vector is a small Image with size equal to BLOCK size

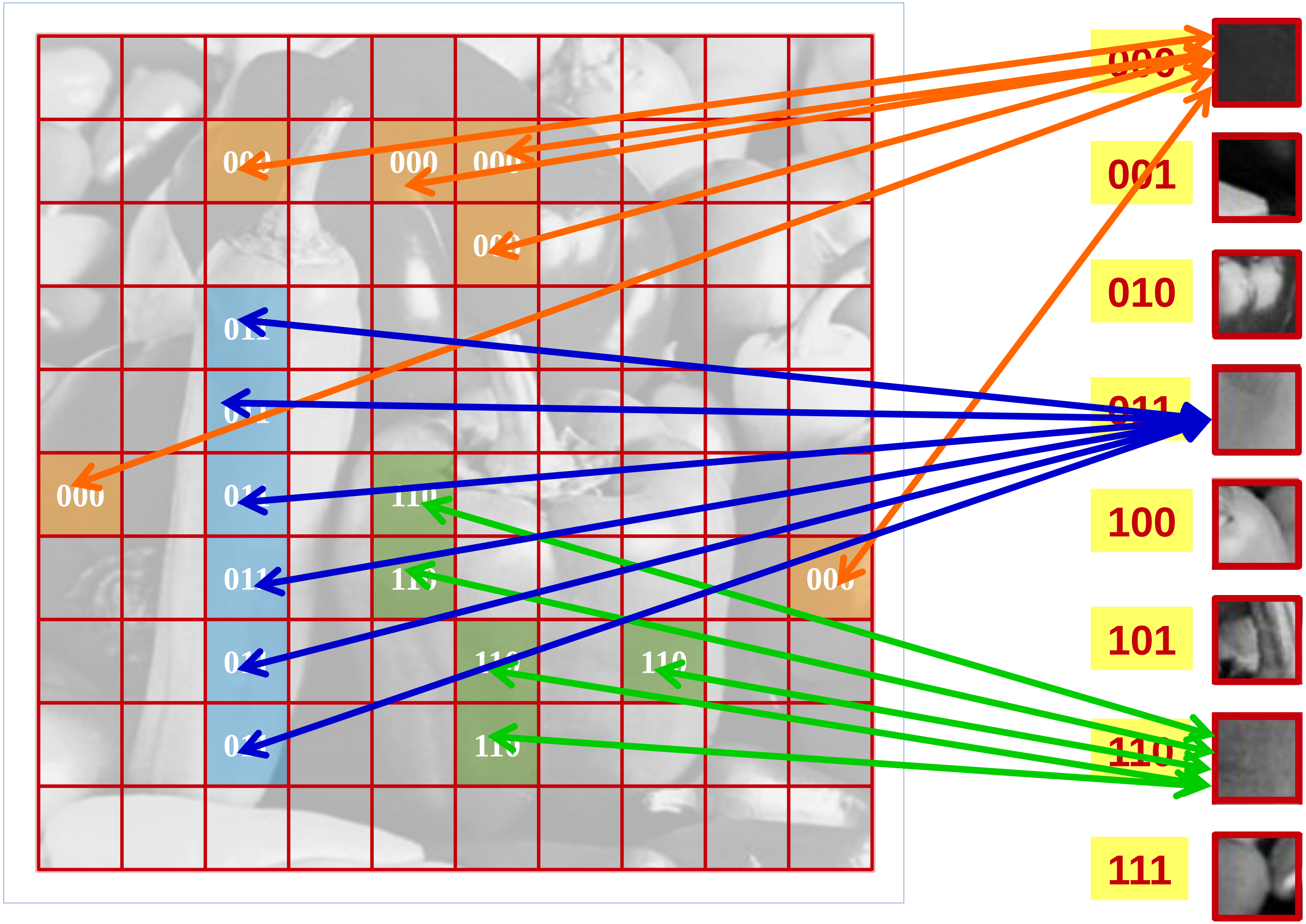
# Image Reconsturction



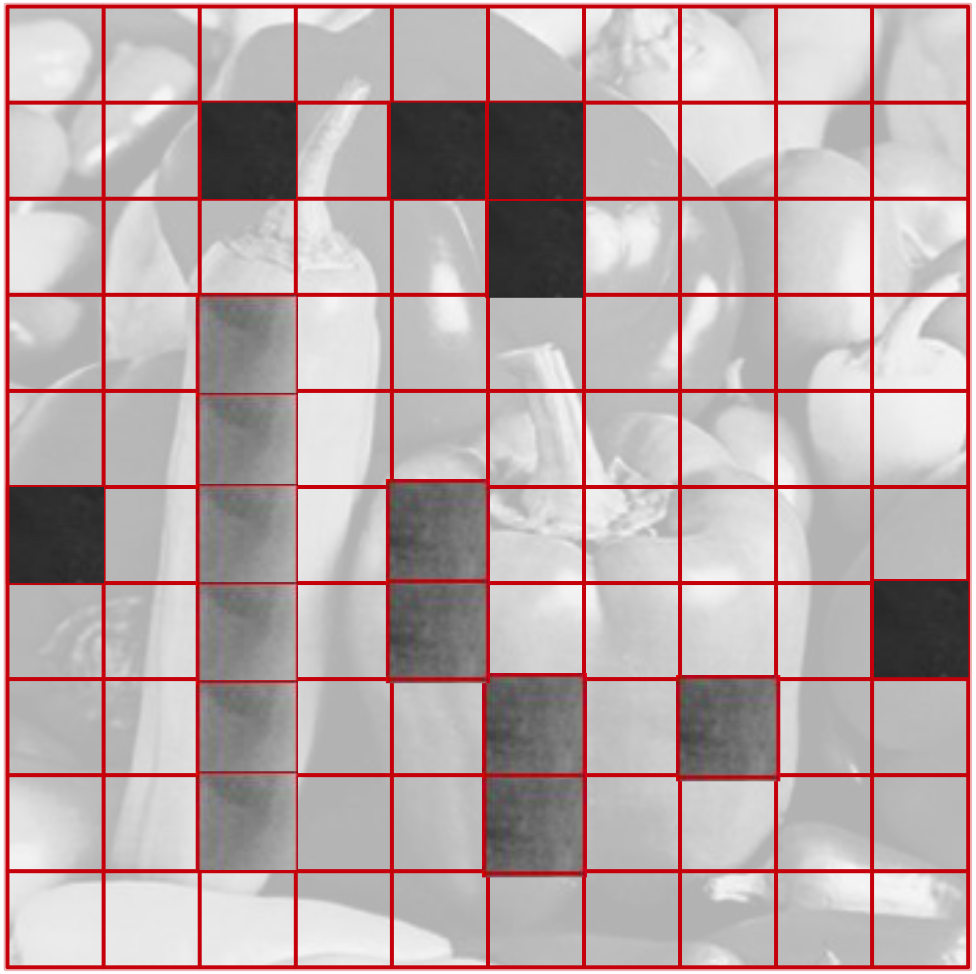
| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |
| | | 000 | | 000 | 000 | | | | |
| | | | | | 000 | | | | |
| | | 011 | | | | | | | |
| | | 011 | | | | | | | |
| 000 | | 011 | | 110 | | | | | |
| | | 011 | | 110 | | | | | 000 |
| | | 011 | | | 110 | | 110 | | |
| | | 011 | | | 110 | | | | |
| | | | | | | | | | |

| | |
|---|---|
| 000 |  |
| 001 |  |
| 010 | |
| 011 | |
| 100 | |
| 101 | |
| 110 | |
| 111 | |

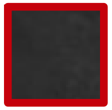
# Substitute Each LABEL with Corresponding Vector in the Codebook



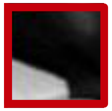
# Obtained Constructed Image



000



001



010

011

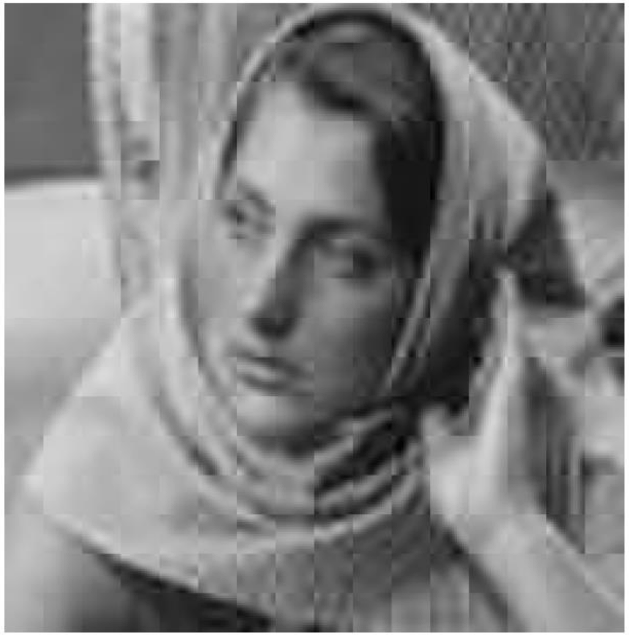
100

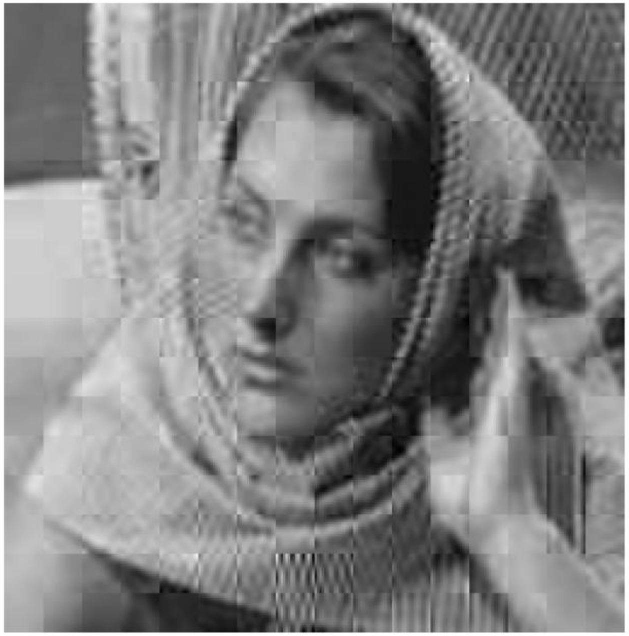
101

110

111

# Vector Quantization  Samples





**Original**

**Blocking Effect**

**Number of Vectors in codebook is small**

# Introduction of Vector Quantization (1/2)

- Efficient scheme for image compression Component
- **Codebooks**
- Generated by using the iterative clustering algorithm
- **Encoder**
- Image is first partitioned into non-overlapping rectangular blocks (vectors)
- Each vector is quantized (indexed) to the closest codeword in the codebook
- **Decoder**
- Select the corresponding codeword in the codebook via indexes

# Introduction of Vector Quantization (2/2)

- What is closest codeword
  - Small Normal distance
  -

$$d(x,y)=\sum_{i=1}^{n}|x_i-y_i|$$

- How to generate codebooks
  - Cluster algorithm

Linde-Buzo-Gray (LBG)

# LBG algorithm

1. Divide image into blocks. Choose a block (k-dimension) $X=(x_1, x_1,…,x_1)$ as initial vector.
2. Spit X vector into two vector
   $Y=(y_1, y_1,…,y_1)$ and $Z=(z_1, z_1,…,z_1)$
   $y_i=x_i-\delta$ ,$z_i=x_i+\delta$
3. Y and Z are centroids. For all blocks, find the nearest centroid. Re-compute the centroid of blocks and get new centroid Y' and Z'.
4. Recursively, do Y' and Z'. Repeat 2 ,3 step. Until find enough number of codevector.

# Vector Quantization using Splitting (Example)

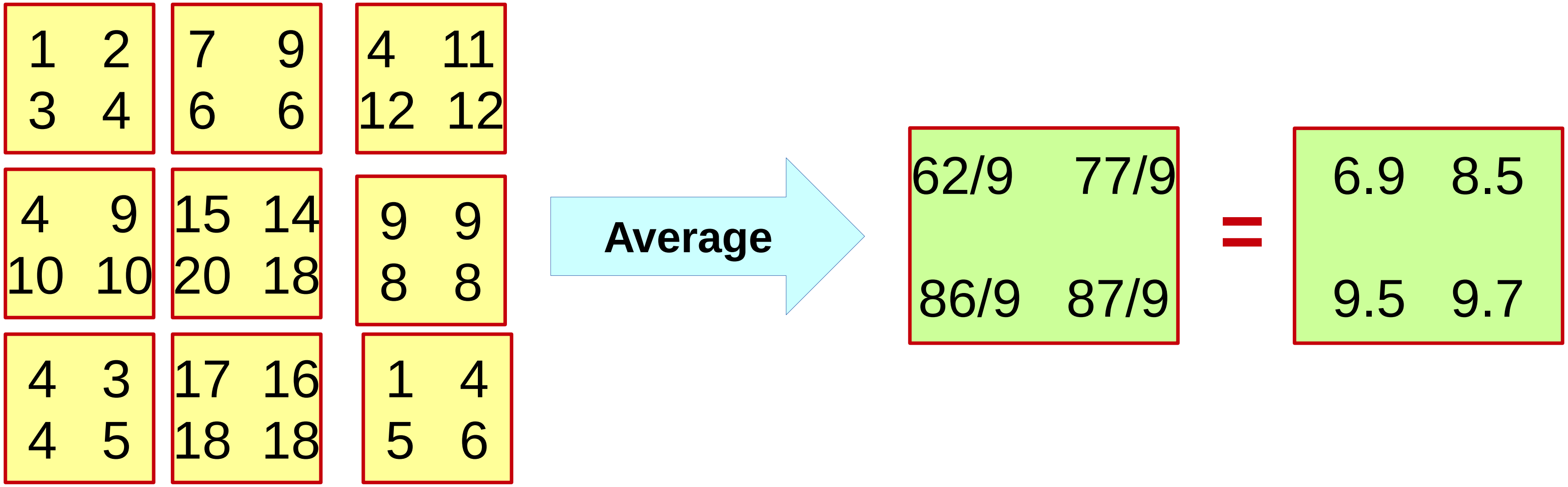
*(a )Compress the following Image Using Vector Quantization*
*(initialize LBG Algorithm using Splitting)*
*(Each pixel is saved in 8 bits)*

**Vector size = 2*2, Number of Vectors in Codebook = 4**

| | | | | | |
|---|---|---|---|---|---|
| 1 | 2 | 7 | 9 | 4 | 11 |
| 3 | 4 | 6 | 6 | 12 | 12 |
| 4 | 9 | 15 | 14 | 9 | 9 |
| 10 | 10 | 20 | 18 | 8 | 8 |
| 4 | 3 | 17 | 16 | 1 | 4 |
| 4 | 5 | 18 | 18 | 5 | 6 |

**(b) Reconstruct the Compressed Image,**
**Calculate Mean Square error between Original and Reconstructed Image**
**(c ) Calculate Compression Ratio**
**(d) Re-Calculate Compression Ratio if the image is 600*600 pixels**

# Vector Quantization using Splitting (Apply Splitting)



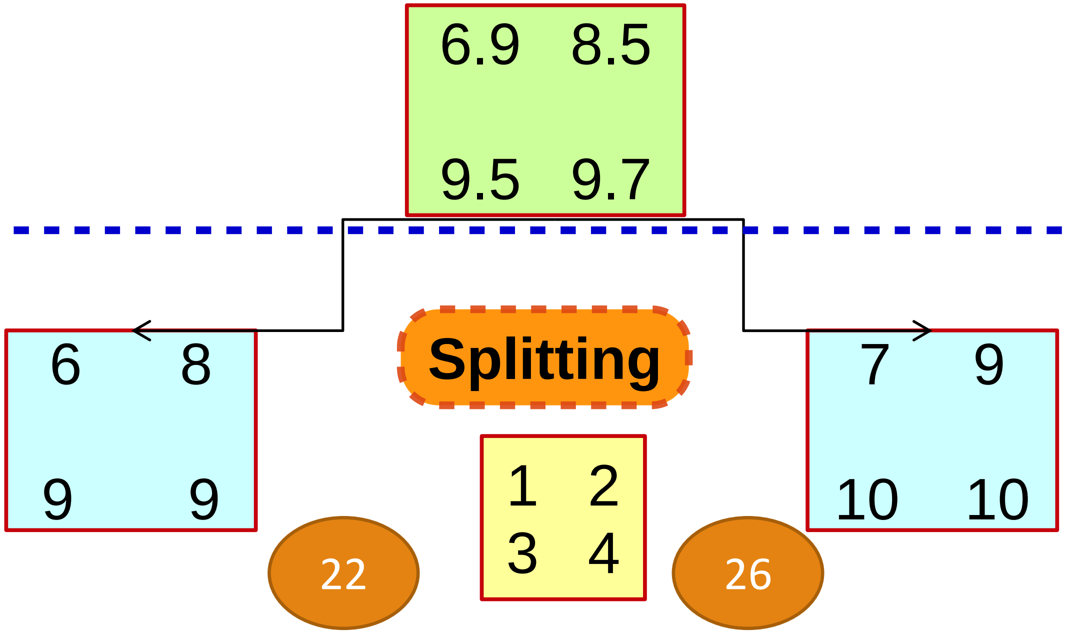
$$\begin{matrix} \boxed{\begin{matrix}1 & 2\\ 3 & 4\end{matrix}} & \boxed{\begin{matrix}7 & 9\\ 6 & 6\end{matrix}} & \boxed{\begin{matrix}4 & 11\\ 12 & 12\end{matrix}} \\ \boxed{\begin{matrix}4 & 9\\ 10 & 10\end{matrix}} & \boxed{\begin{matrix}15 & 14\\ 20 & 18\end{matrix}} & \boxed{\begin{matrix}9 & 9\\ 8 & 8\end{matrix}} \\ \boxed{\begin{matrix}4 & 3\\ 4 & 5\end{matrix}} & \boxed{\begin{matrix}17 & 16\\ 18 & 18\end{matrix}} & \boxed{\begin{matrix}1 & 4\\ 5 & 6\end{matrix}} \end{matrix} \xrightarrow{\textbf{Average}} \boxed{\begin{matrix}62/9 & 77/9\\ 86/9 & 87/9\end{matrix}} = \boxed{\begin{matrix}6.9 & 8.5\\ 9.5 & 9.7\end{matrix}}$$

# Vector Quantization using Splitting (Apply Splitting)

$\begin{bmatrix}1&2\\3&4\end{bmatrix}$ $\begin{bmatrix}7&9\\6&6\end{bmatrix}$ $\begin{bmatrix}4&11\\12&12\end{bmatrix}$

$\begin{bmatrix}4&9\\10&10\end{bmatrix}$ $\begin{bmatrix}15&14\\20&18\end{bmatrix}$ $\begin{bmatrix}9&9\\8&8\end{bmatrix}$

$\begin{bmatrix}4&3\\4&5\end{bmatrix}$ $\begin{bmatrix}17&16\\18&18\end{bmatrix}$ $\begin{bmatrix}1&4\\5&6\end{bmatrix}$



**Nearest Vector**

|1-6|+|2-8|+|3-9|+|4-9|=22

|1-7|+|2-9|+|3-10|+|4-10|=26

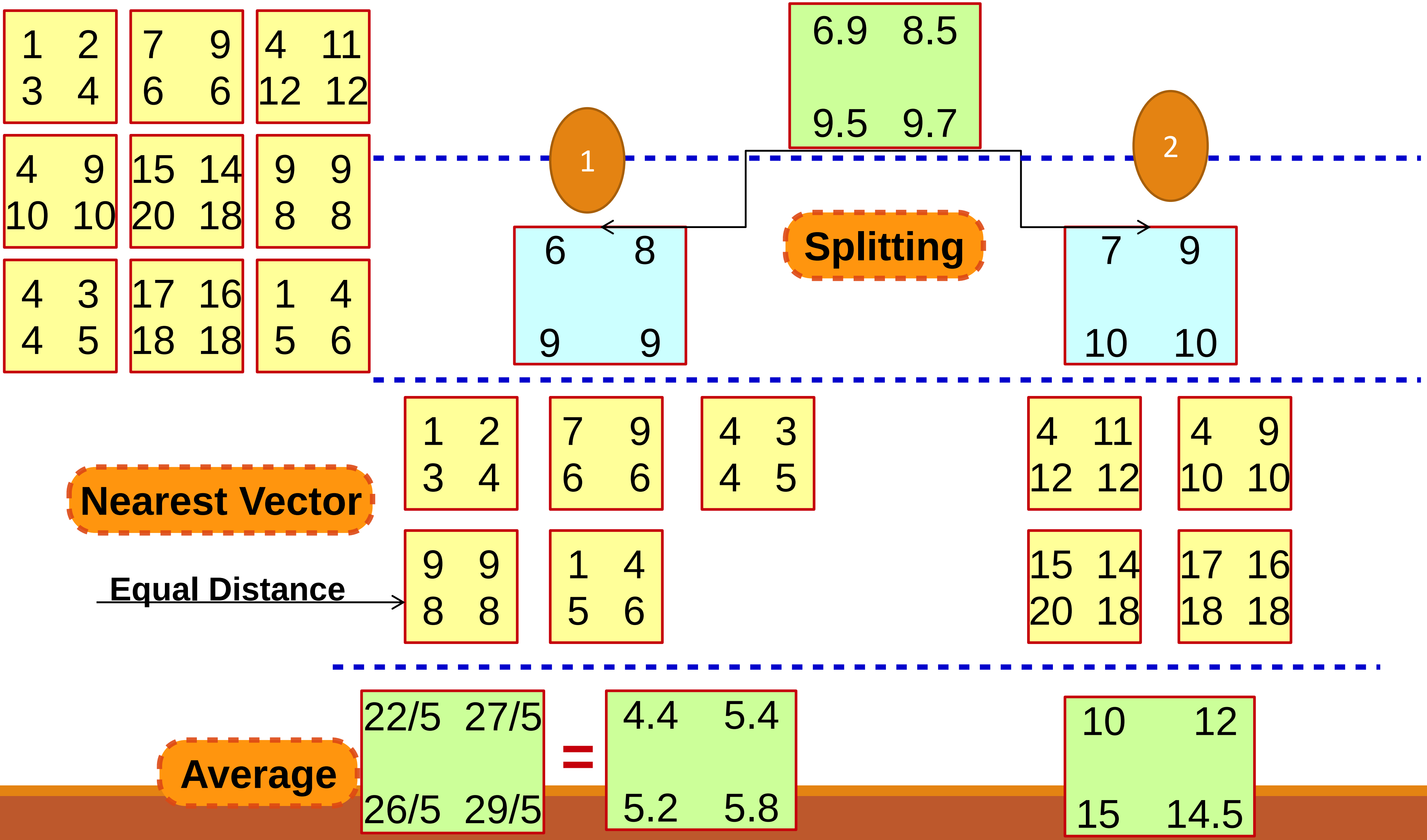
(1+7+4+4+15+9+4+17+1)/9=6.9

(4+6+12+10+18+8+5+18+6)/9=9.7

(2+9+11+9+14+9+3+16+4)/9=8.5

(3+6+12+10+20+8+4+18+5)/9=9.5

# Vector Quantization using Splitting (Apply Splitting)



$\begin{bmatrix}1&2\\3&4\end{bmatrix}$ $\begin{bmatrix}7&9\\6&6\end{bmatrix}$ $\begin{bmatrix}4&11\\12&12\end{bmatrix}$

$\begin{bmatrix}4&9\\10&10\end{bmatrix}$ $\begin{bmatrix}15&14\\20&18\end{bmatrix}$ $\begin{bmatrix}9&9\\8&8\end{bmatrix}$

$\begin{bmatrix}4&3\\4&5\end{bmatrix}$ $\begin{bmatrix}17&16\\18&18\end{bmatrix}$ $\begin{bmatrix}1&4\\5&6\end{bmatrix}$

$\begin{bmatrix}6.9&8.5\\9.5&9.7\end{bmatrix}$

Splitting

1 $\begin{bmatrix}6&8\\9&9\end{bmatrix}$

2 $\begin{bmatrix}7&9\\10&10\end{bmatrix}$

Nearest Vector

$\begin{bmatrix}1&2\\3&4\end{bmatrix}$ $\begin{bmatrix}7&9\\6&6\end{bmatrix}$ $\begin{bmatrix}4&3\\4&5\end{bmatrix}$ $\begin{bmatrix}9&9\\8&8\end{bmatrix}$ $\begin{bmatrix}1&4\\5&6\end{bmatrix}$

Equal Distance

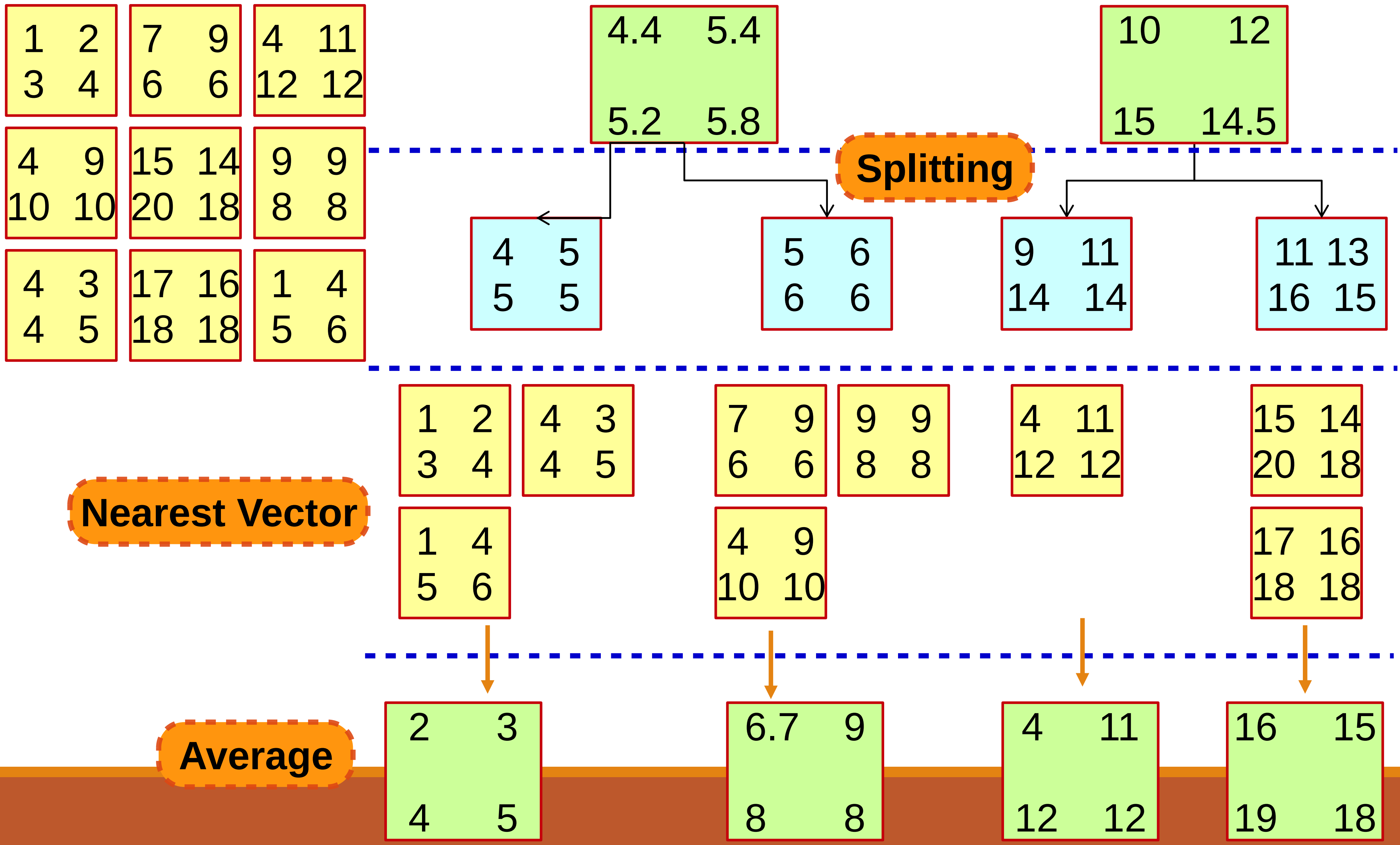
$\begin{bmatrix}4&11\\12&12\end{bmatrix}$ $\begin{bmatrix}4&9\\10&10\end{bmatrix}$ $\begin{bmatrix}15&14\\20&18\end{bmatrix}$ $\begin{bmatrix}17&16\\18&18\end{bmatrix}$

Average

$\begin{bmatrix}22/5&27/5\\26/5&29/5\end{bmatrix} = \begin{bmatrix}4.4&5.4\\5.2&5.8\end{bmatrix}$

$\begin{bmatrix}10&12\\15&14.5\end{bmatrix}$

# Vector Quantization using Splitting (Apply Splitting)



1 2
3 4

7 9
6 6

4 11
12 12

4 9
10 10

15 14
20 18

9 9
8 8

4 3
4 5

17 16
18 18

1 4
5 6

4.4 5.4
5.2 5.8

10 12
15 14.5

Splitting

4 5
5 5

5 6
6 6

9 11
14 14

11 13
16 15

Nearest Vector

1 2
3 4

4 3
4 5

1 4
5 6

7 9
6 6

9 9
8 8

4 9
10 10

4 11
12 12

15 14
20 18

17 16
18 18

Average

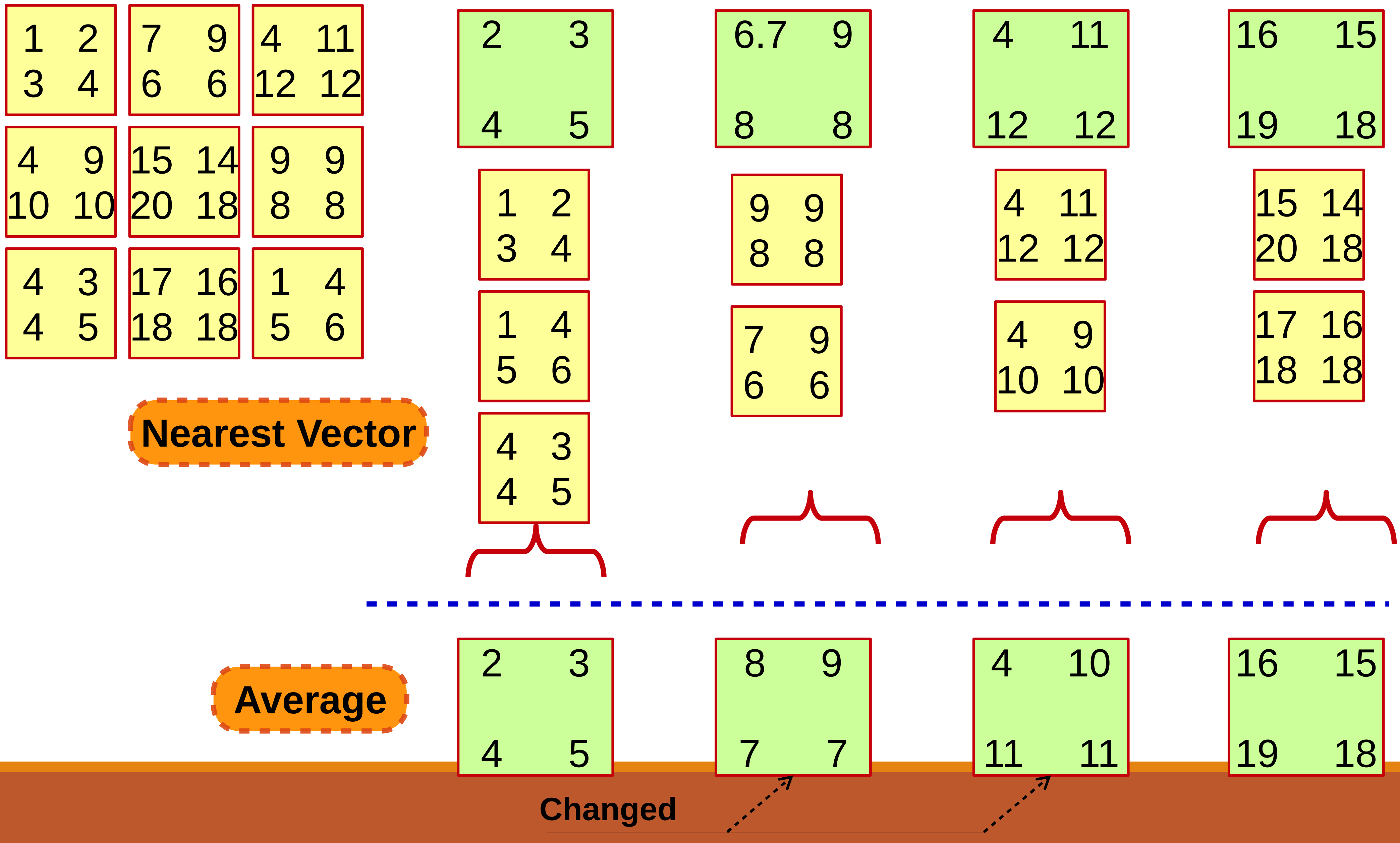
2 3
4 5

6.7 9
8 8

4 11
12 12

16 15
19 18

# Vector Quantization using Splitting (LBG Algorithm)



| | | |
|---|---|---|
| 1 2<br>3 4 | 7 9<br>6 6 | 4 11<br>12 12 |
| 4 9<br>10 10 | 15 14<br>20 18 | 9 9<br>8 8 |
| 4 3<br>4 5 | 17 16<br>18 18 | 1 4<br>5 6 |

| | | | |
|---|---|---|---|
| 2 3<br>4 5 | 6.7 9<br>8 8 | 4 11<br>12 12 | 16 15<br>19 18 |

Nearest Vector

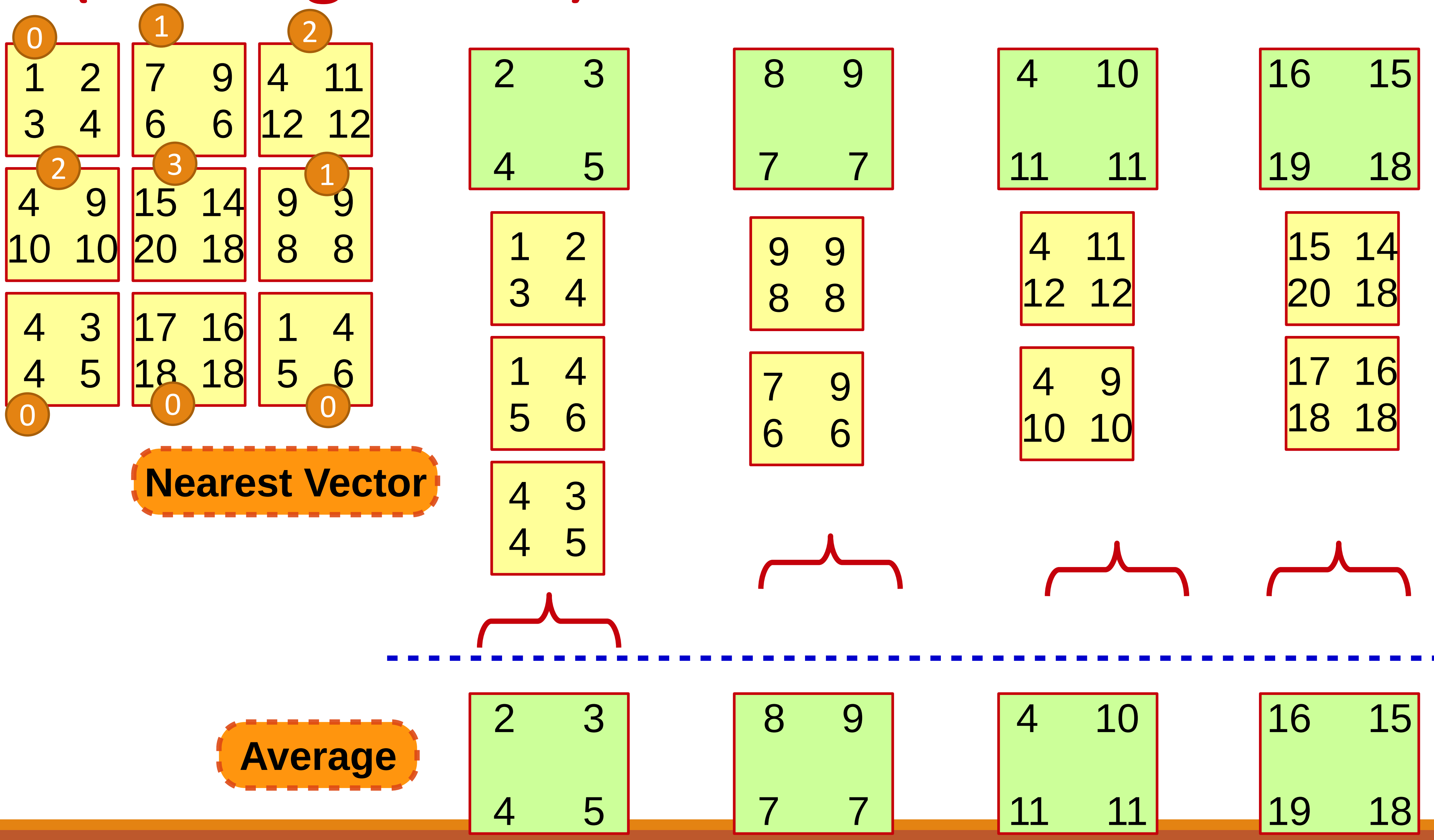
| | | | |
|---|---|---|---|
| 1 2<br>3 4 | 9 9<br>8 8 | 4 11<br>12 12 | 15 14<br>20 18 |
| 1 4<br>5 6 | 7 9<br>6 6 | 4 9<br>10 10 | 17 16<br>18 18 |
| 4 3<br>4 5 | | | |

Average

| | | | |
|---|---|---|---|
| 2 3<br>4 5 | 8 9<br>7 7 | 4 10<br>11 11 | 16 15<br>19 18 |

Changed

# Vector Quantization using Splitting (LBG Algorithm)



0
1 2
3 4

1
7 9
6 6

2
4 11
12 12

2
4 9
10 10

3
15 14
20 18

1
9 9
8 8

4 3
4 5
0

17 16
18 18
0

1 4
5 6
0

**Nearest Vector**

2 3
4 5

8 9
7 7

4 10
11 11

16 15
19 18

1 2
3 4

1 4
5 6

4 3
4 5

9 9
8 8

7 9
6 6

4 11
12 12

4 9
10 10

15 14
20 18

17 16
18 18

**Average**

2 3
4 5

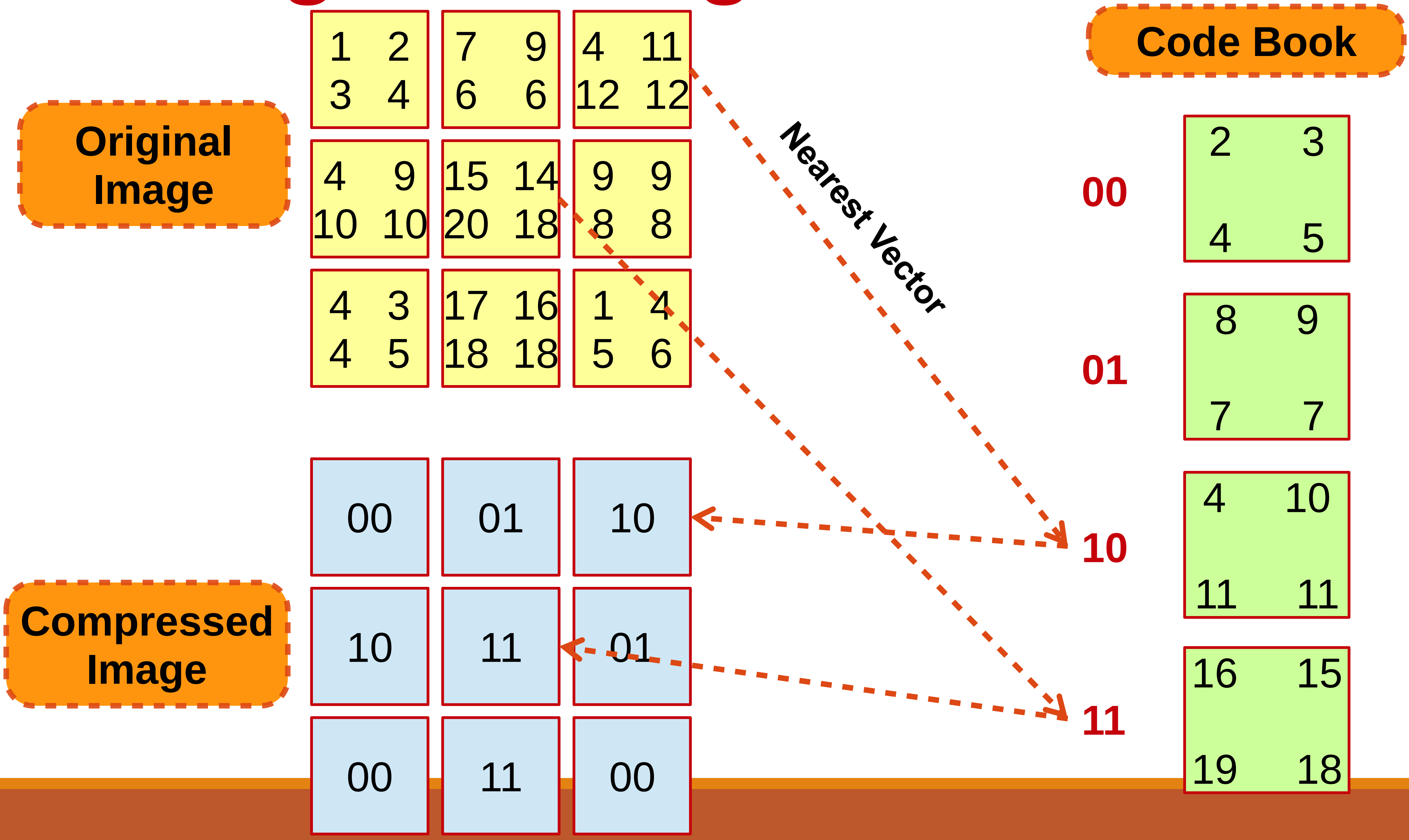
8 9
7 7

4 10
11 11

16 15
19 18

*No Change (Stop Iteration)*

# Image Encoding



Original Image

| | | |
|---|---|---|
| 1 2<br>3 4 | 7 9<br>6 6 | 4 11<br>12 12 |
| 4 9<br>10 10 | 15 14<br>20 18 | 9 9<br>8 8 |
| 4 3<br>4 5 | 17 16<br>18 18 | 1 4<br>5 6 |

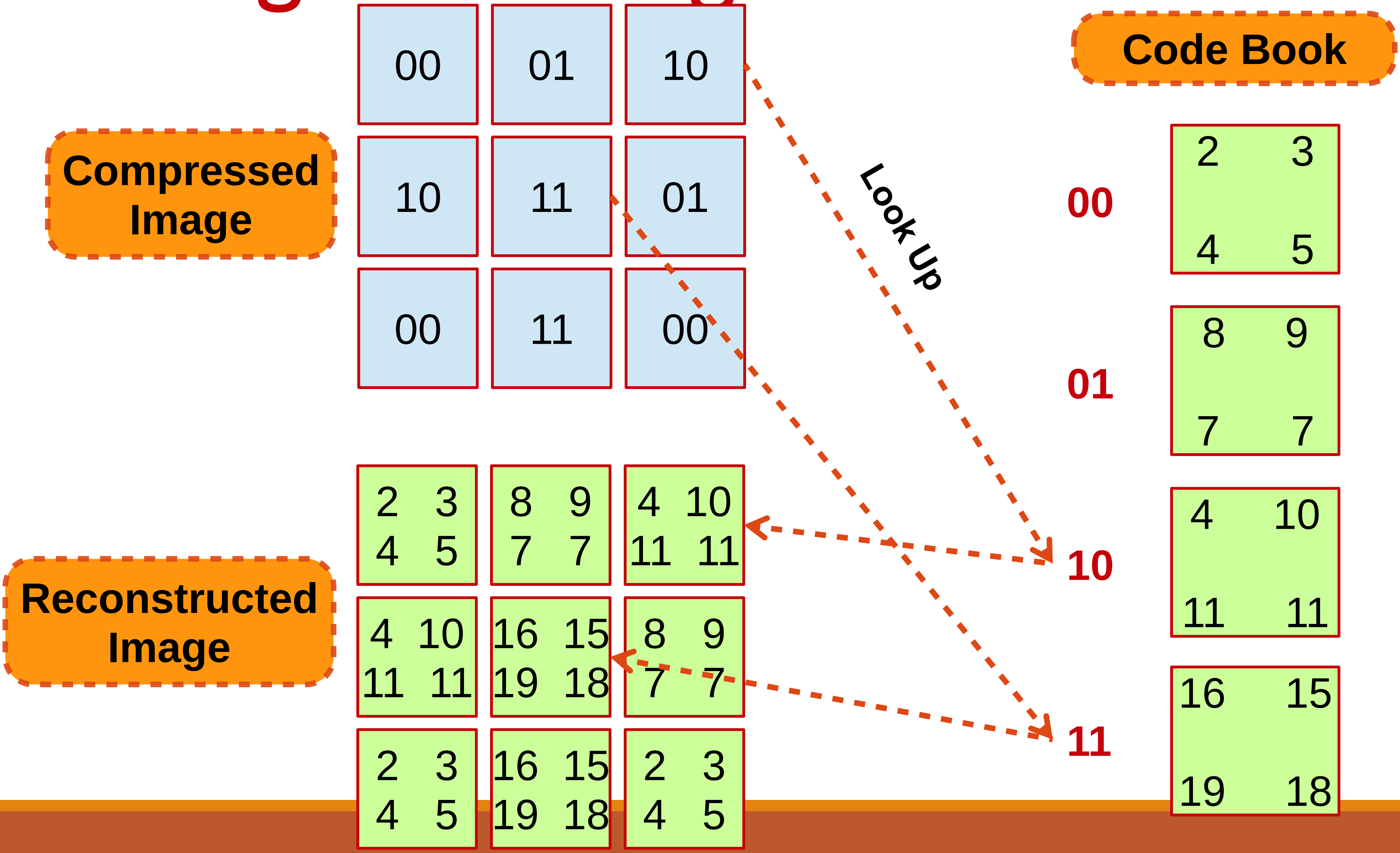
Compressed Image

| | | |
|---|---|---|
| 00 | 01 | 10 |
| 10 | 11 | 01 |
| 00 | 11 | 00 |

Code Book

| | |
|---|---|
| 00 | 2 3<br>4 5 |
| 01 | 8 9<br>7 7 |
| 10 | 4 10<br>11 11 |
| 11 | 16 15<br>19 18 |

# Image Decoding



Compressed Image

| | | |
|---|---|---|
| 00 | 01 | 10 |
| 10 | 11 | 01 |
| 00 | 11 | 00 |

Look Up

Code Book

| | | |
|---|---|---|
| 00 | 2 | 3 |
| | 4 | 5 |
| 01 | 8 | 9 |
| | 7 | 7 |
| 10 | 4 | 10 |
| | 11 | 11 |
| 11 | 16 | 15 |
| | 19 | 18 |

Reconstructed Image

| | | | | | |
|---|---|---|---|---|---|
| 2 | 3 | 8 | 9 | 4 | 10 |
| 4 | 5 | 7 | 7 | 11 | 11 |
| 4 | 10 | 16 | 15 | 8 | 9 |
| 11 | 11 | 19 | 18 | 7 | 7 |
| 2 | 3 | 16 | 15 | 2 | 3 |
| 4 | 5 | 19 | 18 | 4 | 5 |

# Mean Squared Error

**Original Image**

| | | | | | |
|---|---|---|---|---|---|
| 1 | 2 | 7 | 9 | 4 | 11 |
| 3 | 4 | 6 | 6 | 12 | 12 |
| 4 | 9 | 15 | 14 | 9 | 9 |
| 10 | 10 | 20 | 18 | 8 | 8 |
| 4 | 3 | 17 | 16 | 1 | 4 |
| 4 | 5 | 18 | 18 | 5 | 6 |

**Reconstructed Image**

| | | | | | |
|---|---|---|---|---|---|
| 2 | 3 | 8 | 9 | 4 | 10 |
| 4 | 5 | 7 | 7 | 11 | 11 |
| 4 | 10 | 16 | 15 | 8 | 9 |
| 11 | 11 | 19 | 18 | 7 | 7 |
| 2 | 3 | 16 | 15 | 2 | 3 |
| 4 | 5 | 19 | 18 | 4 | 5 |

**Squared Error**

| | | | | | |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 | 1 |
| 4 | 0 | 1 | 1 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 | 1 |

**Mean Squared Error = 30/36 = 0.833**

# Compression Ratio

# Image Decoding

- **Original Image Size**=

  6*6 (pixels) * 8 bits/pixel = 6x6x8= 288 bits

- **Number of Blocks** (vectors) in Image =

  (6*6)/ (2*2)=36/4 = 9 blocks

- Each Block is substituted by 2 Bits Label

- **Labels size** = 9 blocks * 2 bits = 18 bits

- **Codebook size** =

  4 Vectors * (2*2) pixels/vector * 8 bits/pixel = 4*2*2*8=128 bits

- **Total Compressed size** = Codebook +Labels = 128 + 18 = 146 bits

- **Compression Ratio** = 288/146 = 1.97:1

**Code Book**

| | |
|---|---|
| 2 | 3 |
| 4 | 5 |

| | |
|---|---|
| 8 | 9 |
| 7 | 7 |

| | |
|---|---|
| 4 | 10 |
| 11 | 11 |

| | |
|---|---|
| 16 | 15 |
| 19 | 18 |

# Compression Ratio

- What about if Image size 600 x 600

- **Original Image Size**=

  600*600 (pixels) * 8 bits/pixel = 2,880,000 bits

- **Number of Blocks** (vectors) in Image =

  (600*600)/ (2*2)=360000/4 = 90,000 blocks

- Each Block is substituted by 2 Bit Label
- **Labels size** = 90,000 blocks * 2 bits = 180,000 bits
- **Codebook size** = 128 bits (as before)
- **Total Compressed size** = 128 + 180,000 =~ 180,000 bits
- **Compression Ratio** = 2,880,000/180,000 = 16:1

  (each **4 pixels** = 32 bits are substituted with **2 bits label**)