اصول پردازش تصویر

Principles of Image Processing

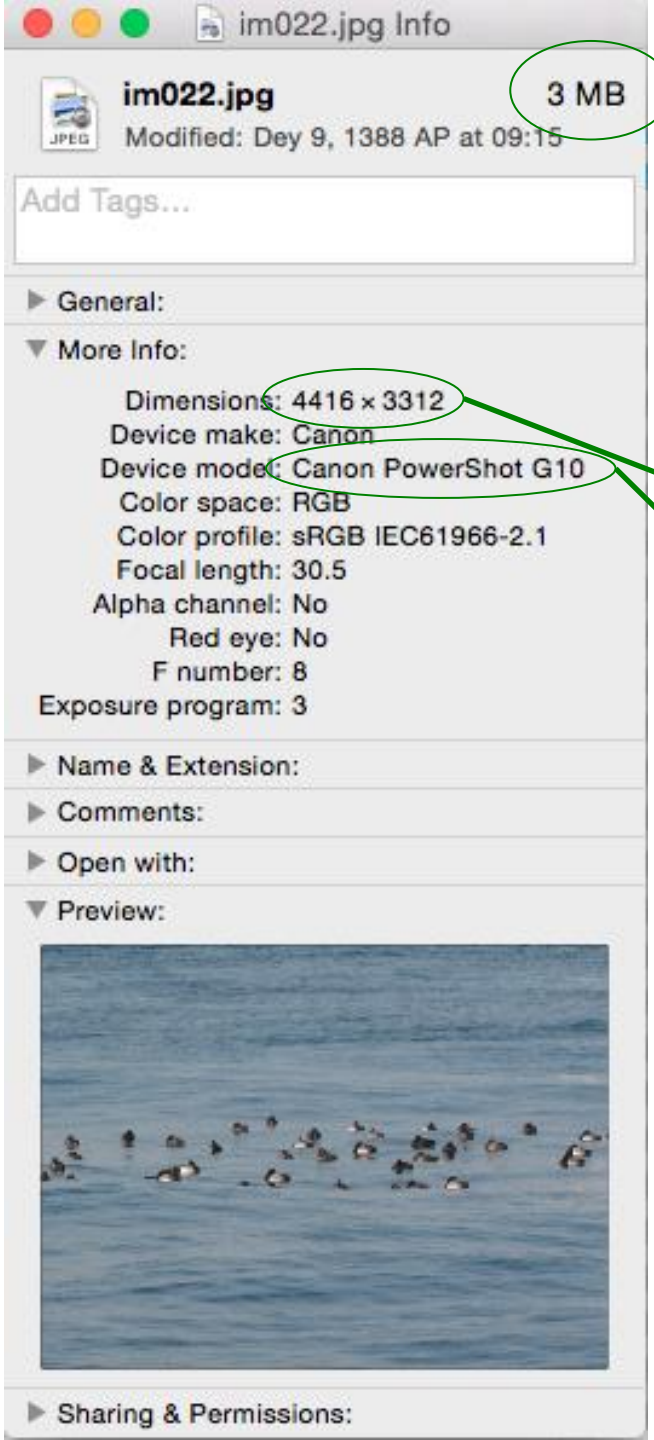
مصطفی کمالی تبریزی

۲۴ و ۲۹ آذر ۱۳۹۹

جلسه بیست پنجم و بیست ششم

Image and Video Compression

Why Image Compression?



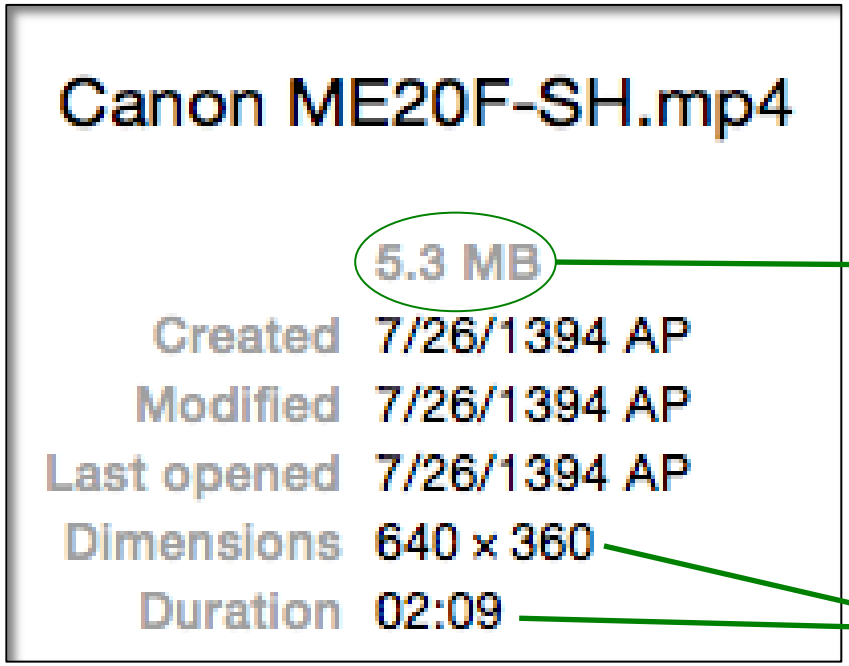
Size on disk: 3 MB

$4{,}416 \times 3{,}312 \times 3 = 43{,}877{,}376$ B
≈ 44 MB

15 MB Camera

Compression Rate = 1 – 3/44 = 0.9318

Why Video Compression?



Size on disk: 5.3 MB

$640 \times 360 \times 3 \times 30 \times 129$
$= 2{,}674{,}944{,}000$ B
≈ 2.7 GB

Compression = 2.7GB / 5.3MB = 509

Compression Rate = 1 – 5.3/2700 = 0.9980

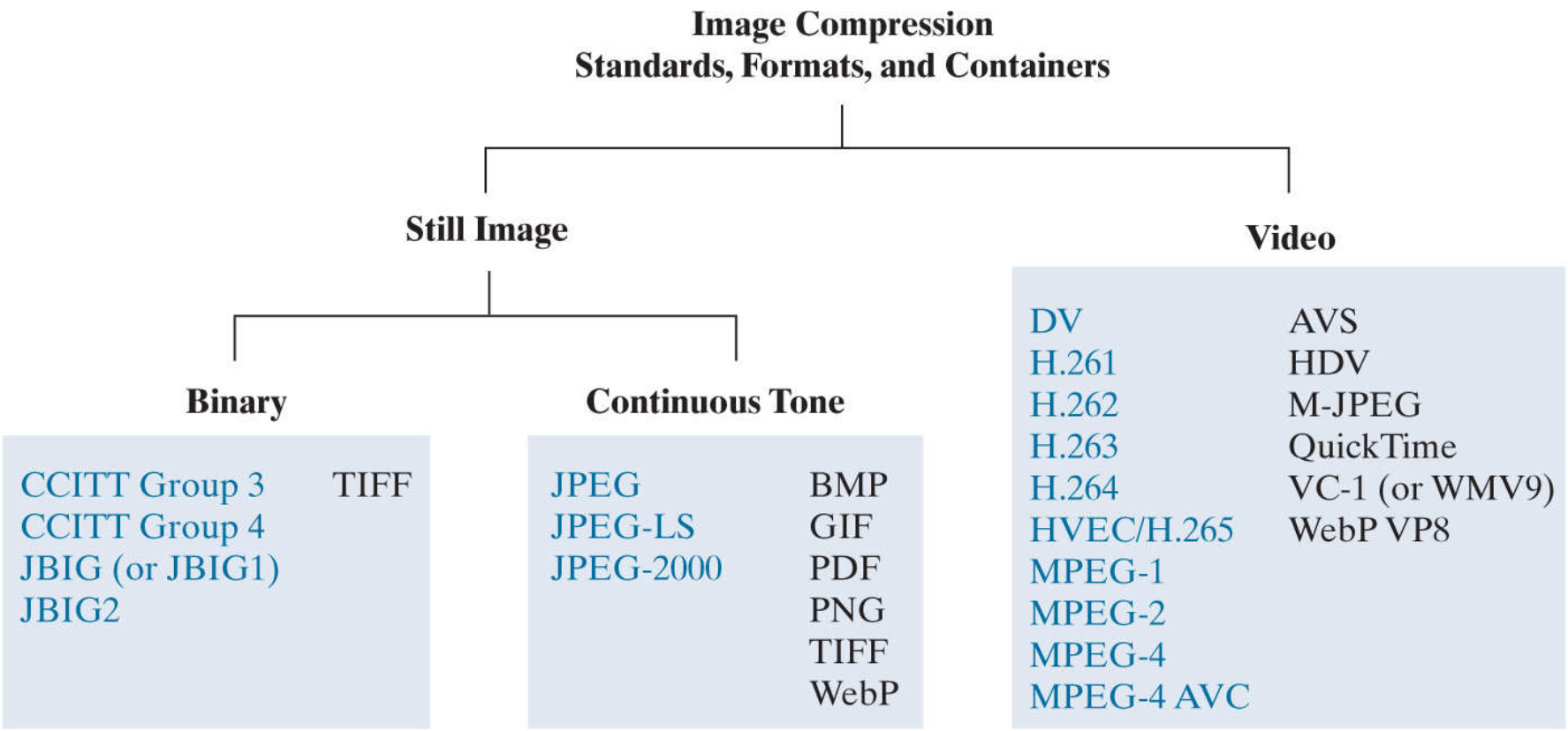
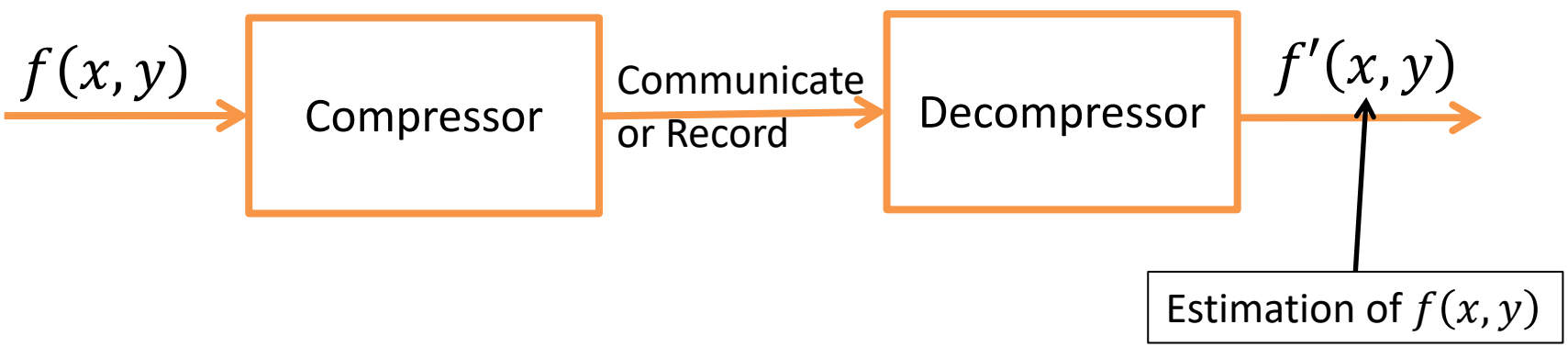


FIGURE 8.6
Some popular image compression standards, file formats, and containers. Internationally sanctioned entries are shown in blue; all others are in black.

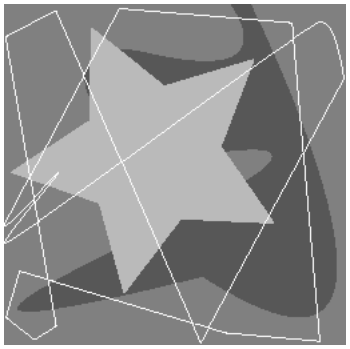
Lossless Compression

Lossy Compression

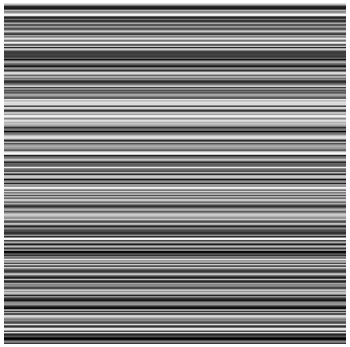


Basic Data Redundancies

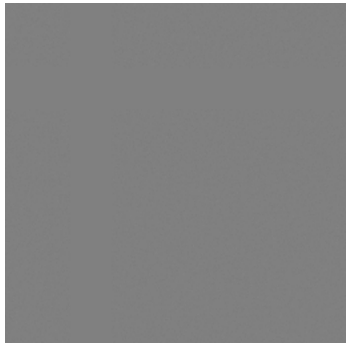
1. ***Coding*** (Code length for each pixel)
   - 8-bit
   - 16-bit
   - variable-length



2. ***Interpixel***
   (Spatial / Geometric / Interframe)
   - Correlation between pixels
   - Run-length coding

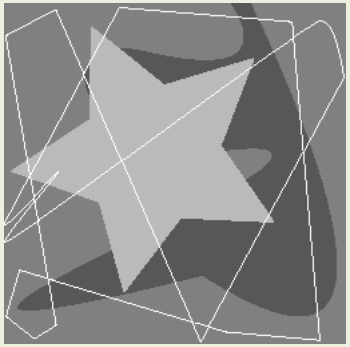


3. ***Psychovisual***
   - Importance for normal visual perception
   - Quantization

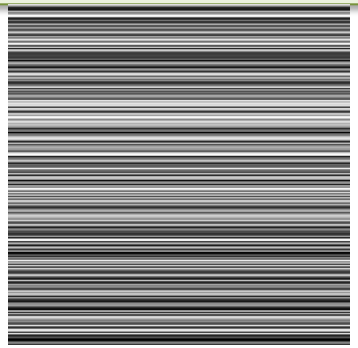


Basic Data Redundancies

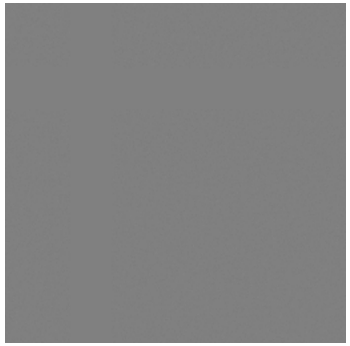
1. ***Coding*** (Code length for each pixel)
   - 8-bit
   - 16-bit
   - variable-length



2. ***Interpixel***
   (Spatial / Geometric / Interframe)
   - Correlation between pixels
   - Run-length coding



3. ***Psychovisual***
   - Importance for normal visual perception
   - Quantization



Example of Variable-Length Coding

r_k	$p_r(r_k)$	Code 1	$l_1(r_k)$	Code 2	$l_2(r_k)$
$r_{87} = 87$	0.25	01010111	8	01	2
$r_{128} = 128$	0.47	10000000	8	1	1
$r_{186} = 186$	0.25	11000100	8	000	3
$r_{255} = 255$	0.03	11111111	8	001	3
r_k for $k \neq 87, 128, 186, 255$	0	—	8	—	0

TABLE 8.1
Example of variable-length coding.

Huffman Coding

Original source		Source reduction			
Symbol	Probability	1	2	3	4
a_2	0.4	0.4	0.4	0.4	0.6
a_6	0.3	0.3	0.3	0.3	0.4
a_1	0.1	0.1	0.2	0.3	
a_4	0.1	0.1	0.1		
a_3	0.06	0.1			
a_5	0.04				

FIGURE 8.7
Huffman source reductions.

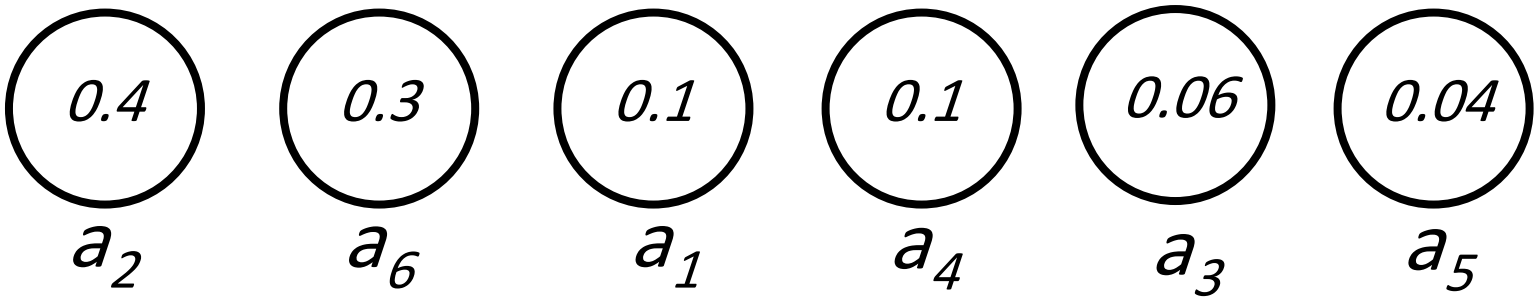
Huffman Coding

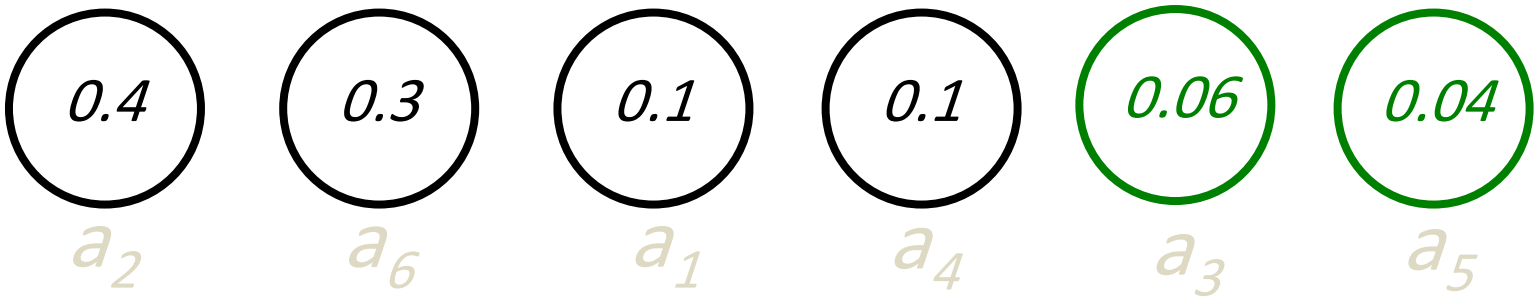
Original source			Source reduction							
Symbol	Probability	Code	1		2		3		4	
a_2	0.4	1	0.4	1	0.4	1	0.4	1	0.6	0
a_6	0.3	00	0.3	00	0.3	00	0.3	00	0.4	1
a_1	0.1	011	0.1	011	0.2	010	0.3	01		
a_4	0.1	0100	0.1	0100	0.1	011				
a_3	0.06	01010	0.1	0101						
a_5	0.04	01011								

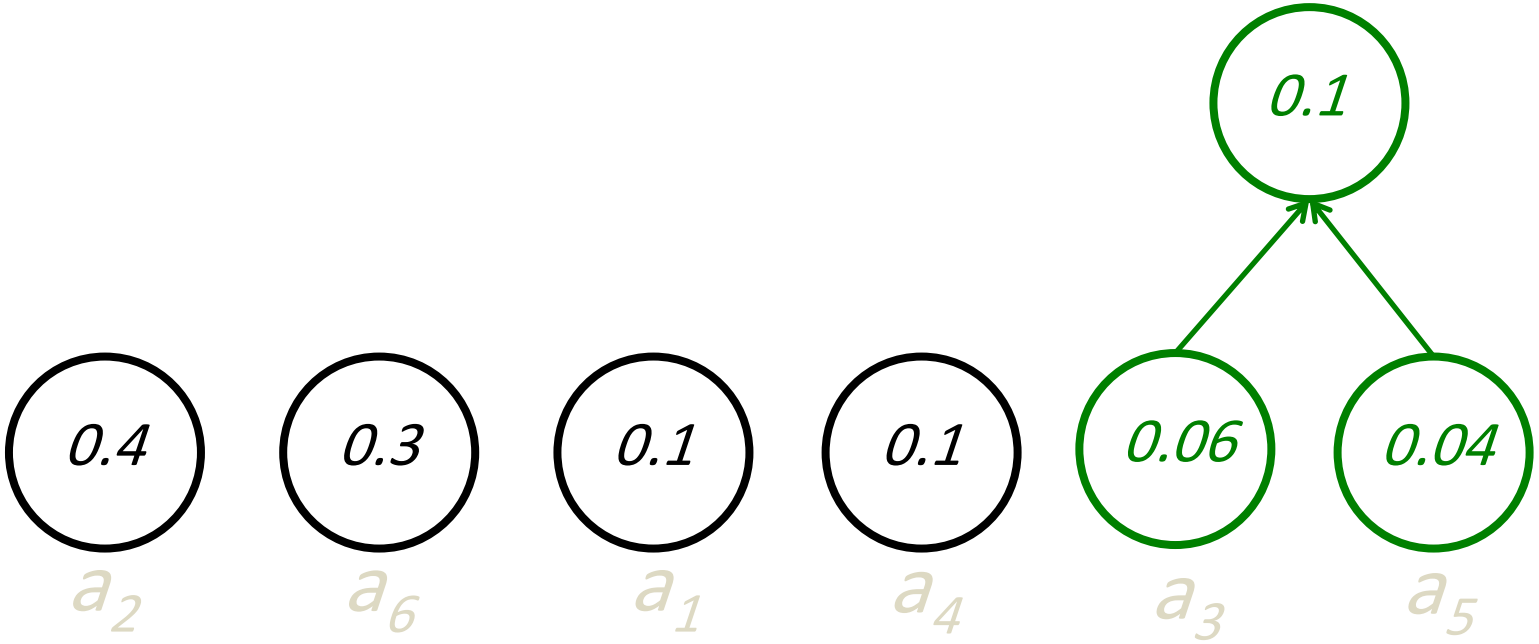
FIGURE 8.8
Huffman code assignment procedure.

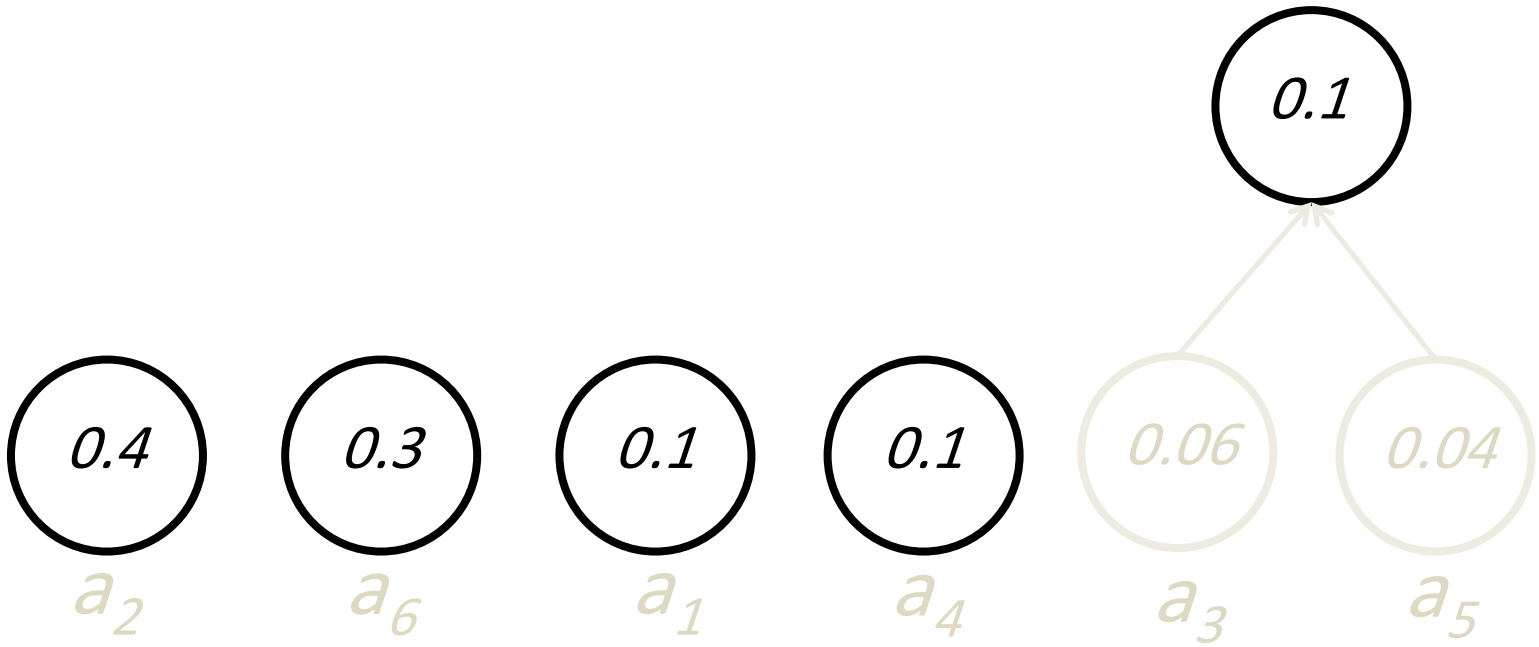
Huffman Coding; Example

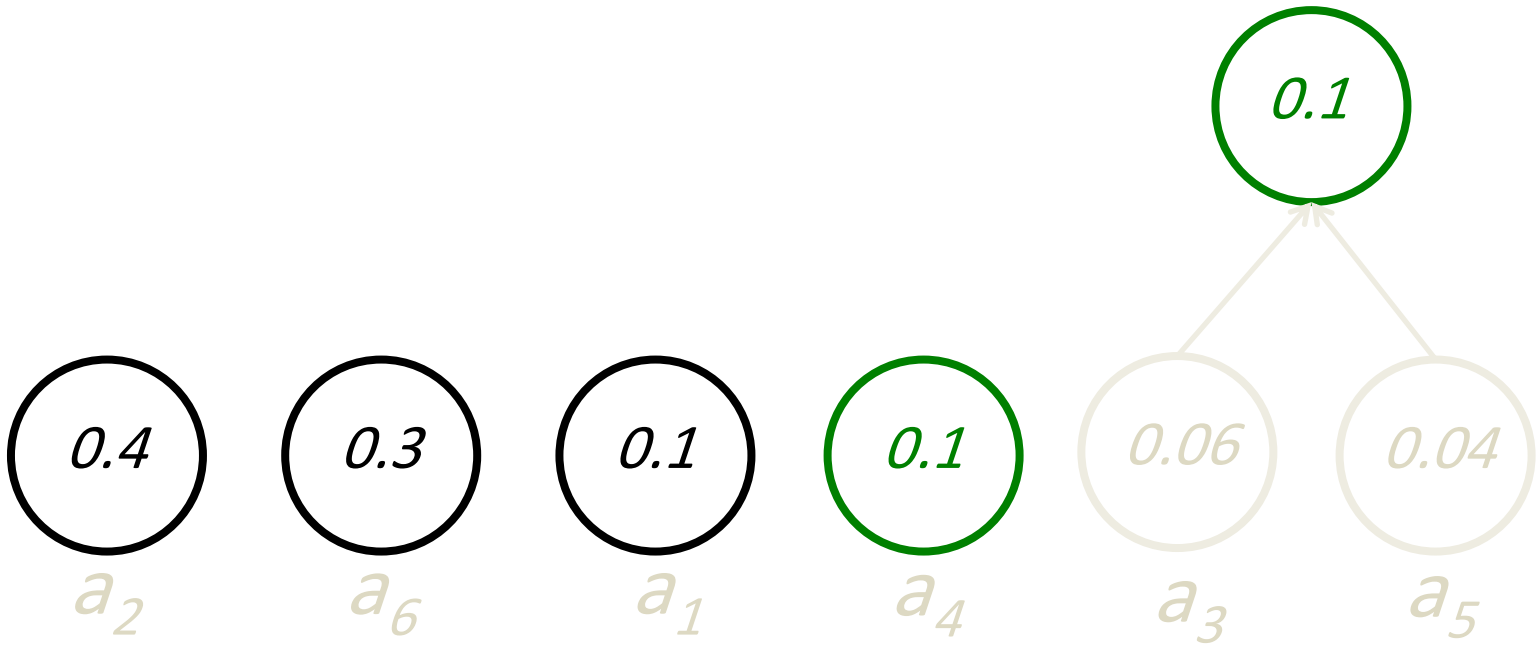
Symbols	Probability
a_1	0.1
a_2	0.4
a_3	0.06
a_4	0.1
a_5	0.04
a_6	0.3

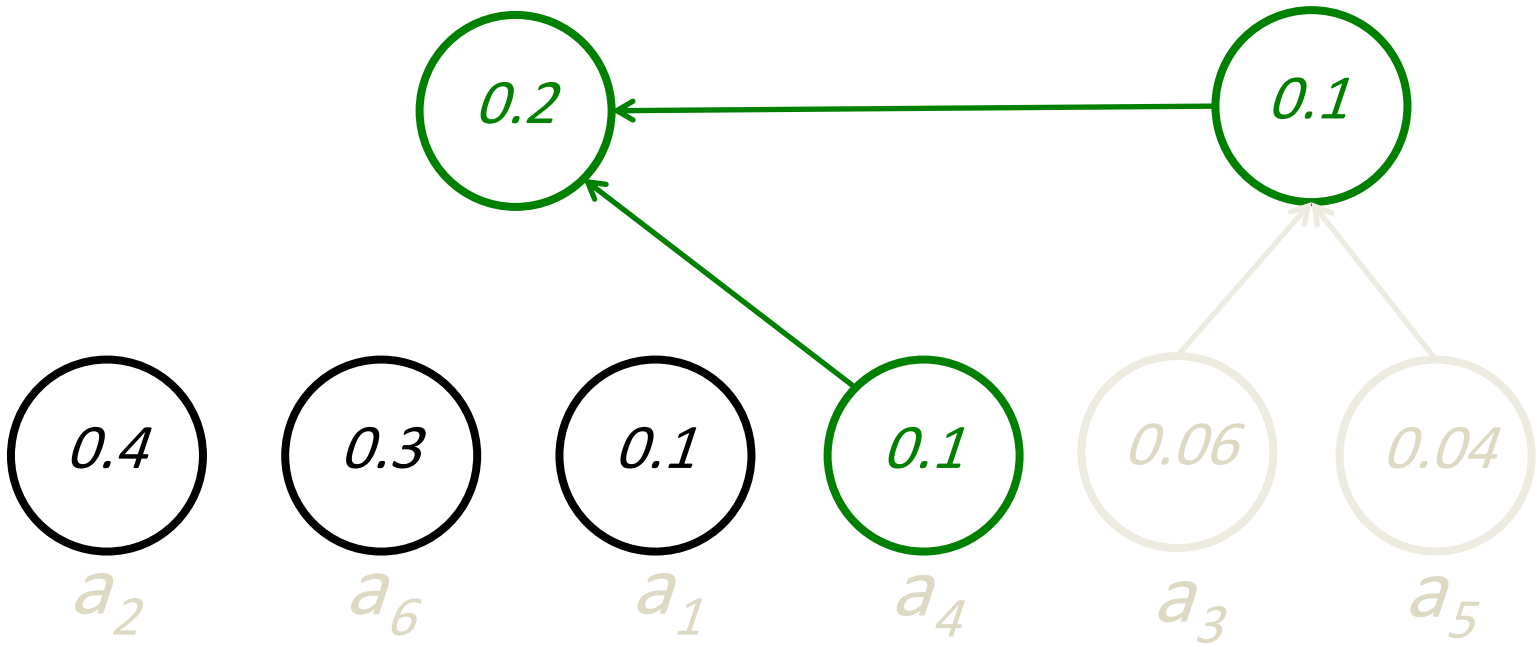

0.4
0.3
0.1
0.1
0.06
0.04
a_2
a_6
a_1
a_4
a_3
a_5

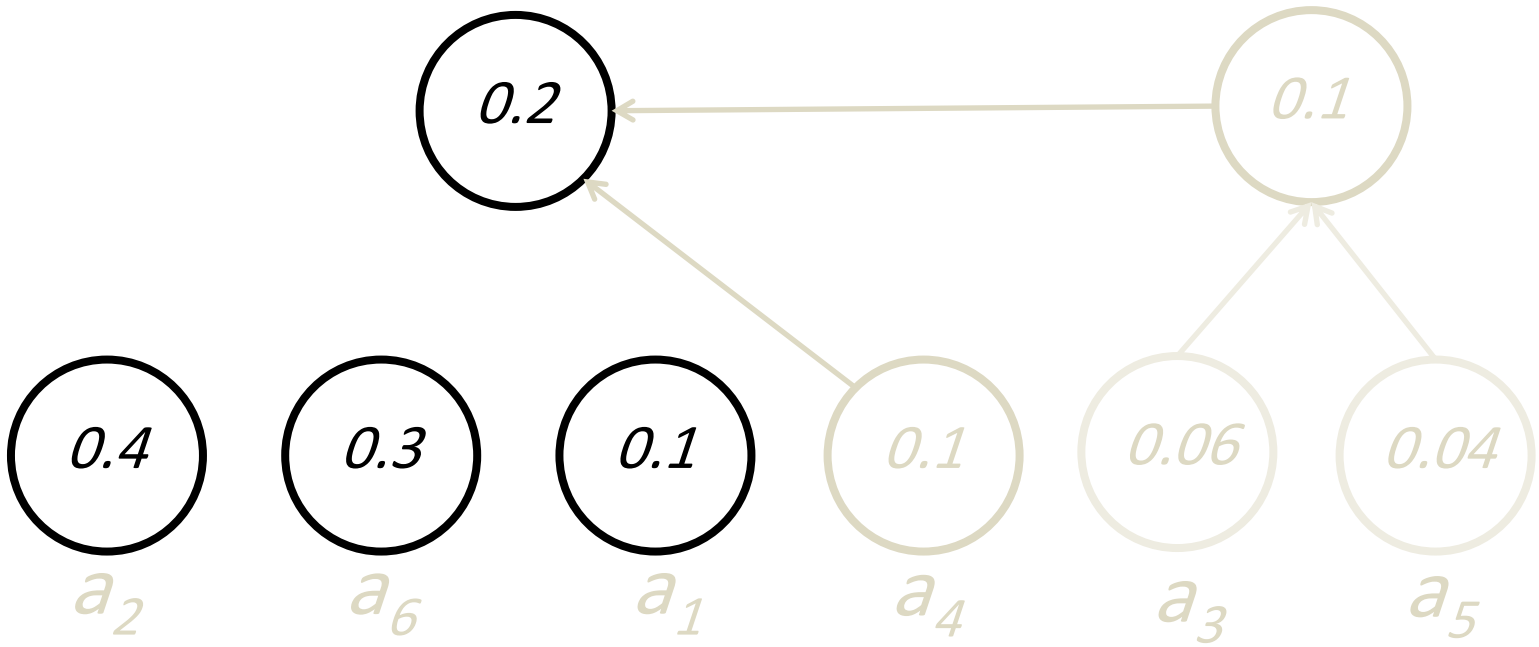

0.4
0.3
0.1
0.1
0.06
0.04
a_2
a_6
a_1
a_4
a_3
a_5

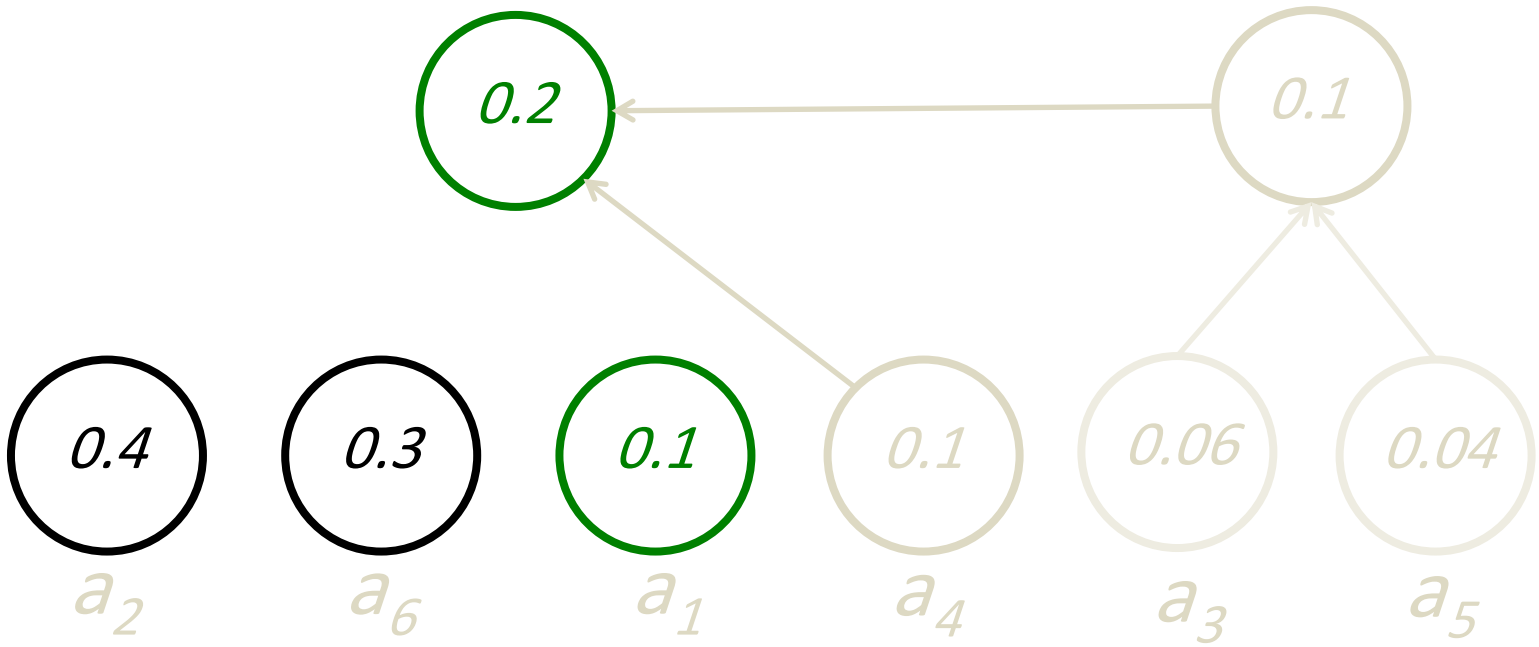

0.1
0.4
0.3
0.1
0.1
0.06
0.04
a_2
a_6
a_1
a_4
a_3
a_5

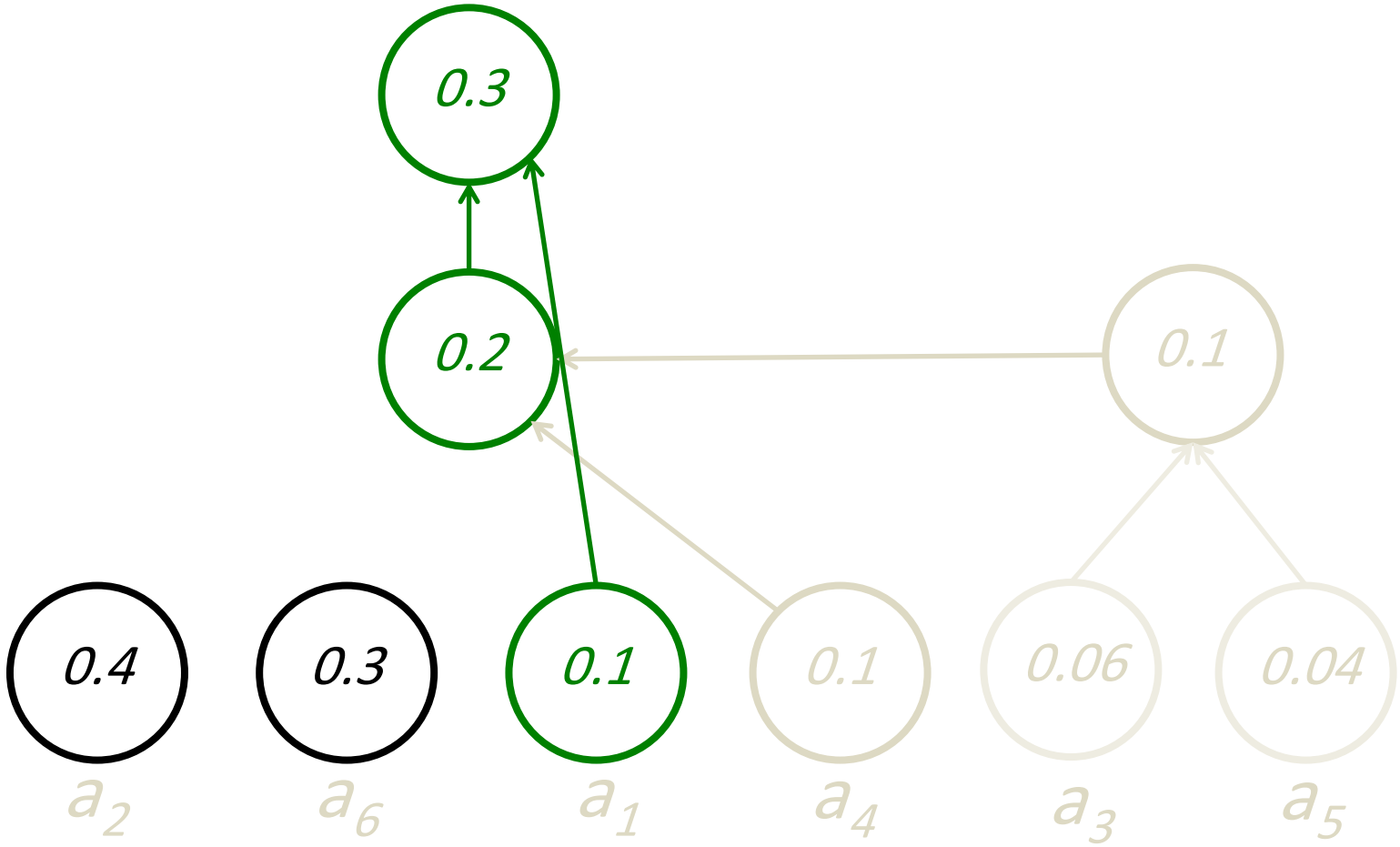

0.1
0.4
0.3
0.1
0.1
0.06
0.04
a_2
a_6
a_1
a_4
a_3
a_5

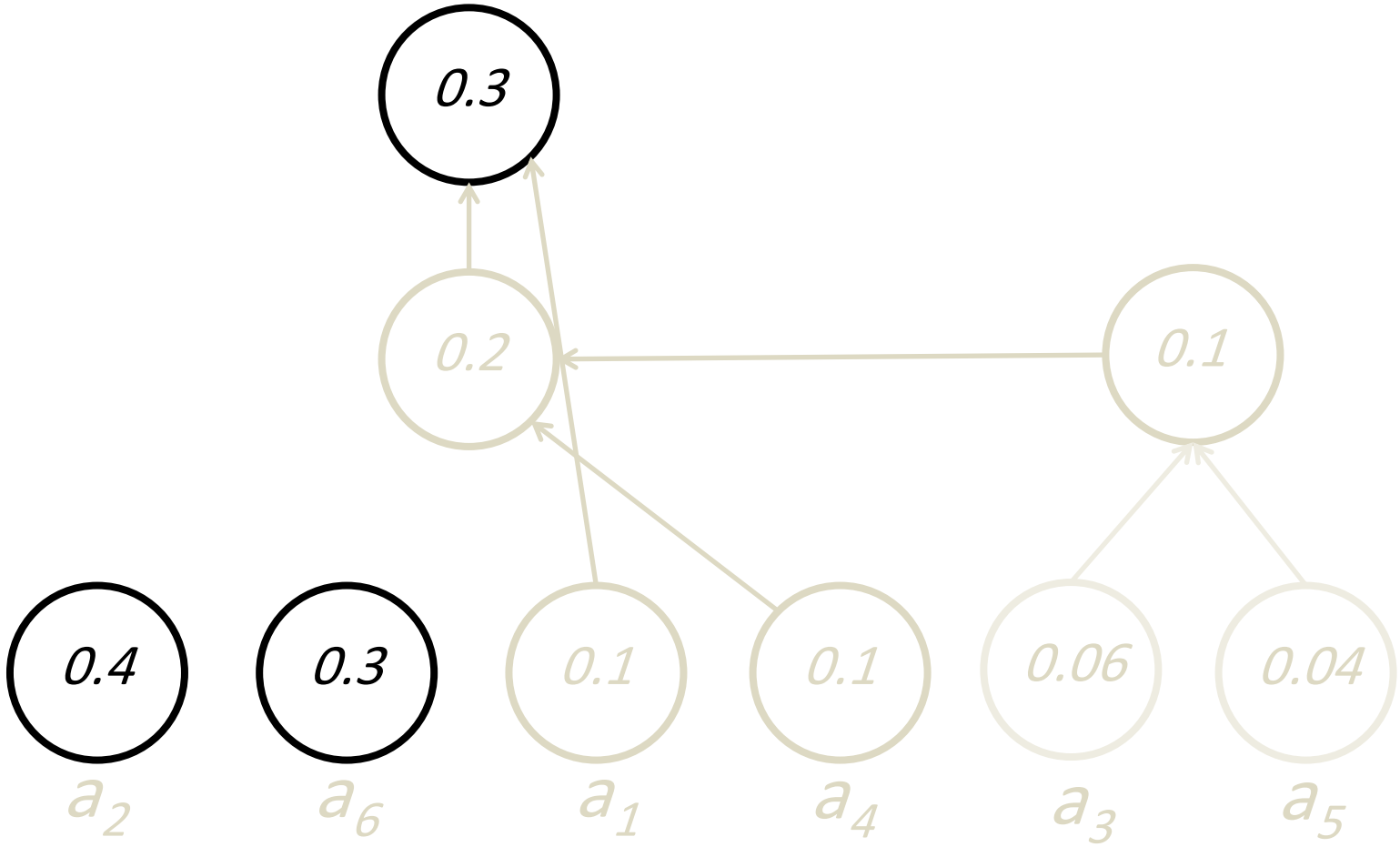

0.1
0.4
0.3
0.1
0.1
0.06
0.04
a_2
a_6
a_1
a_4
a_3
a_5

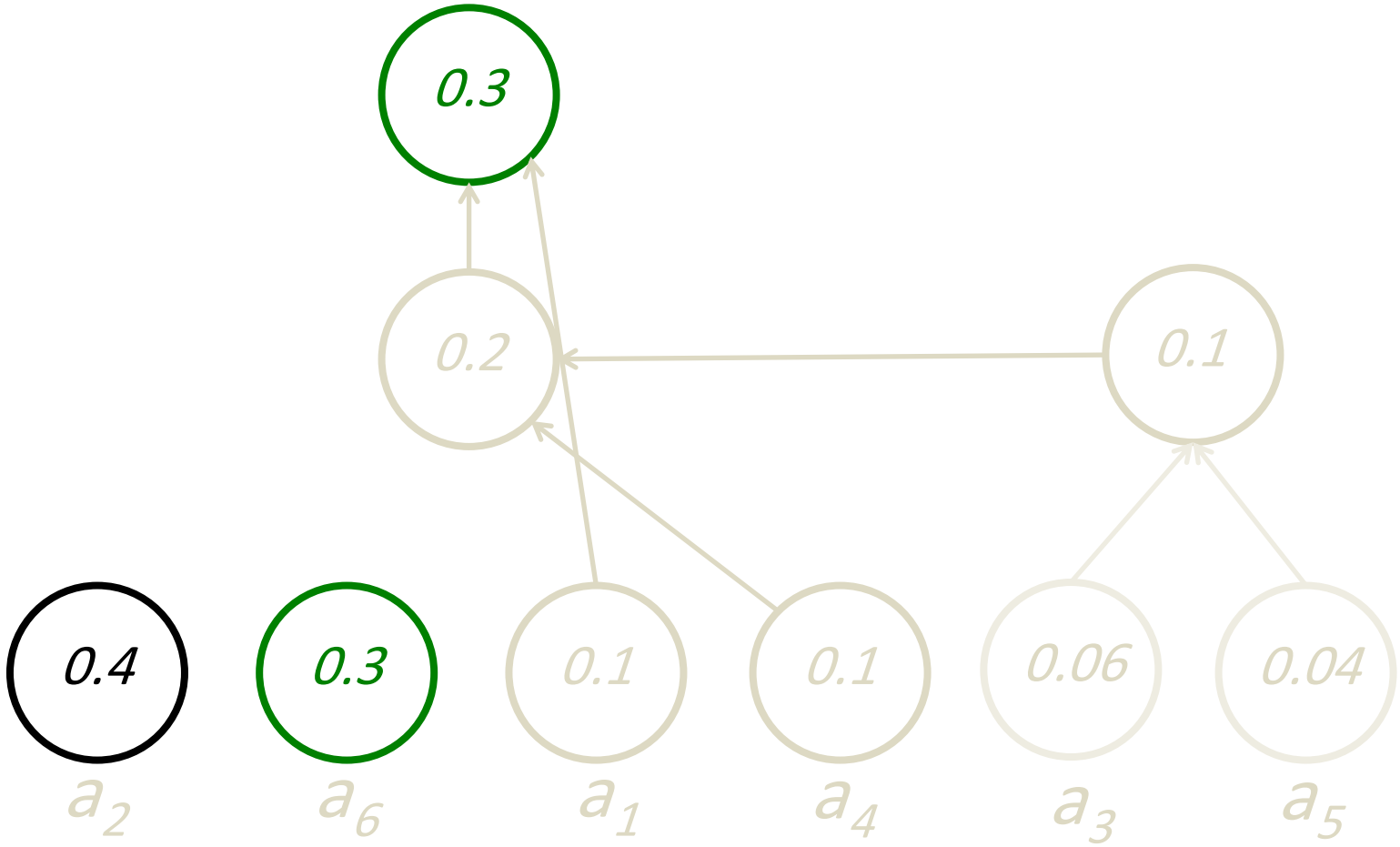

0.2
0.1
0.4
0.3
0.1
0.1
0.06
0.04
a_2
a_6
a_1
a_4
a_3
a_5

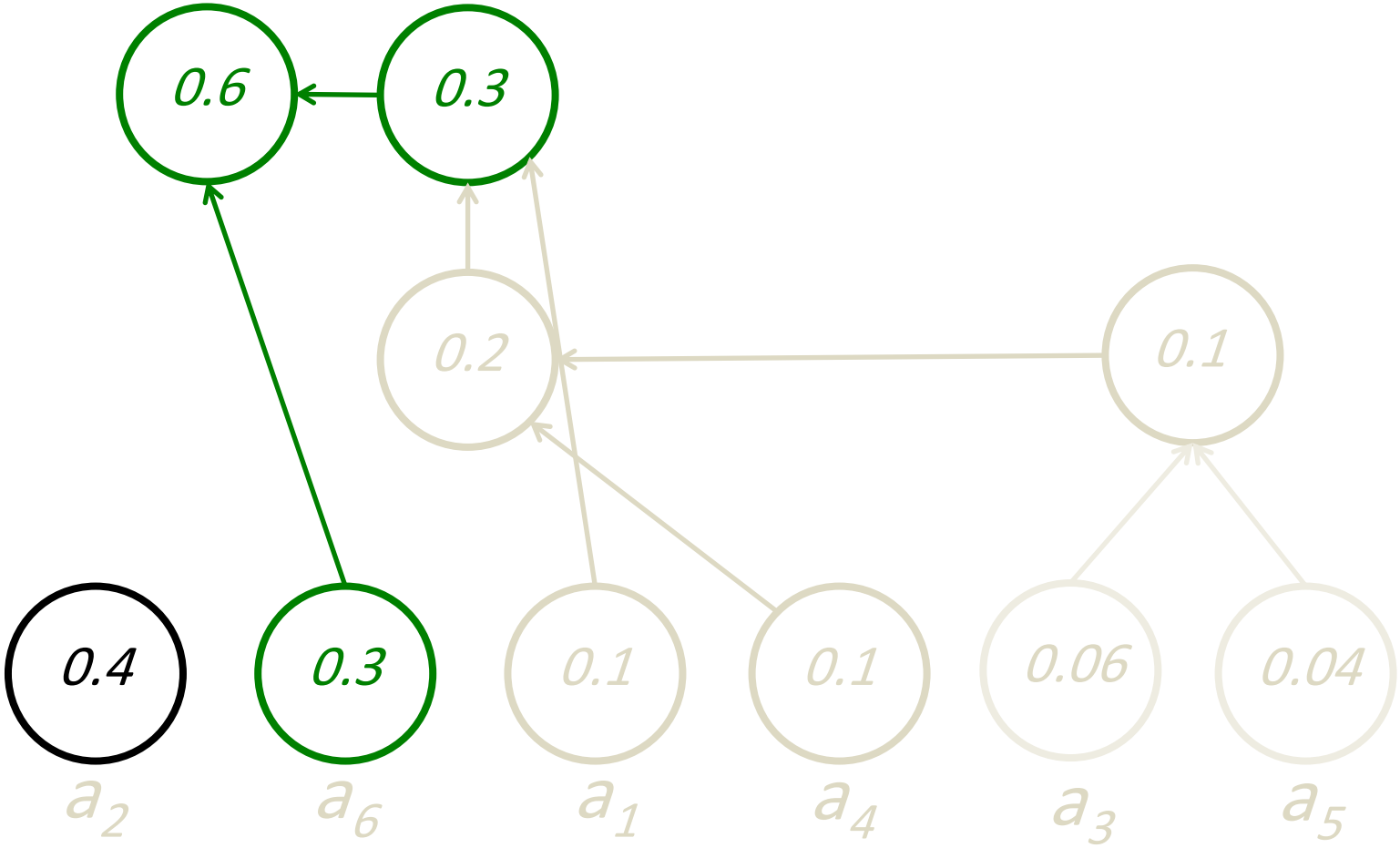

0.2
0.1
0.4
0.3
0.1
0.1
0.06
0.04
a_2
a_6
a_1
a_4
a_3
a_5

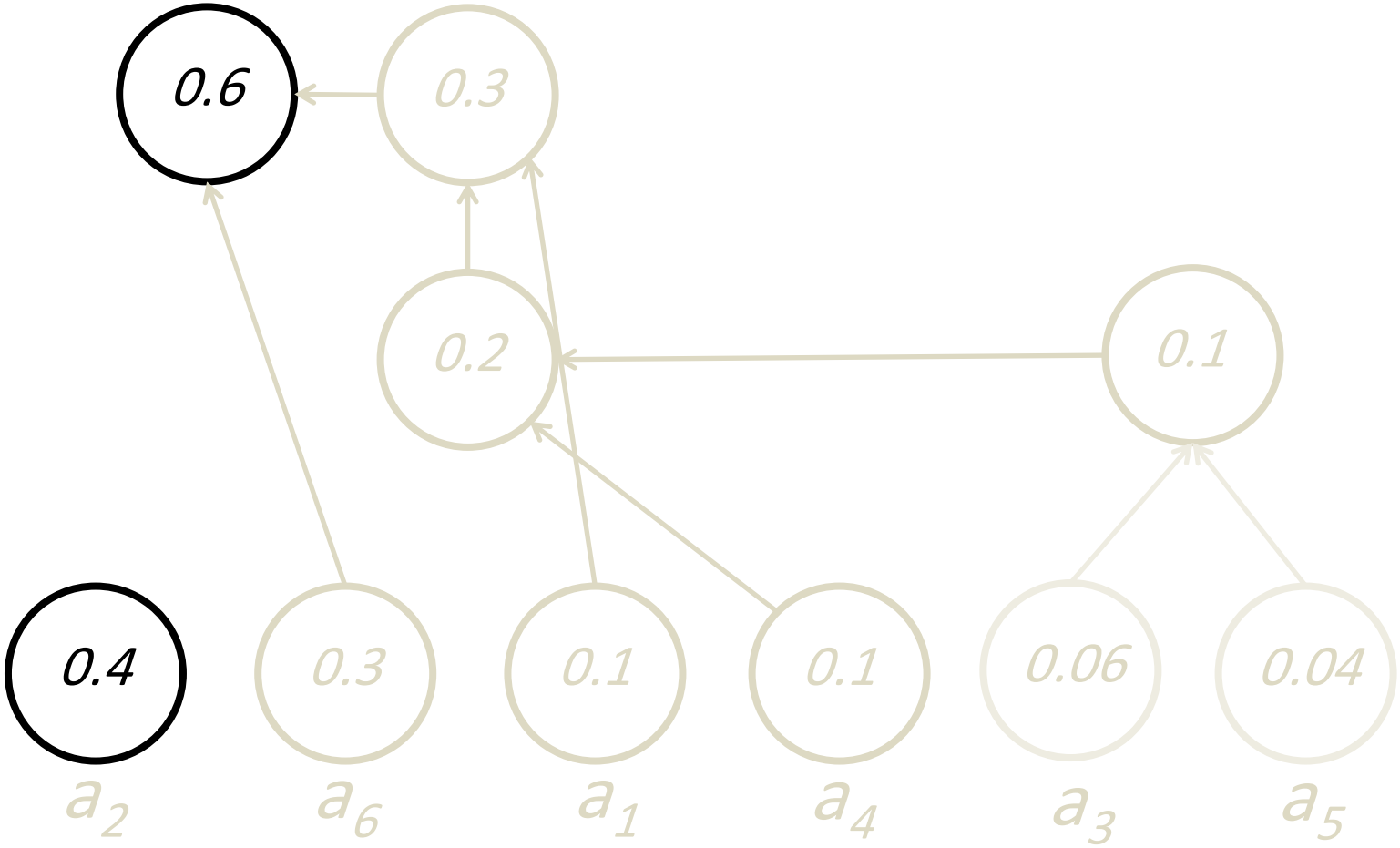

0.2
0.1
0.4
0.3
0.1
0.1
0.06
0.04
a_2
a_6
a_1
a_4
a_3
a_5

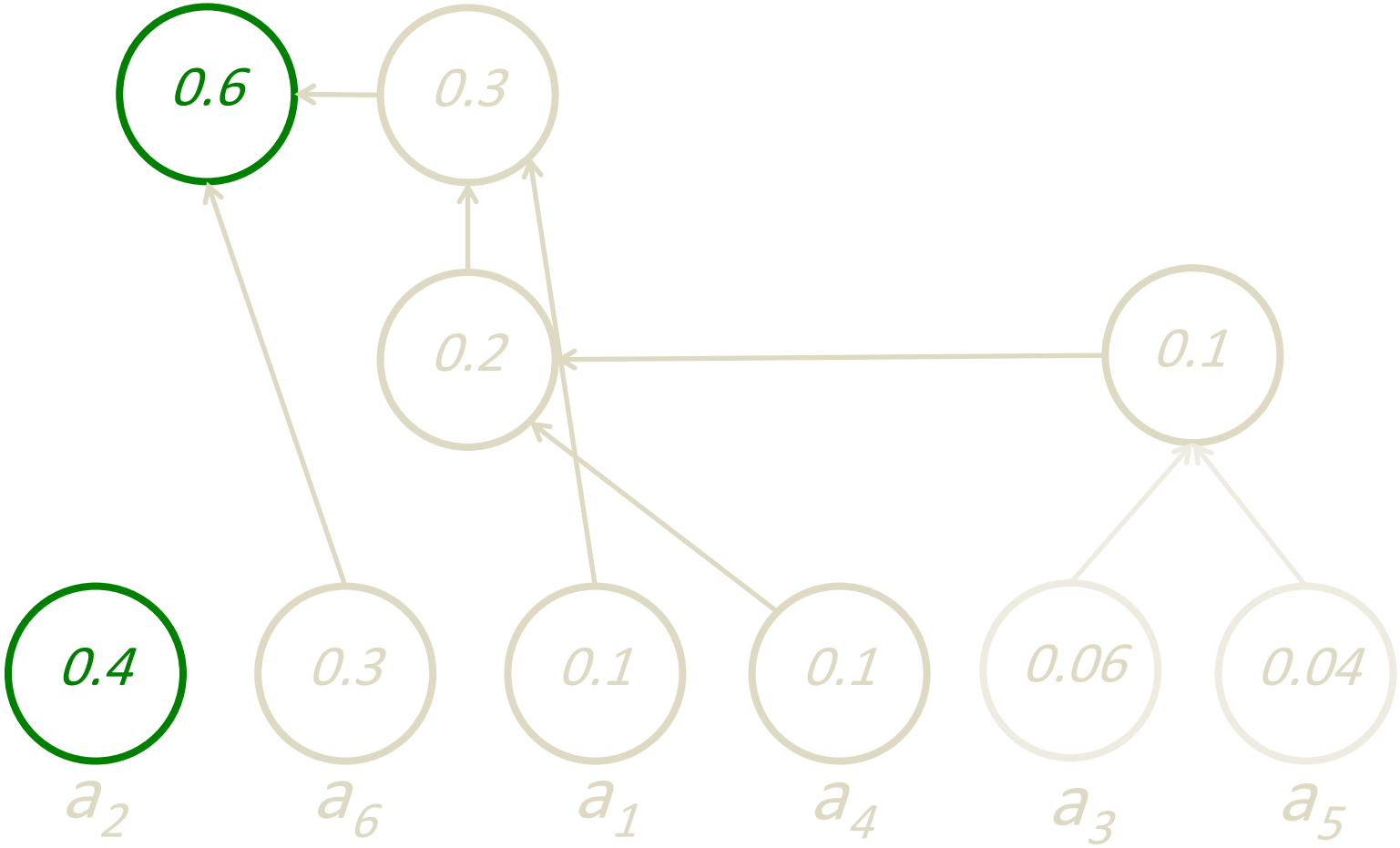

0.3
0.2
0.1
0.4
0.3
0.1
0.1
0.06
0.04
a_2
a_6
a_1
a_4
a_3
a_5

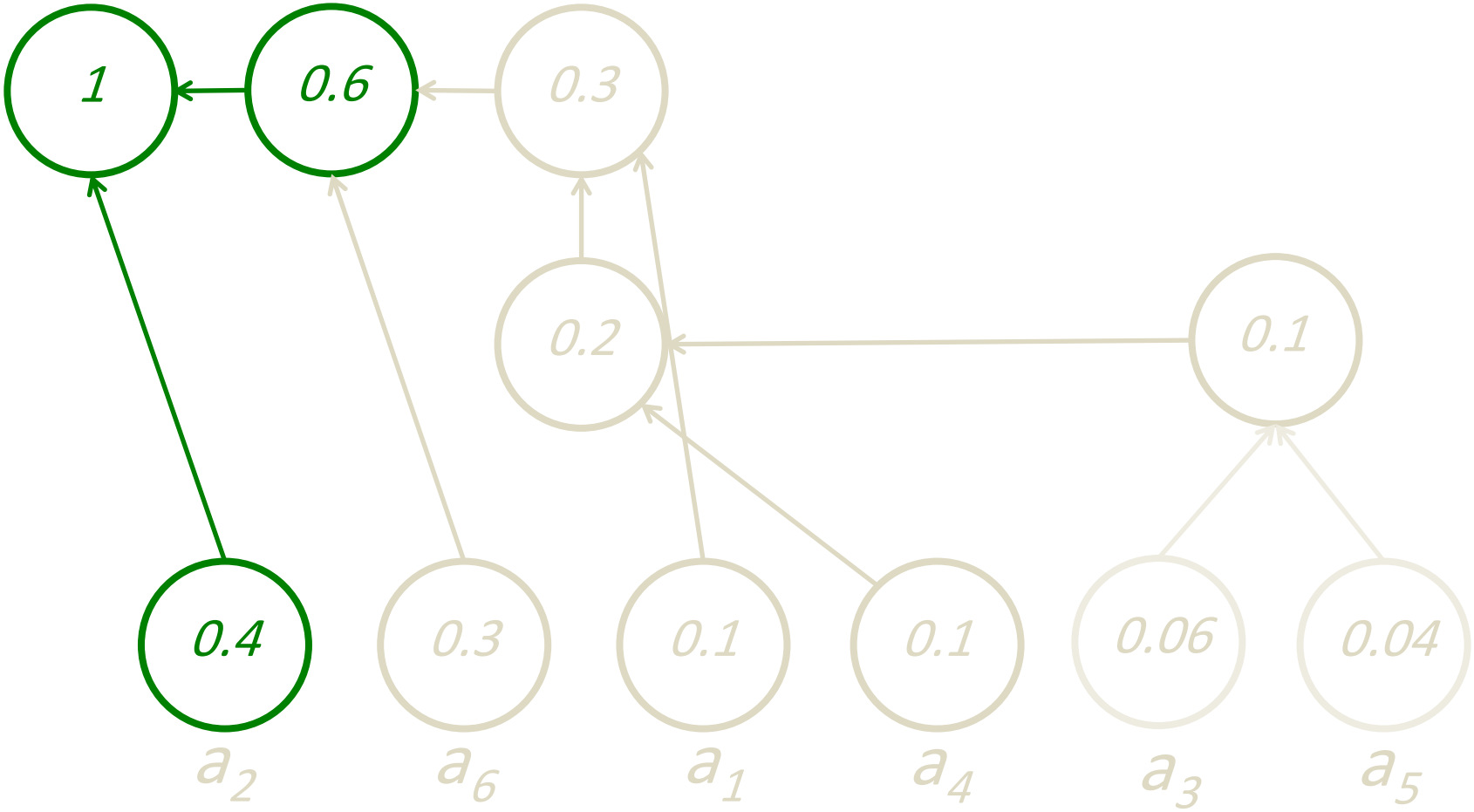

0.3
0.2
0.1
0.4
0.3
0.1
0.1
0.06
0.04
a_2
a_6
a_1
a_4
a_3
a_5

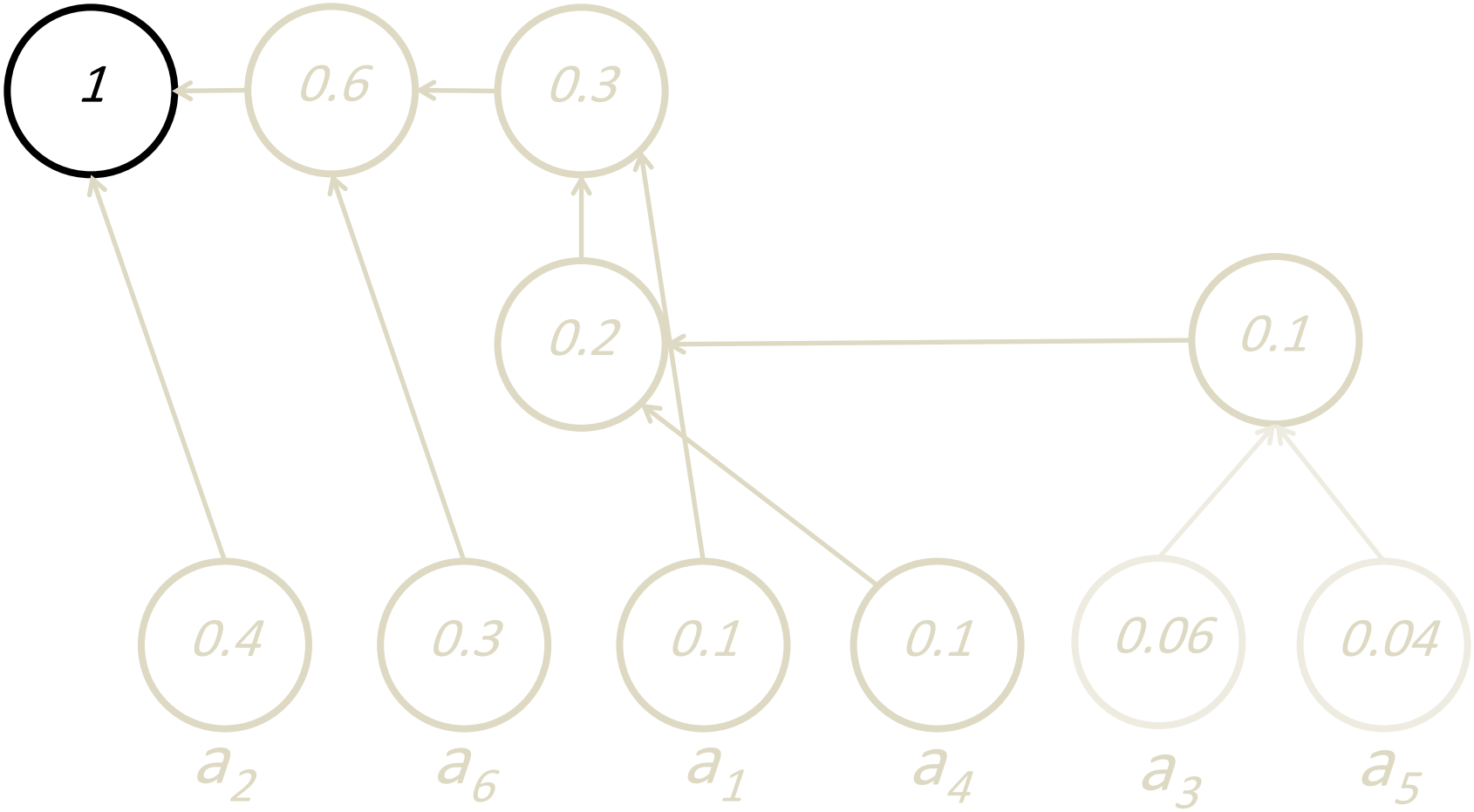

0.3
0.2
0.1
0.4
0.3
0.1
0.1
0.06
0.04
a_2
a_6
a_1
a_4
a_3
a_5

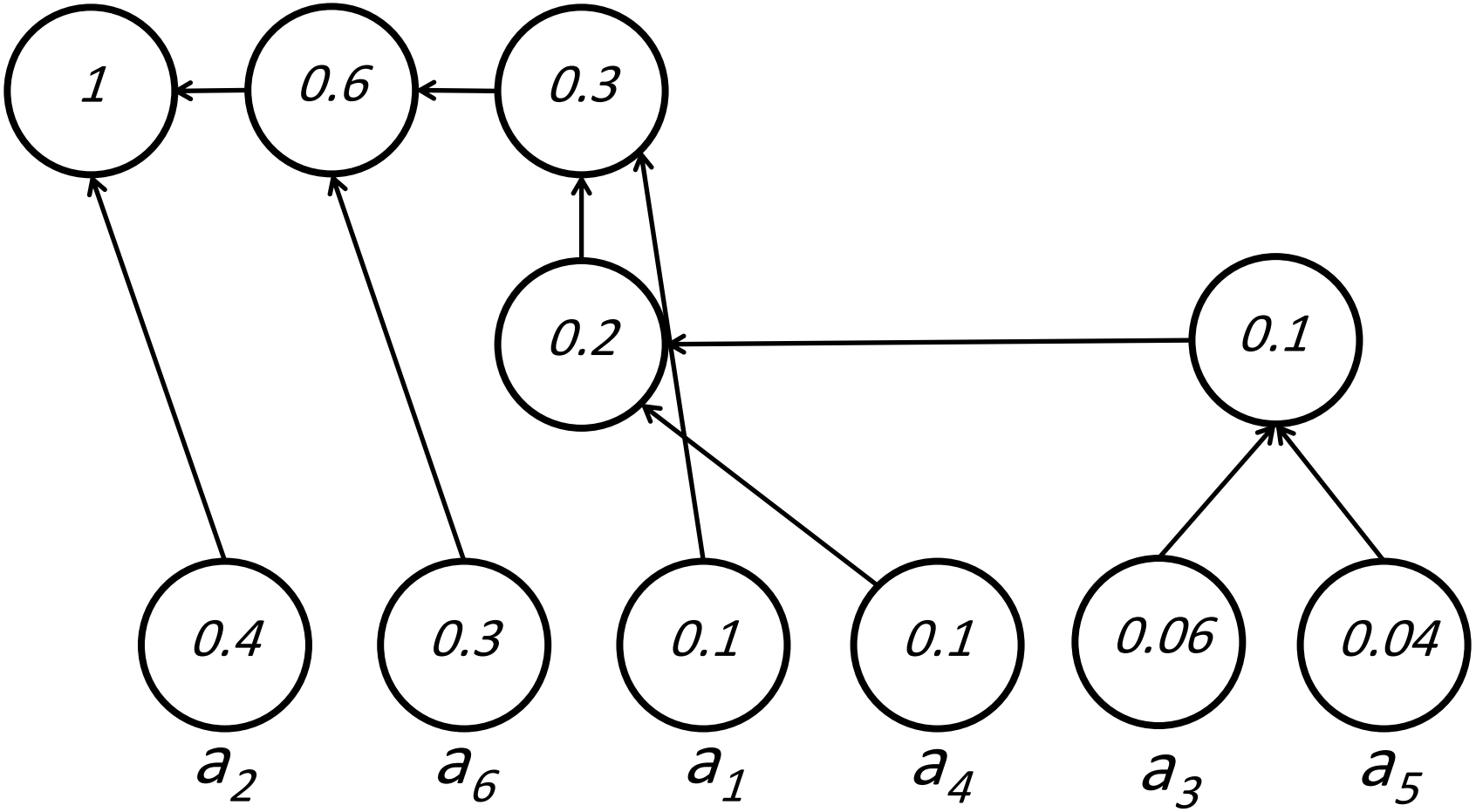

0.6
0.3
0.2
0.1
0.4
0.3
0.1
0.1
0.06
0.04
a2
a6
a1
a4
a3
a5

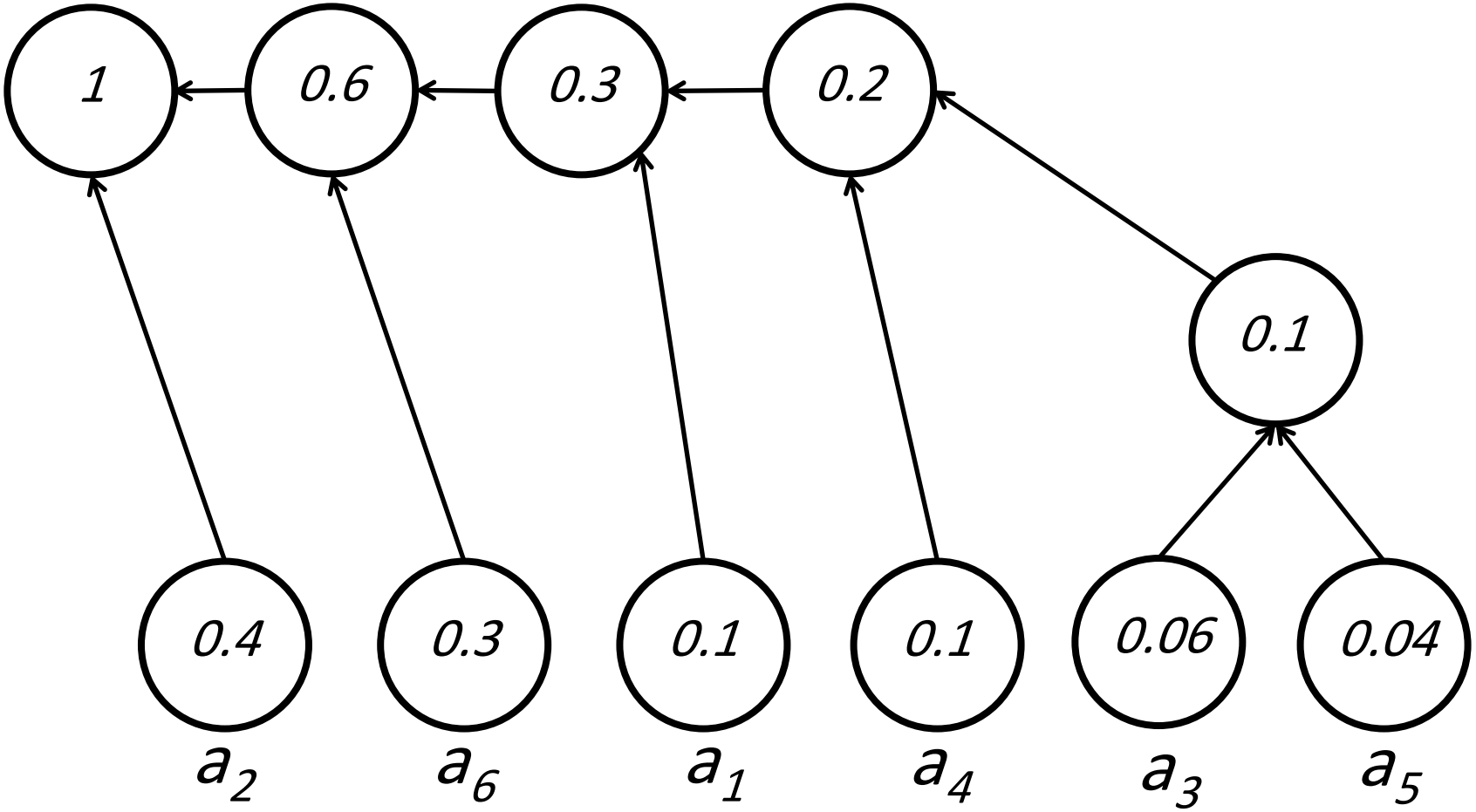

0.6
0.3
0.2
0.1
0.4
0.3
0.1
0.1
0.06
0.04
a_2
a_6
a_1
a_4
a_3
a_5

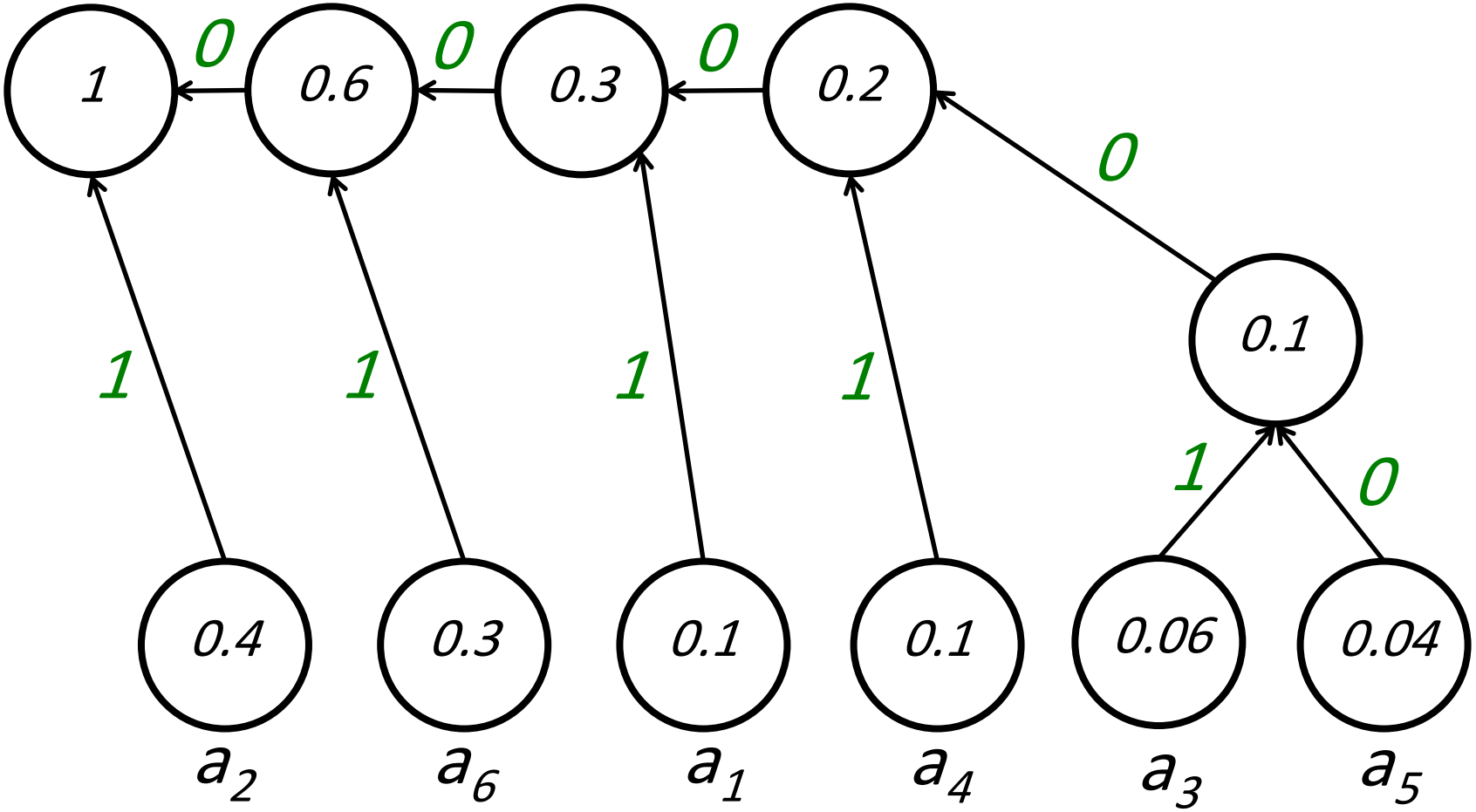

0.6
0.3
0.2
0.1
0.4
0.3
0.1
0.1
0.06
0.04
a_2
a_6
a_1
a_4
a_3
a_5

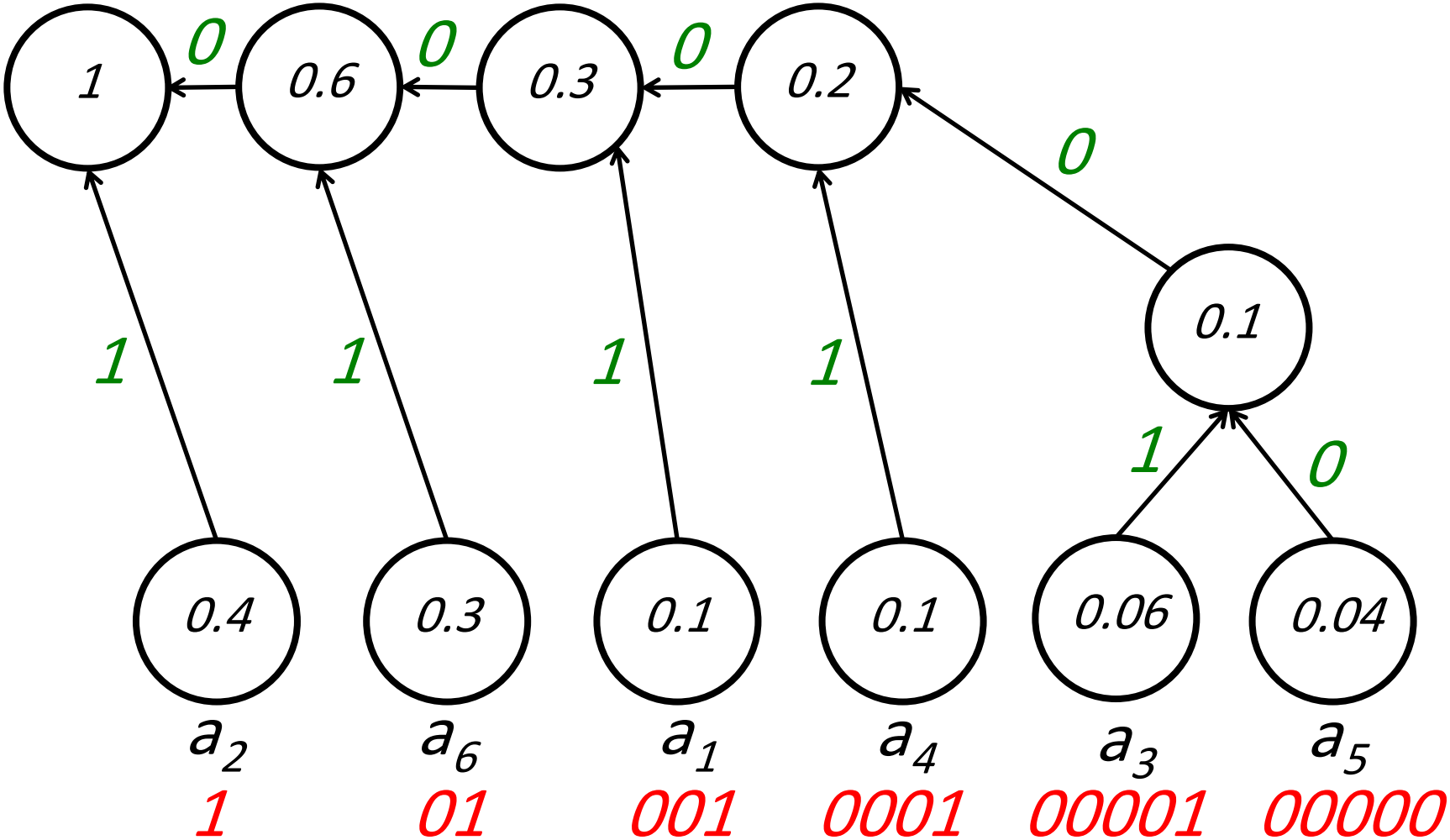

1
0.6
0.3
0.2
0.1
0.4
0.3
0.1
0.1
0.06
0.04
a_2
a_6
a_1
a_4
a_3
a_5


1
0.6
0.3
0.2
0.1
0.4
0.3
0.1
0.1
0.06
0.04
a_2
a_6
a_1
a_4
a_3
a_5


1
0.6
0.3
0.2
0.1
0.4
0.3
0.1
0.1
0.06
0.04
a_2
a_6
a_1
a_4
a_3
a_5


1
0.6
0.3
0.2
0.1
0.4
0.3
0.1
0.1
0.06
0.04
a_2
a_6
a_1
a_4
a_3
a_5


1
0.6
0.3
0.2
0.1
0
0
0
0
1
1
1
1
1
0
0.4
0.3
0.1
0.1
0.06
0.04
a_2
a_6
a_1
a_4
a_3
a_5


1
0.6
0.3
0.2
0.1
0.4
0.3
0.1
0.1
0.06
0.04
0
0
0
0
1
1
1
1
1
0
a_2
a_6
a_1
a_4
a_3
a_5
1
01
001
0001
00001
00000

Huffman Coding; Example

Symbols	Probability	Code
a_1	0.1	001
a_2	0.4	1
a_3	0.06	00001
a_4	0.1	0001
a_5	0.04	00000
a_6	0.3	01

Huffman Coding; Example

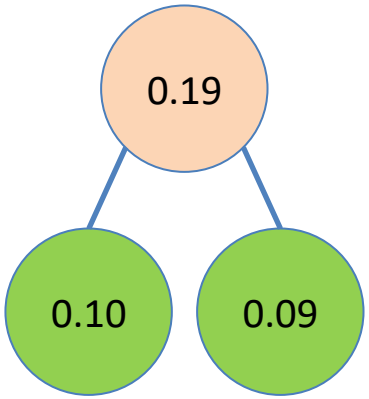
Symbols	Probability	Code	Code Length
a_2	0.4	1	1
a_6	0.3	01	2
a_1	0.1	001	3
a_4	0.1	0001	4
a_3	0.06	00001	5
a_5	0.04	00000	5

Average Length:

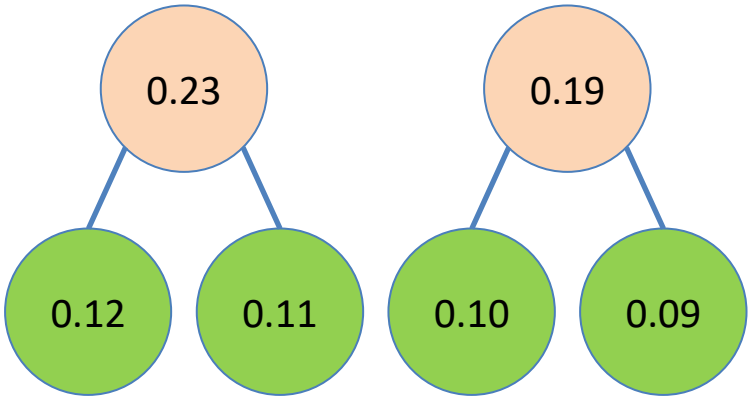
$$0.4 \times 1 + 0.3 \times 2 + 0.1 \times 3 + 0.1 \times 4 + 0.06 \times 5 + 0.04 \times 5 = 2.2$$

a_1	0.09
a_2	0.10
a_3	0.11
a_4	0.12
a_5	0.13
a_6	0.14
a_7	0.15
a_8	0.16

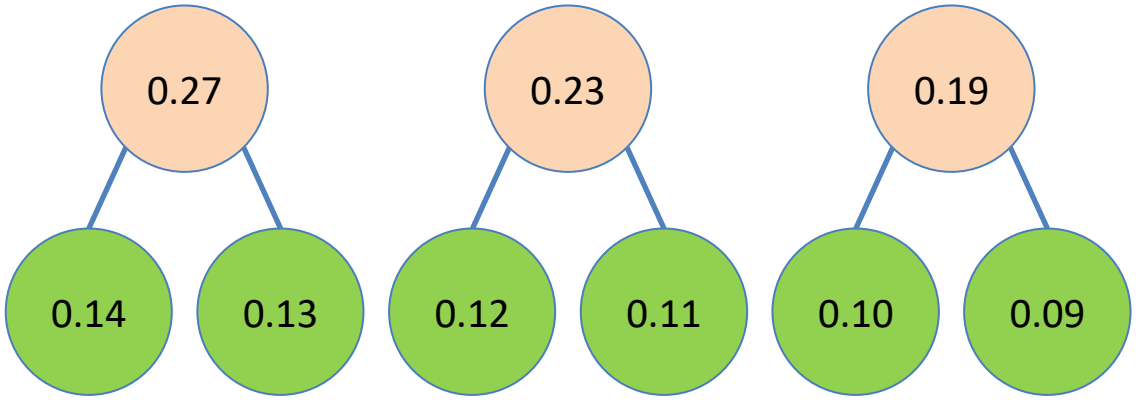
a_1	0.09	0.11
a_2	0.10	0.12
a_3	0.11	0.13
a_4	0.12	0.14
a_5	0.13	0.15
a_6	0.14	0.16
a_7	0.15	0.19
a_8	0.16	



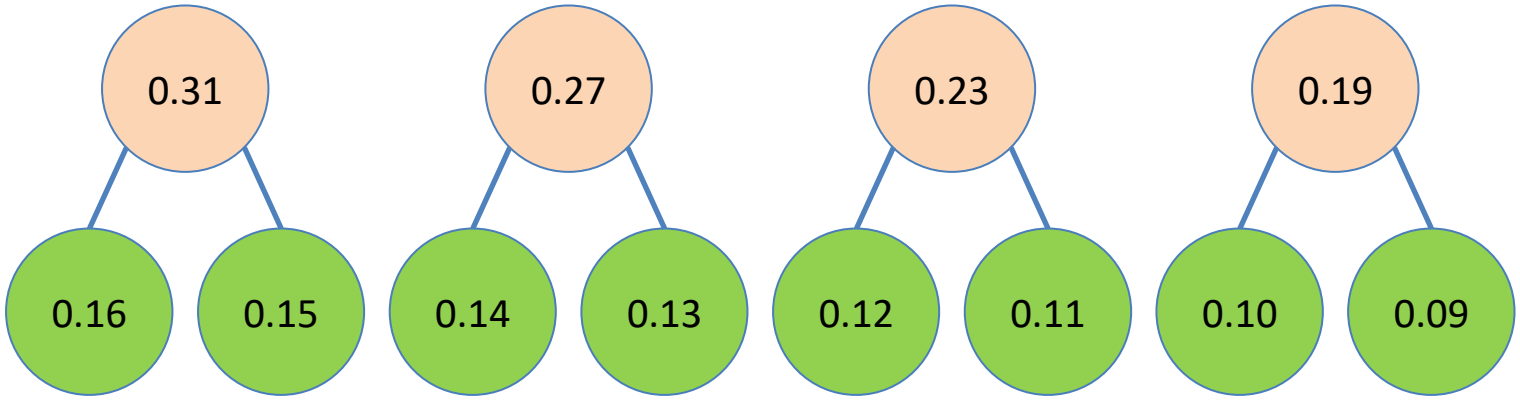
a_1	0.09	0.11	0.13
a_2	0.10	0.12	0.14
a_3	0.11	0.13	0.15
a_4	0.12	0.14	0.16
a_5	0.13	0.15	0.19
a_6	0.14	0.16	0.23
a_7	0.15	0.19	
a_8	0.16		



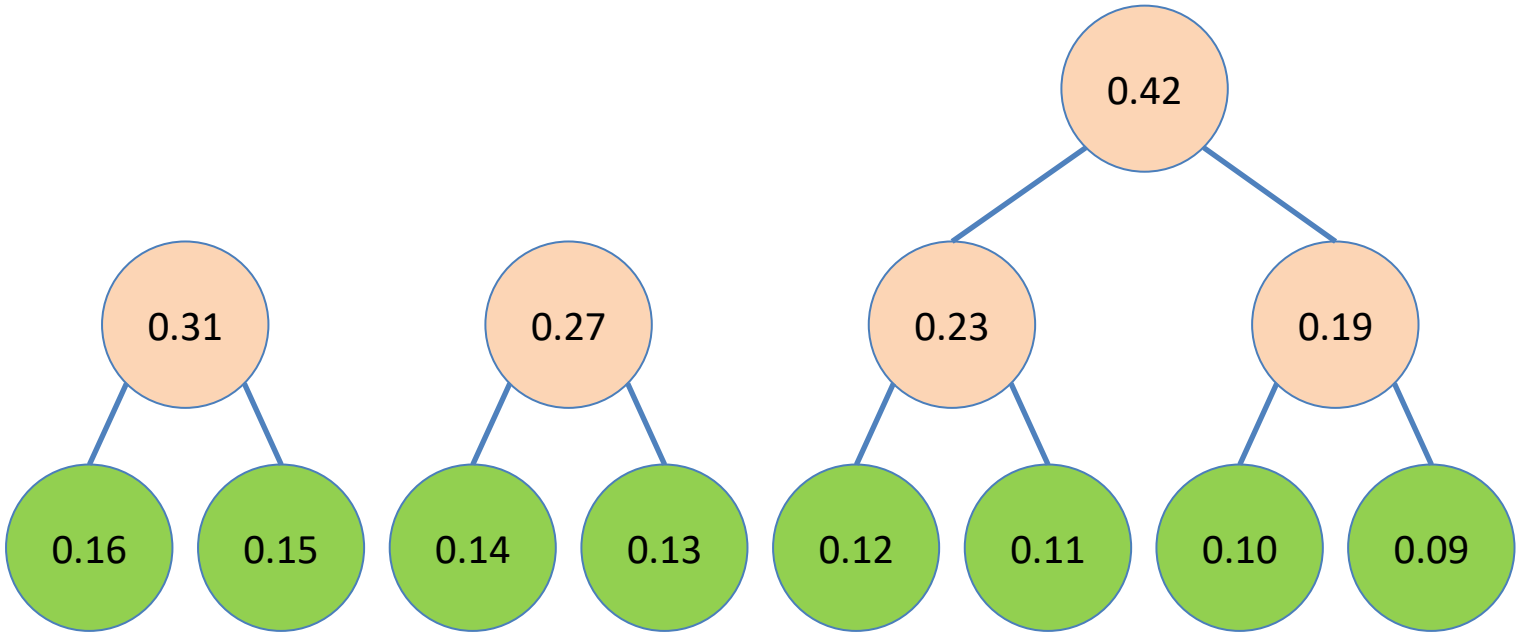
a_1	0.09	0.11	0.13	0.15
a_2	0.10	0.12	0.14	0.16
a_3	0.11	0.13	0.15	0.19
a_4	0.12	0.14	0.16	0.23
a_5	0.13	0.15	0.19	0.27
a_6	0.14	0.16	0.23	
a_7	0.15	0.19		
a_8	0.16			



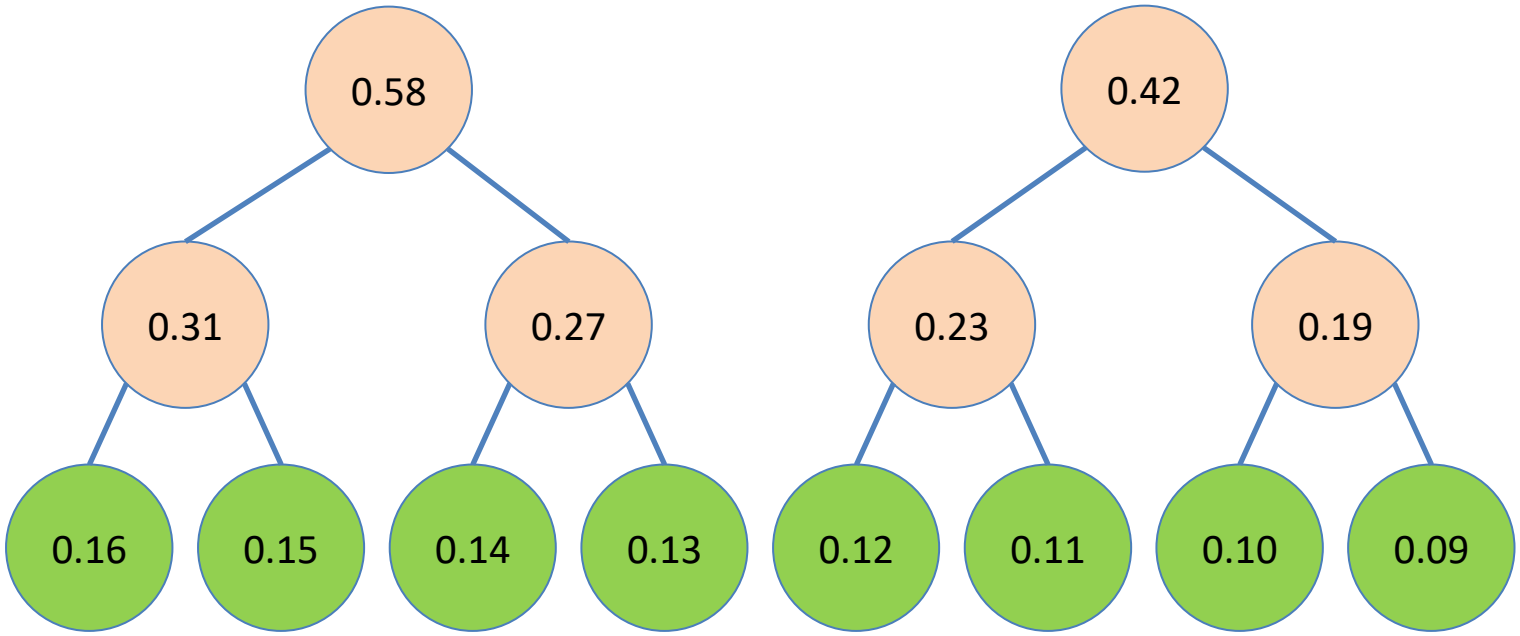
a_1	0.09	0.11	0.13	0.15	0.19
a_2	0.10	0.12	0.14	0.16	0.23
a_3	0.11	0.13	0.15	0.19	0.27
a_4	0.12	0.14	0.16	0.23	0.31
a_5	0.13	0.15	0.19	0.27	
a_6	0.14	0.16	0.23		
a_7	0.15	0.19			
a_8	0.16				



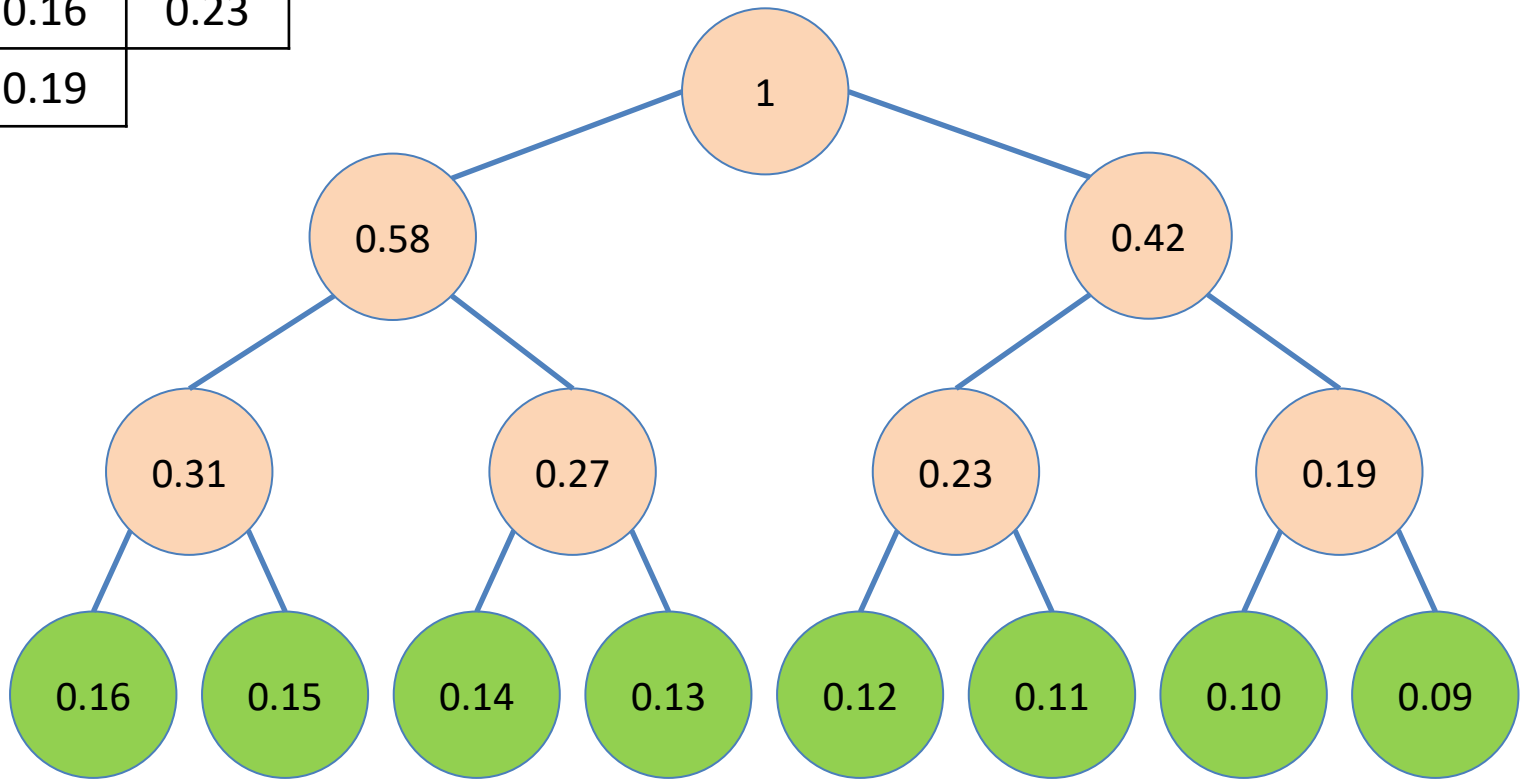
a_1	0.09	0.11	0.13	0.15	0.19	0.27
a_2	0.10	0.12	0.14	0.16	0.23	0.31
a_3	0.11	0.13	0.15	0.19	0.27	0.42
a_4	0.12	0.14	0.16	0.23	0.31	
a_5	0.13	0.15	0.19	0.27		
a_6	0.14	0.16	0.23			
a_7	0.15	0.19				
a_8	0.16					



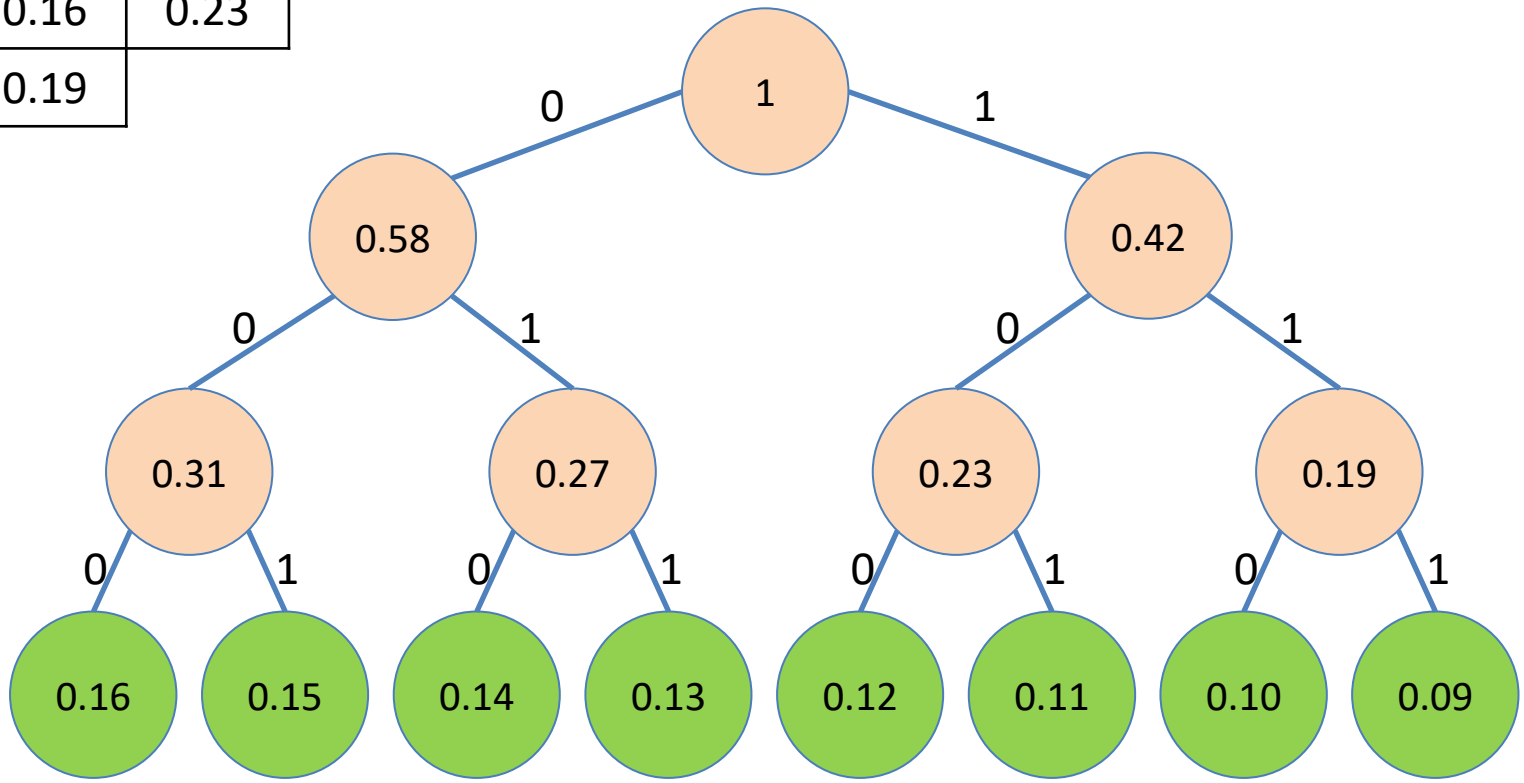
a_1	0.09	0.11	0.13	0.15	0.19	0.27	0.42
a_2	0.10	0.12	0.14	0.16	0.23	0.31	0.58
a_3	0.11	0.13	0.15	0.19	0.27	0.42	
a_4	0.12	0.14	0.16	0.23	0.31		
a_5	0.13	0.15	0.19	0.27			
a_6	0.14	0.16	0.23				
a_7	0.15	0.19					
a_8	0.16						



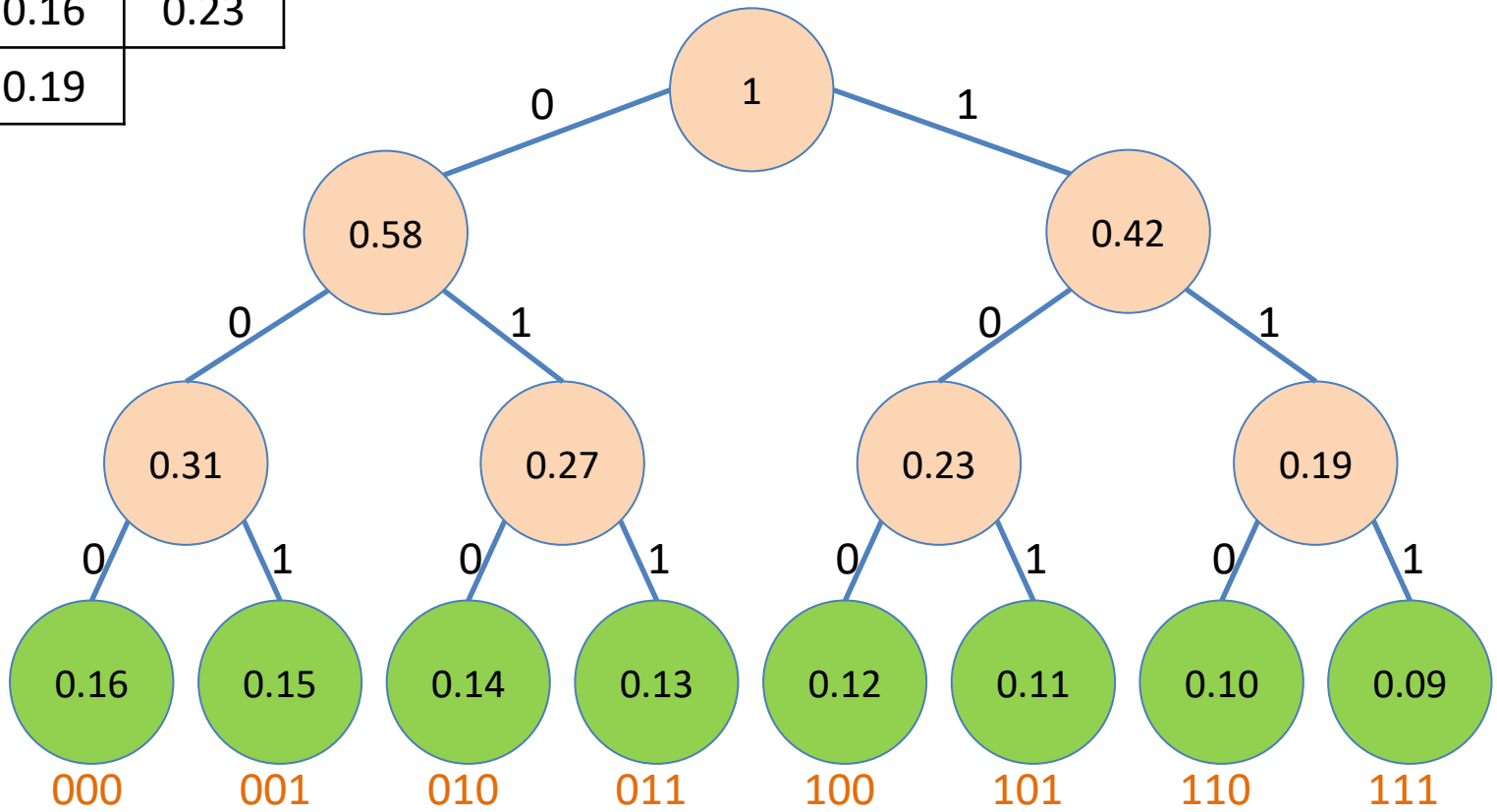
a_1	0.09	0.11	0.13	0.15	0.19	0.27	0.42	1
a_2	0.10	0.12	0.14	0.16	0.23	0.31	0.58	
a_3	0.11	0.13	0.15	0.19	0.27	0.42		
a_4	0.12	0.14	0.16	0.23	0.31			
a_5	0.13	0.15	0.19	0.27				
a_6	0.14	0.16	0.23					
a_7	0.15	0.19						
a_8	0.16							



a_1	0.09	0.11	0.13	0.15	0.19	0.27	0.42	1
a_2	0.10	0.12	0.14	0.16	0.23	0.31	0.58	
a_3	0.11	0.13	0.15	0.19	0.27	0.42		
a_4	0.12	0.14	0.16	0.23	0.31			
a_5	0.13	0.15	0.19	0.27				
a_6	0.14	0.16	0.23					
a_7	0.15	0.19						
a_8	0.16							



a_1	0.09	0.11	0.13	0.15	0.19	0.27	0.42	1
a_2	0.10	0.12	0.14	0.16	0.23	0.31	0.58	
a_3	0.11	0.13	0.15	0.19	0.27	0.42		
a_4	0.12	0.14	0.16	0.23	0.31			
a_5	0.13	0.15	0.19	0.27				
a_6	0.14	0.16	0.23					
a_7	0.15	0.19						
a_8	0.16							



Information vs. Data

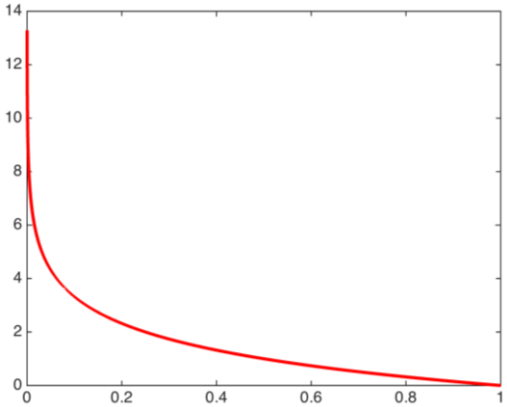
Variables: $a_1,\ a_2, \dots,\ a_{256}$

If we know nothing: $p(a_i) = \frac{1}{256}$

If we know histogram h: $p(a_i \mid h) = h(i)$

How much information each variable contains:

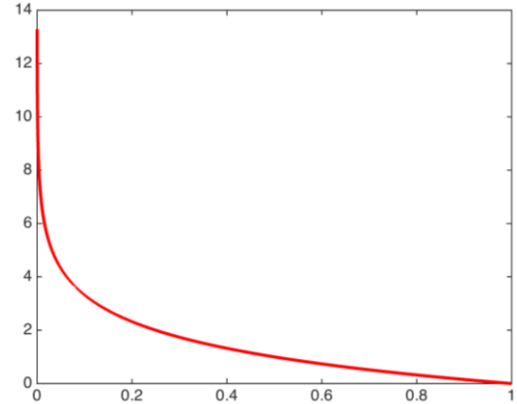
$$I(a_i) = \log_2 \frac{1}{p(a_i)} = -\log_2 p(a_i)$$



Self Information / Units of Information

Self Information / Units of Information

$$I(a_i) = -\log_2 p(a_i) \longrightarrow$$



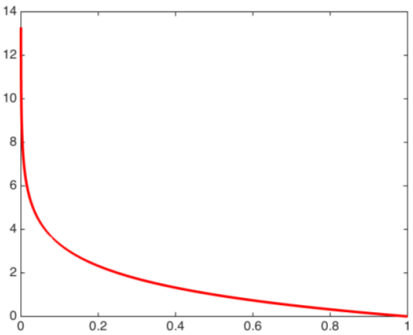
A symbol with high probability gives not much information when it happens.

Entropy: $H = -\sum_{i=1}^{n} p(a_i) \log_2 p(a_i)$

<u>Shannon's theorem:</u> Entropy is the minimum average code word length per source symbol that can be achieved.

Entropy of the previous example $= 2.1435$
Average Length Obtained $= 2.2$

$$H = -\sum_{i=1}^{n} p(a_i)\log_2 p(a_i) = \sum_{i=1}^{n} p(a_i)\log_2 \frac{1}{p(a_i)}$$



Entropy: $\mathrm{H} = 0.4 \times 1.3219 + 0.3 \times 1.7370 + 0.1 \times 3.3219 + 0.1 \times 3.3219 + 0.06 \times 4.0589 + 0.04 \times 4.6439 = 2.1435$

Average Length in Huffman Coding:

$0.4 \times 1 + 0.3 \times 2 + 0.1 \times 3 + 0.1 \times 4 + 0.06 \times 5 + 0.04 \times 5 = 2.2$

Symbols	Probability	Code	Code Length
a_2	0.4	1	1
a_6	0.3	01	2
a_1	0.1	001	3
a_4	0.1	0001	4
a_3	0.06	00001	5
a_5	0.04	00000	5

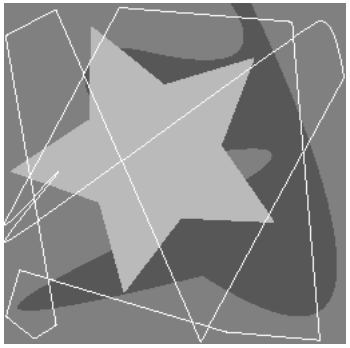
Entropy = 6.8198

Average Length in Huffman Coding = 6.8656

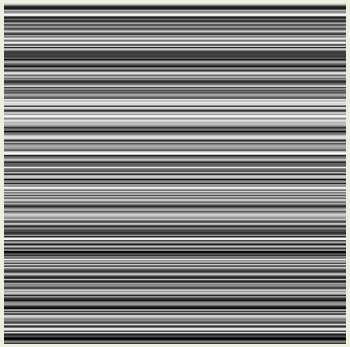
Not much smaller than 8

Basic Data Redundancies

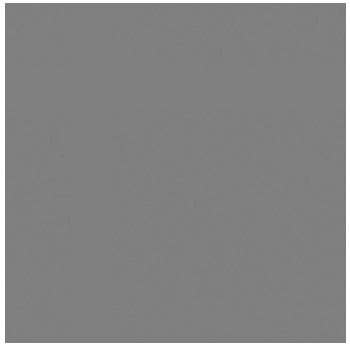
1. ***Coding*** (Code length for each pixel)
   - 8-bit
   - 16-bit
   - variable-length



2. ***Interpixel***
   (Spatial / Geometric / Interframe)
   - Correlation between pixels
   - Run-length coding



3. ***Psychovisual***
   - Importance for normal visual perception
   - Quantization



Singular Value Decomposition

Singular Value Decomposition (SVD)

For an $m \times n$ matrix A of rank r, there exists a factorization:

$$\boxed{A_{m\times n} = U_{m\times m}\,\Sigma_{m\times n}\,V^t_{n\times n}}$$

Σ is a diagonal matrix, its diagonal entries are called *Singular Values*.
U is an orthogonal matrix, its columns are *Left Singular Vectors*.
V is an orthogonal matrix, its columns are *Right Singular Vectors*.

$$\Sigma = diag(\sigma_1, \dots, \sigma_r)$$
$$rank(A) = r$$

Singular Values $\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq \cdots \geq \sigma_r$

Singular Value Decomposition (SVD)

$$m > n \quad \underbrace{\begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \\ * & * & * \\ * & * & * \end{bmatrix}_{m\times n}}_{A} = \underbrace{\begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{bmatrix}_{m\times m}}_{U} \underbrace{\begin{bmatrix} * & & \\ & * & \\ & & * \\ & & \\ & & \end{bmatrix}_{m\times n}}_{\Sigma} \underbrace{\begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \end{bmatrix}_{n\times n}}_{V^T}$$

$$m < n \quad \underbrace{\begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix}_{m\times n}}_{A} = \underbrace{\begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \end{bmatrix}_{m\times m}}_{U} \underbrace{\begin{bmatrix} * & & & \\ & * & & \\ & & * & \end{bmatrix}_{m\times n}}_{\Sigma} \underbrace{\begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix}_{n\times n}}_{V^T}$$

$$A_k = \sum_{i=1}^{k} \sigma_i u_i v_i$$

column notation: sum of rank 1 matrices

Application 1: Matrix Rank

The rank of A is the number of non-zero singular values

$$A = U \begin{bmatrix} \sigma_1 & & & & & \\ & \ddots & & & & \\ & & \sigma_r & & & \\ & & & 0 & & \\ & & & & \ddots & \\ & & & & & 0 \end{bmatrix} V^T$$

$$rank(A) = r$$

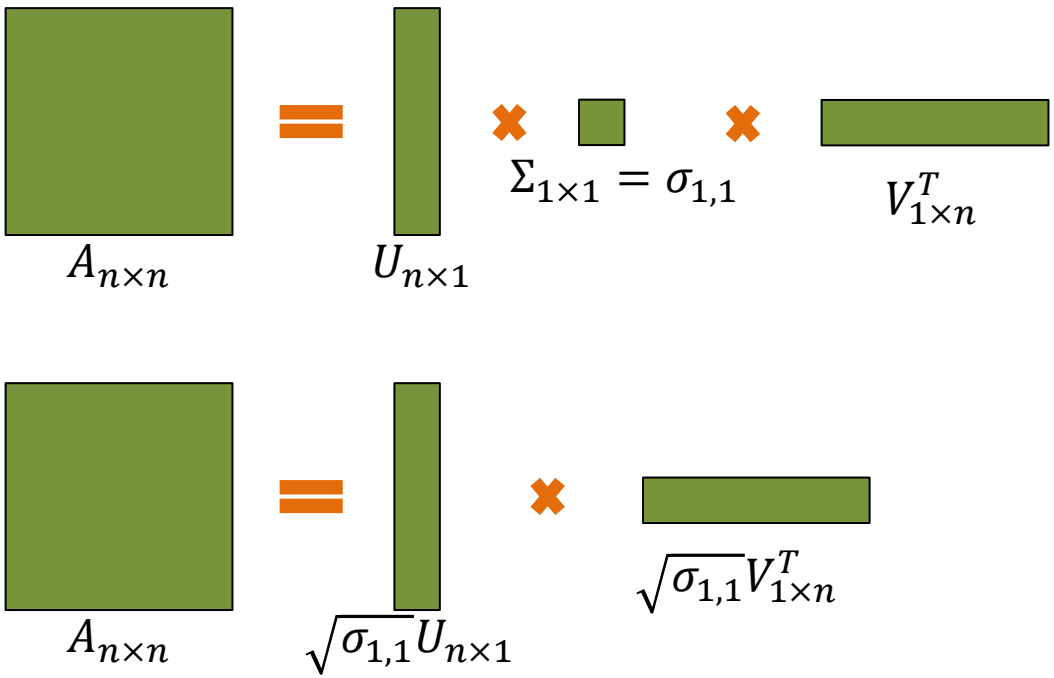
Application 2: Separable Matrices

$$\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} * \begin{bmatrix} b_1 & b_2 & \dots & b_n \end{bmatrix} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} \times \begin{bmatrix} b_1 & b_2 & \dots & b_n \end{bmatrix}$$

Convolution/ Matrix multiplication

Separable Matrices, Rank = 1

Application 2: Separable Matrices



Application 3: Rotation Approximation

$$A = U\,\Sigma\,V^T$$

$$\hat{R} = U\,I\,V^T$$

$$\hat{R} = \underset{R}{\operatorname{argmin}} \|A - R\|_F \qquad R \in SO(3)$$

Frobenius norm: $\|X_{m\times n}\|_F = \left(\sum_{i=1}^{m}\sum_{j=1}^{n} x_{i,j}^2\right)^{\frac{1}{2}} = \sqrt{tr(X^T X)}$

Application 4: Low-Rank Approximation

$$A = U \quad diag(\sigma_1, \dots, \sigma_r, 0, \dots, 0) \quad V^T$$

$$\tilde{A} = U \quad diag(\sigma_1, \dots, \sigma_k, 0, \dots, 0) \quad V^T \qquad k < r$$

$$A_k = \sum_{i=1}^{k} \sigma_i u_i v_i$$

column notation: sum of rank 1 matrices

Application 5: Matrix Inverse (Pseudo Inverse)

SVD decomposition: $A = U\,\Sigma\,V^T$

If A is invertible, inverse: $A^{-1} = V\,\Sigma^{-1}\,U^T$

In general, pseudoinverse: $A^{\dagger} = V\,\Sigma^{-1}\,U^T$
(even for not invertible matrices)

$A^{\dagger}$ is pseudo invese of A iff: $AA^{\dagger}A = A$ and $A^{\dagger}AA^{\dagger} = A^{\dagger}$

$\dagger \leftarrow$ dagger symbol

Application 6: Solving Homogeneous Systems

$$Ah = 0$$

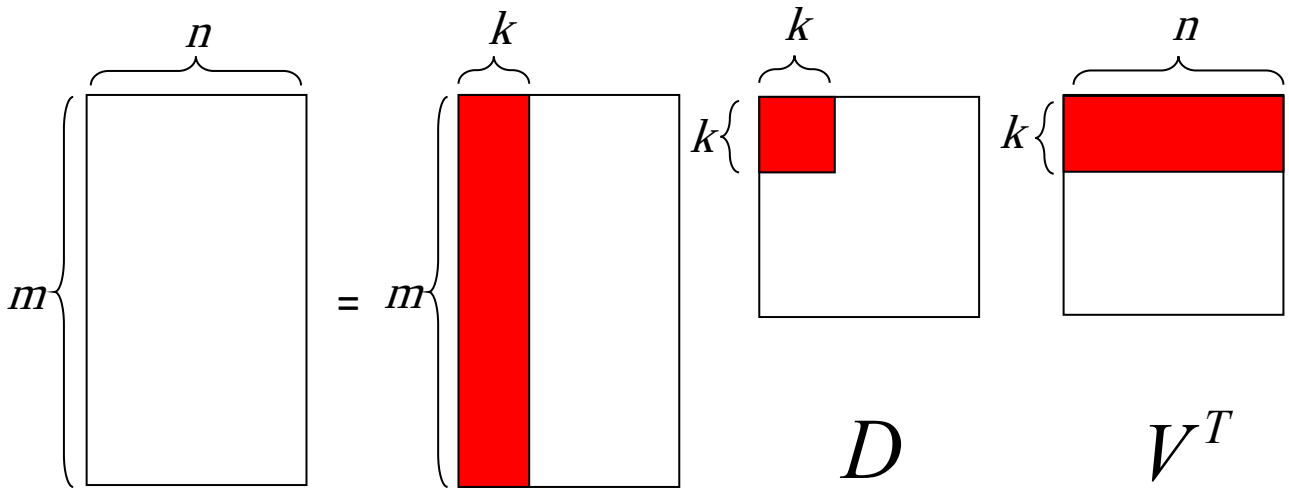
$$A = U\,\Sigma\,V^T$$

$$h = V(:, end)$$

Last column of V

Application 7: SVD for Image Compression

Correlation: Singular vectors corresponding to larger singular values contain more information.



$A^{(k)}$ U

$$A = \sum_{i=1}^{k} \sigma_i u_i v_i$$

The number of entries of $A = mn$

The number of saved entries after SVD $= mk + k + nk = k(m + n + 1)$

$$k(m + n + 1) < mn \;\rightarrow\; k < \frac{mn}{m + n + 1} \approx \frac{m}{2}$$

Original Image, Size: 2208×1656

Compressed x14; k = 200

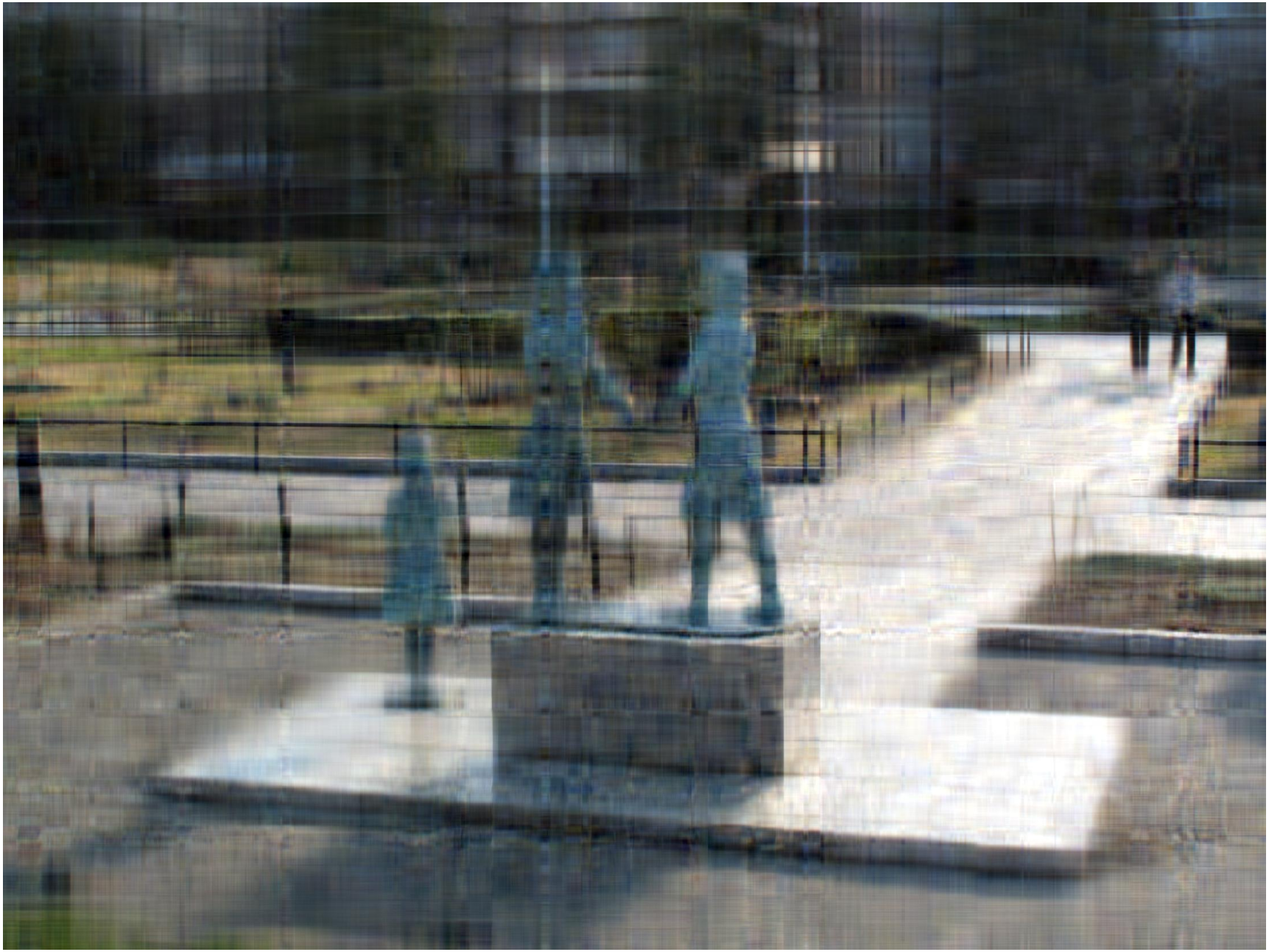
Compressed x28; k = 100

Compressed x35; k = 80

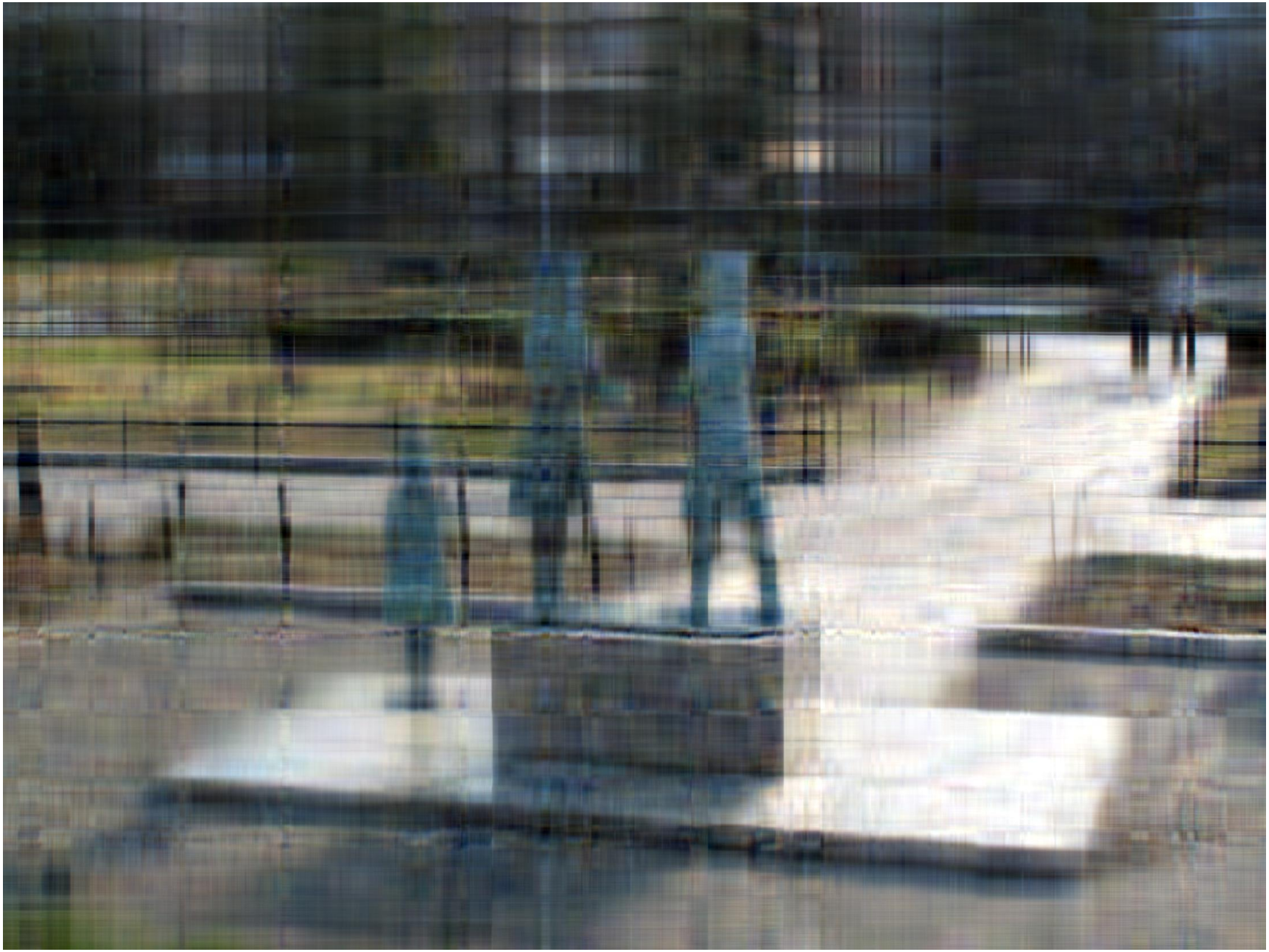
Compressed x57; k = 50

Compressed x95; k = 30

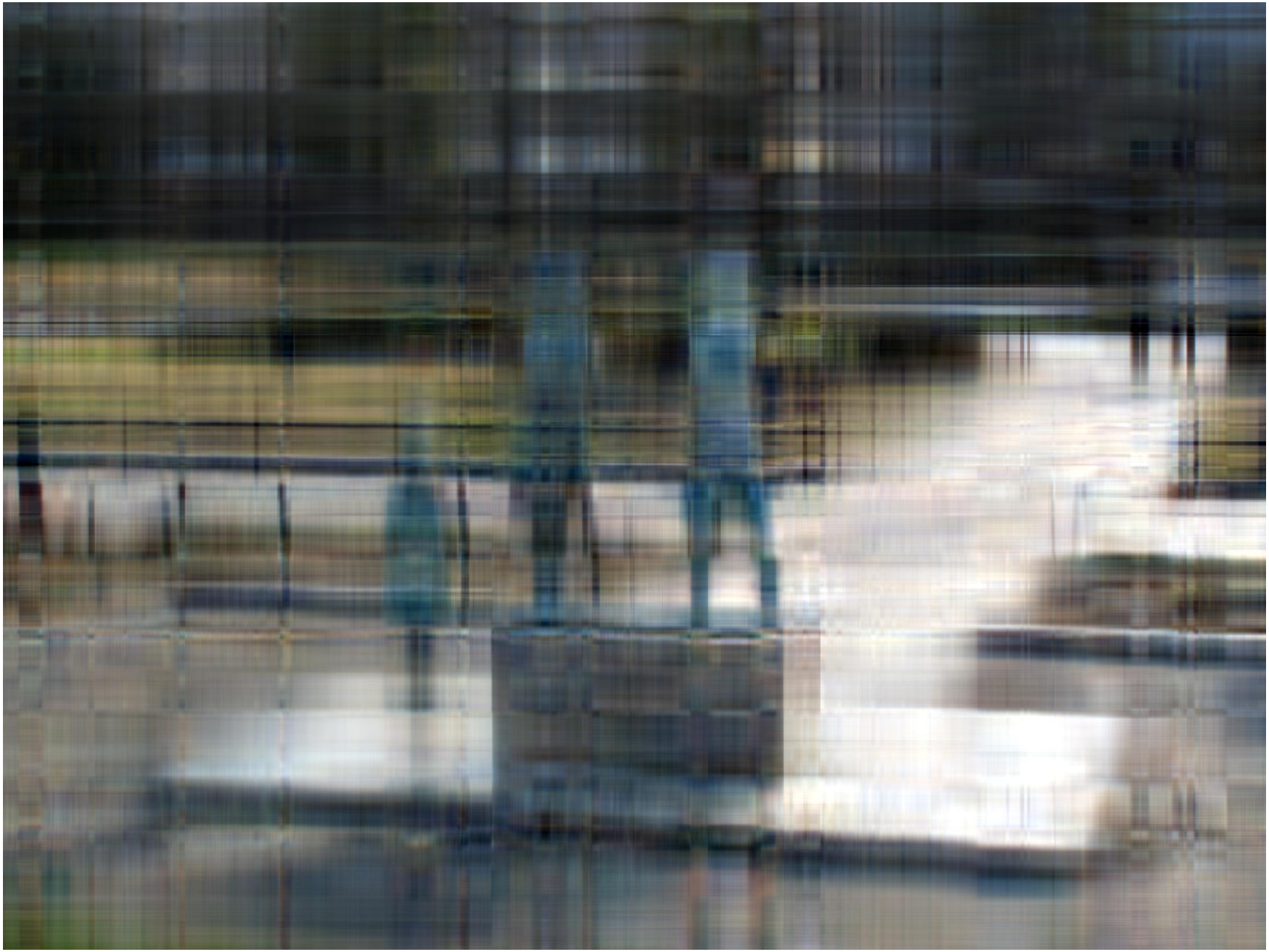
Compressed x142; k = 20



Compressed x189; k = 15



Compressed x284; k = 10



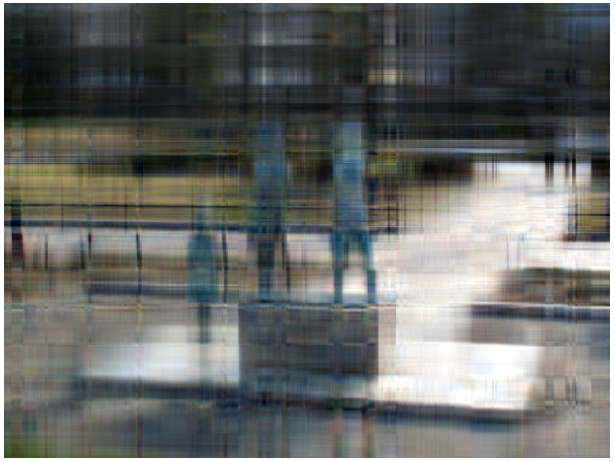
Example of Correlation Between Pixels
Singular Value Decomposition (SVD)

Original Image

Compression = x14

Compression = x35

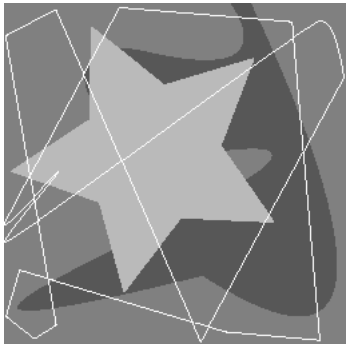
Compression = x142



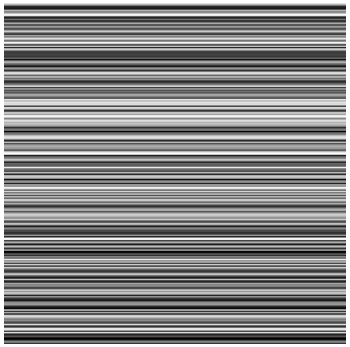
Compression = x284

Basic Data Redundancies

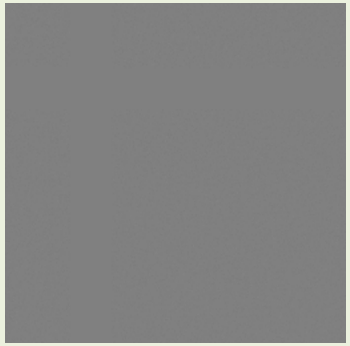
1. ***Coding*** (Code length for each pixel)
   - 8-bit
   - 16-bit
   - variable-length



2. ***Interpixel***
   (Spatial / Geometric / Interframe)
   - Correlation between pixels
   - Run-length coding

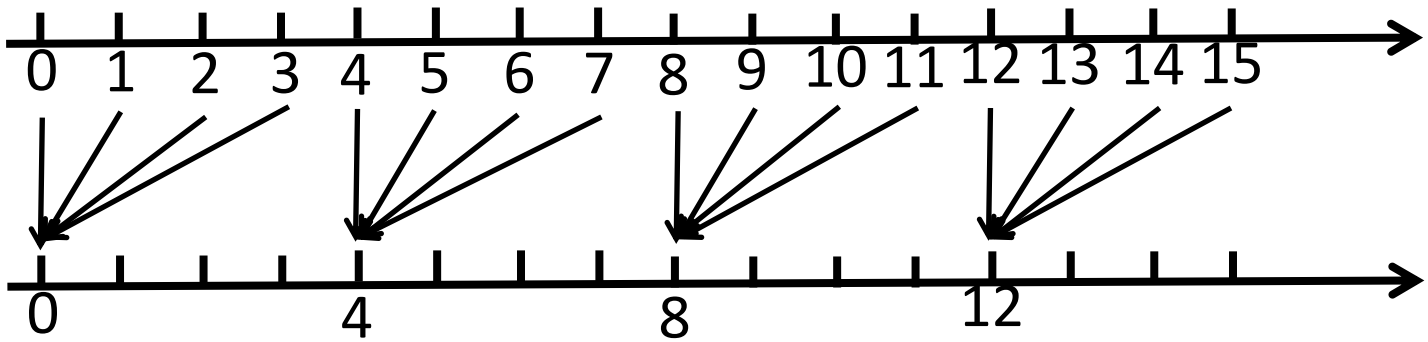


3. ***Psychovisual***
   - Importance for normal visual perception
   - Quantization



Quantization

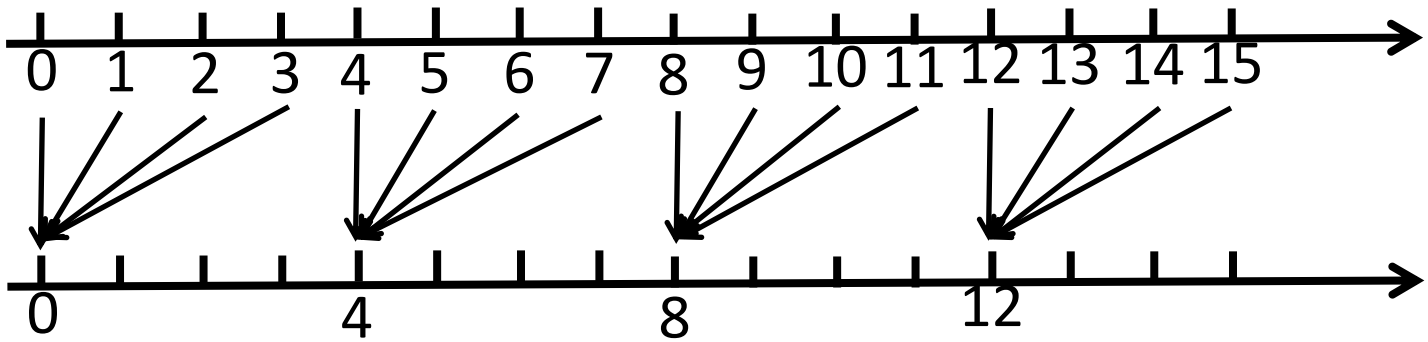
To boost the performance of Huffman coding, we would better have large number of few symbols instead of distributing them between multiple symbols.



$$\left\lfloor \frac{a_i}{k} \right\rfloor \times k$$

Quantization

To boost the performance of Huffman coding, we would better have large number of few symbols instead of distributing them between multiple symbols.



Coder

$$\left\lfloor \frac{a_i}{k} \right\rfloor$$

Decoder

$$\left\lfloor \frac{a_i}{k} \right\rfloor \times k$$

$$\left\lfloor \frac{a_i}{16} \right\rfloor \times 16$$

Entropy = 3.5382

Average Length in Huffman Coding = 3.5985

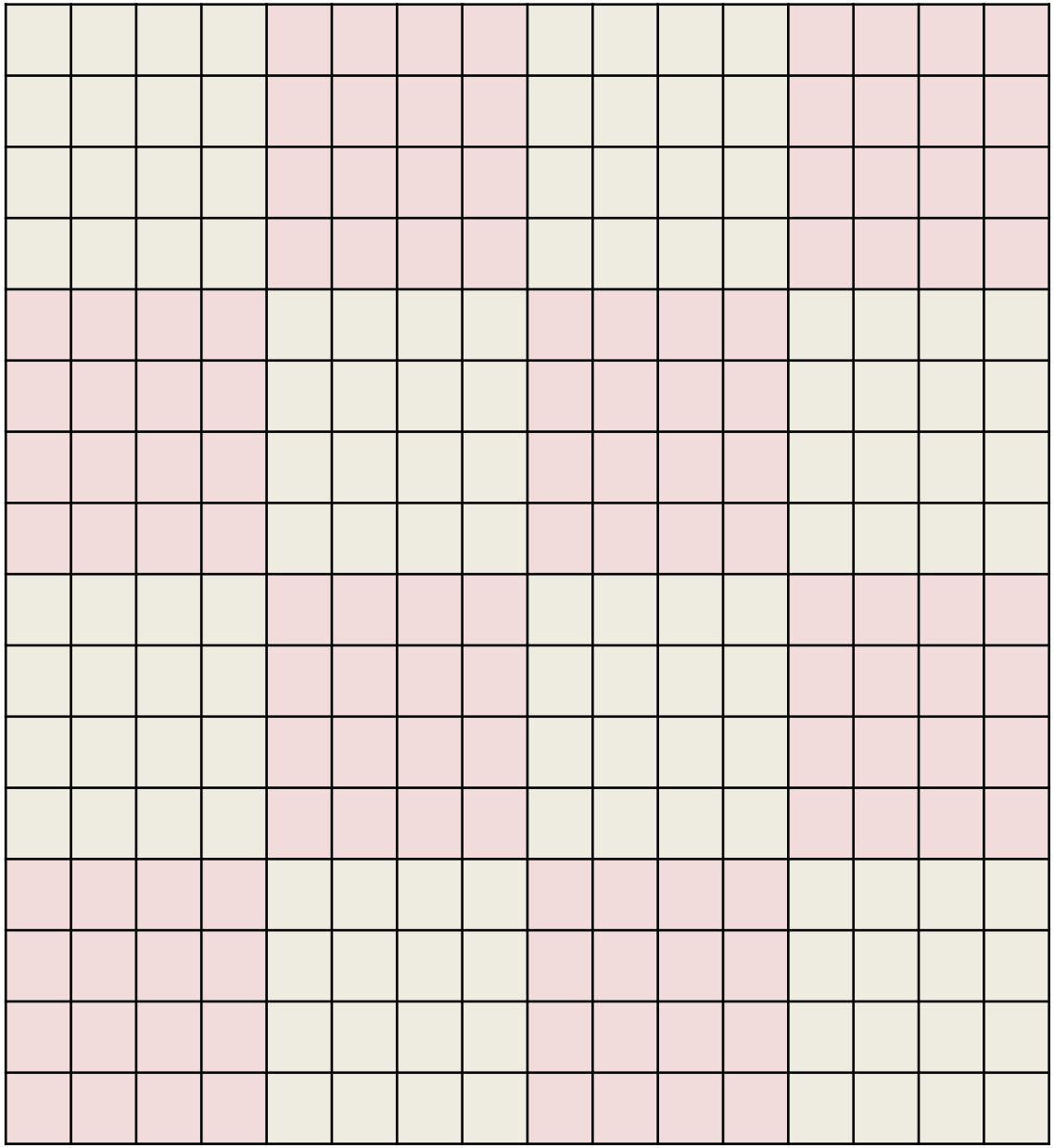
$$\left\lfloor \frac{a_i}{32} \right\rfloor \times 32$$

Entropy = 2.7187

Average Length in Huffman Coding = 2.7922

JPEG Compression

Blocks or Whole Image?



Considering $n \times n$ blocks is more efficient!

Also, far pixels are not necessarily correlated!

In JPEG, $n = 8$

- Preferably convert the original image to $Y'C_BC_R$ color space.
- Possibly downsampling C_B and C_R channels.
- Block splitting

*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*

8×8 *block*

- Subtracting by 128.

Correlation Between Pixels

Goal: Transferring the data to a new domain where few entries can reproduce the original data by going back to the original domain.

Transformation to a new domain

$$g(u,v) = \sum_x \sum_y f(x,y)\Gamma(x,y,u,v)$$

Image in the new domain

Transforming elements

Similar to DFT (Discrete Fourier Transform)?
DFT not necessarily useful every where!
We need to see what we need to do, and what tools can be used for that.

$$\begin{bmatrix} 170 & 171 & 171 & 169 & 169 & 168 & 168 & 167 \\ 171 & 171 & 172 & 170 & 169 & 168 & 168 & 167 \\ 171 & 172 & 173 & 170 & 170 & 169 & 168 & 169 \\ 171 & 172 & 173 & 170 & 171 & 172 & 169 & 169 \\ 173 & 170 & 172 & 174 & 171 & 171 & 171 & 170 \\ 173 & 171 & 172 & 173 & 173 & 170 & 170 & 171 \\ 174 & 174 & 172 & 172 & 172 & 171 & 169 & 170 \\ 175 & 175 & 174 & 171 & 172 & 172 & 172 & 173 \end{bmatrix}$$

An 8×8 block of image

$mean = 171$

$$\begin{bmatrix} -1 & 0 & 0 & -2 & -2 & -3 & -3 & -4 \\ 0 & 0 & 1 & -1 & -2 & -3 & -3 & -4 \\ 0 & 1 & 2 & -1 & -1 & -2 & -3 & -2 \\ 0 & 1 & 2 & -1 & 0 & 1 & -2 & -2 \\ 2 & -1 & 1 & 3 & 0 & 0 & 0 & -1 \\ 2 & 0 & 1 & 2 & 2 & -1 & -1 & 0 \\ 3 & 3 & 1 & 1 & 1 & 0 & -2 & -1 \\ 4 & 4 & 3 & 0 & 1 & 1 & 1 & 2 \end{bmatrix}$$

Subtracted by the mean

$$\begin{bmatrix} -30 & 14 & -2 & -2 & -4 & -1 & -1 & 3 \\ -24 & 18 & -3 & -3 & -3 & 2 & 1 & 3 \\ -12 & 15 & -2 & -6 & -3 & -4 & 4 & 3 \\ -2 & 10 & -5 & 1 & -7 & -5 & 7 & 1 \\ 8 & 7 & -5 & 0 & 6 & 11 & 2 & -2 \\ 10 & 8 & -4 & -1 & 10 & 1 & 0 & 3 \\ 12 & 17 & 0 & 3 & 3 & -5 & 0 & -2 \\ 32 & 11 & 10 & -1 & -3 & -5 & 2 & 3 \end{bmatrix}$$

DCT transform

$$\begin{bmatrix} -30 & 14 & -2 & -2 & -4 & -1 & -1 & 3 \\ -24 & 18 & -3 & -3 & -3 & 2 & 1 & 3 \\ -12 & 15 & -2 & -6 & -3 & -4 & 4 & 3 \\ -2 & 10 & -5 & 1 & -7 & -5 & 7 & 1 \\ 8 & 7 & -5 & 0 & 6 & 11 & 2 & -2 \\ 10 & 8 & -4 & -1 & 10 & 1 & 0 & 3 \\ 12 & 17 & 0 & 3 & 3 & -5 & 0 & -2 \\ 32 & 11 & 10 & -1 & -3 & -5 & 2 & 3 \end{bmatrix}$$

DCT transform

$$\begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 70 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

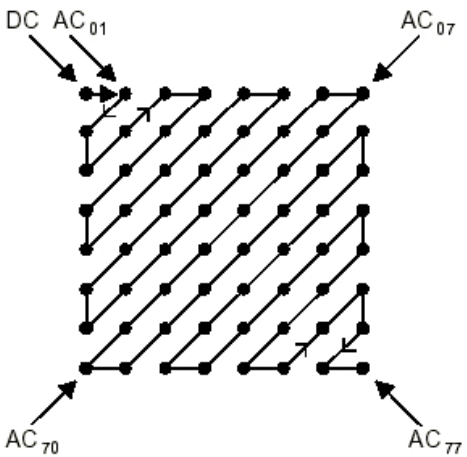
Quantization table

$$\begin{bmatrix} -2 & 1 & 0 & 0 \\ -2 & 2 & 0 \\ -1 & 1 \\ 0 & 1 \\ 0 \end{bmatrix}$$

Quantized

$$\begin{bmatrix} -2 & 1 & 0 & 0 \\ -2 & 2 & 0 & \\ -1 & 1 & & \\ 0 & 1 & & \\ 0 & & & \\ & & & \\ & & & \\ & & & \end{bmatrix}$$

Quantized



Sequence $\rightarrow -2, 1, -2, -1, 2, 0, 0, 0, 1, 0, 0, 1$

Correlation Between Pixels

Original Domain

*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*

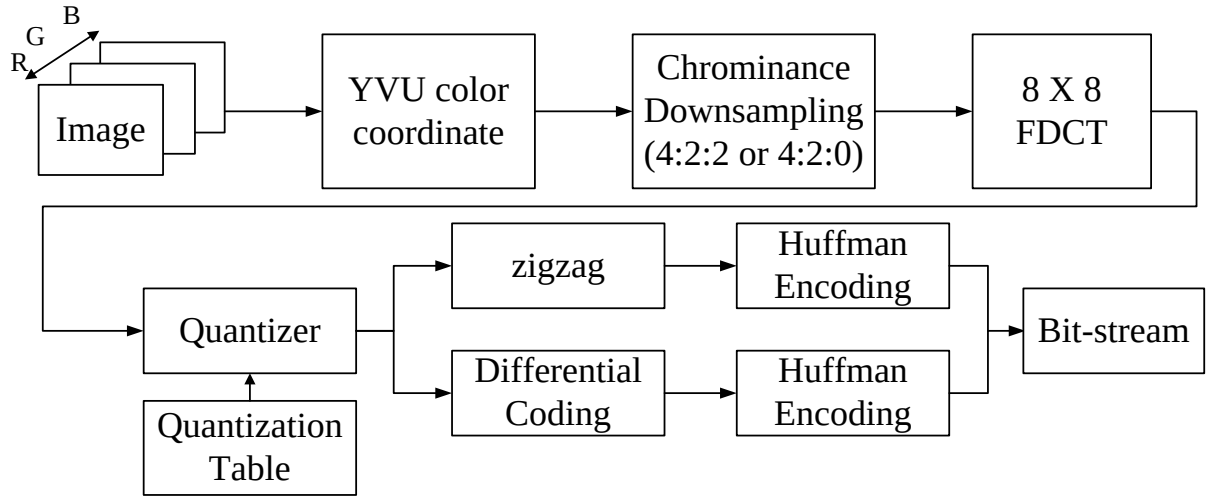
New Domain

+	+	+	+	+	+	+	+
+	+	+	+	+	+	+	+
+	+	+	+	+	+	+	+
+	+	+	+	+	+	+	+
+	+	+	+	+	+	+	+
+	+	+	+	+	+	+	+
+	+	+	+	+	+	+	+
+	+	+	+	+	+	+	+

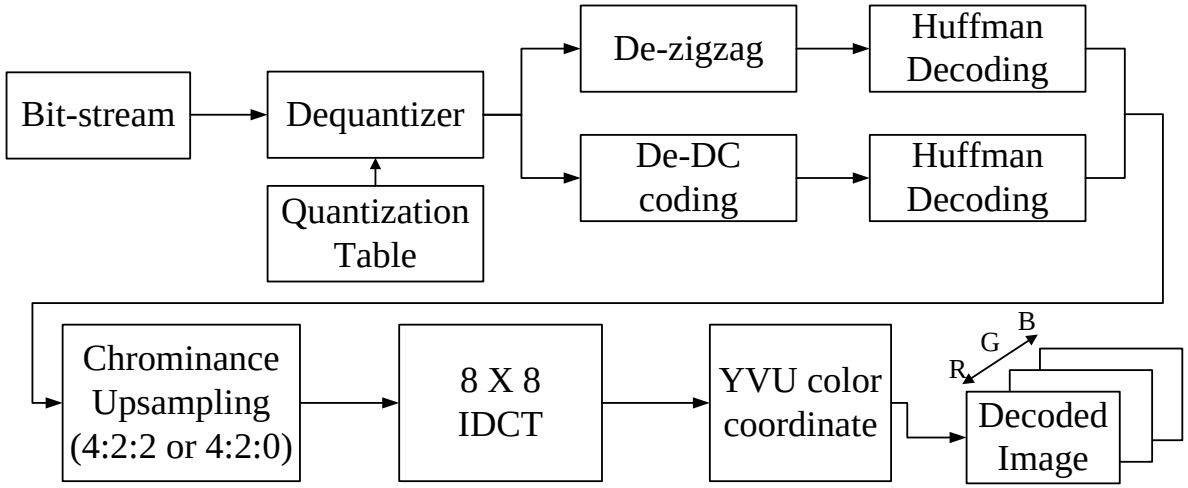
Keeping few elements in the desired domain,
we can reconstruct the original data!

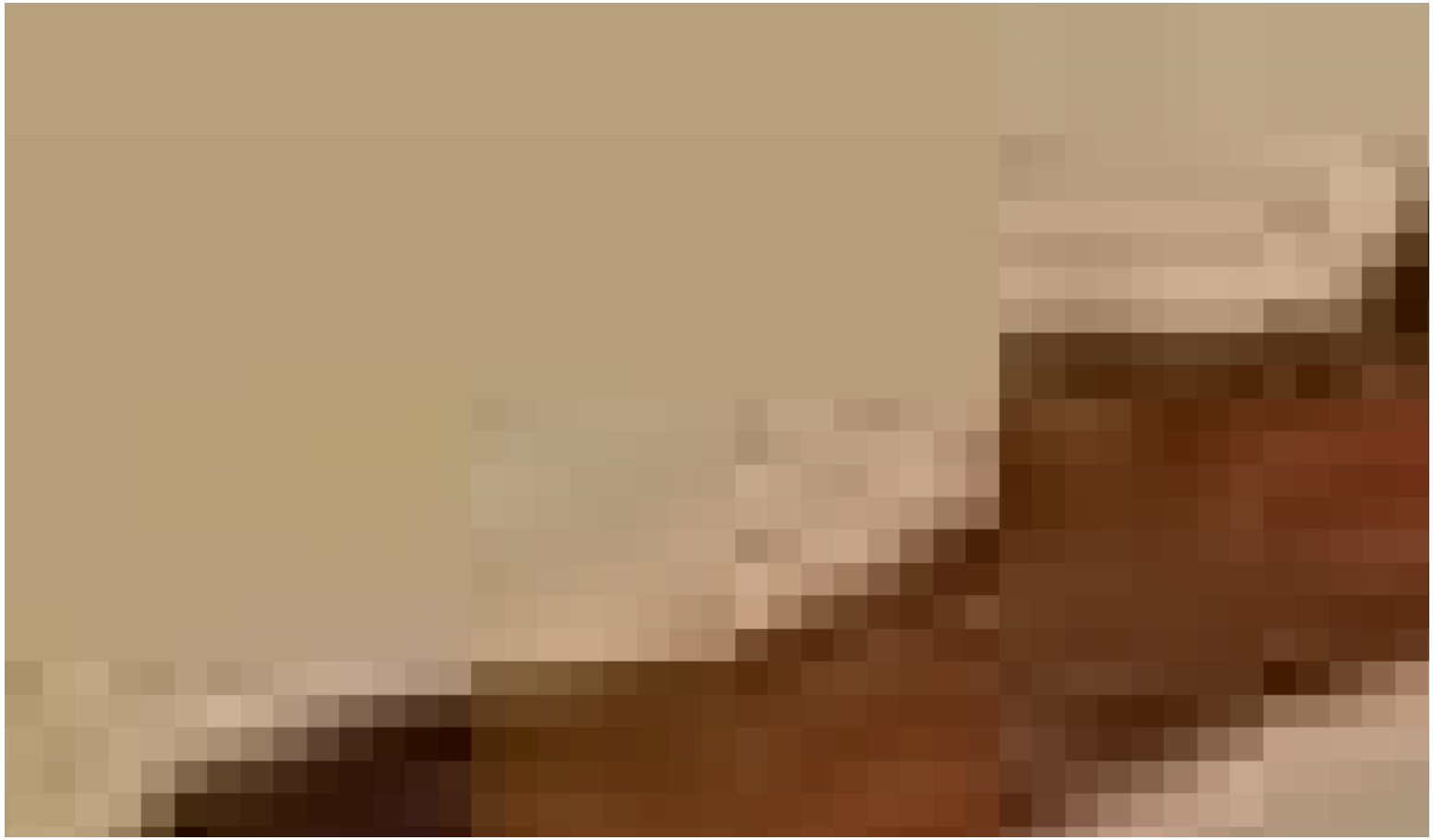
JPEG

The JPEG Encoder

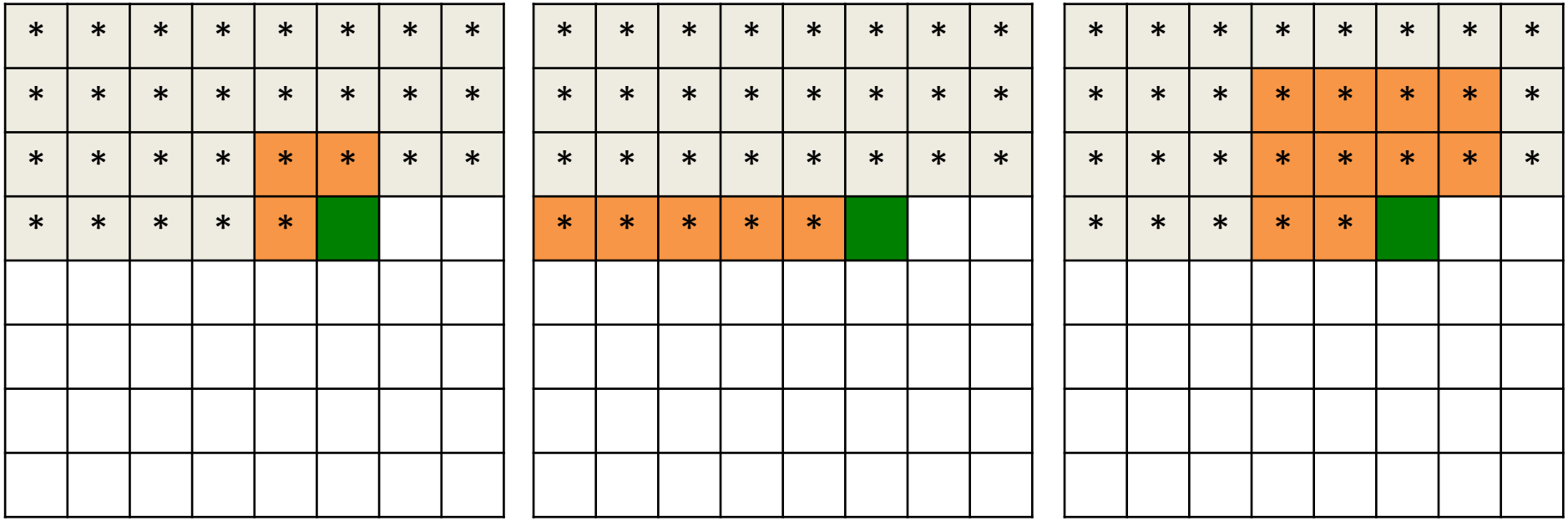


The JPEG Decoder





Predictive Coding (PNG)



MPEG

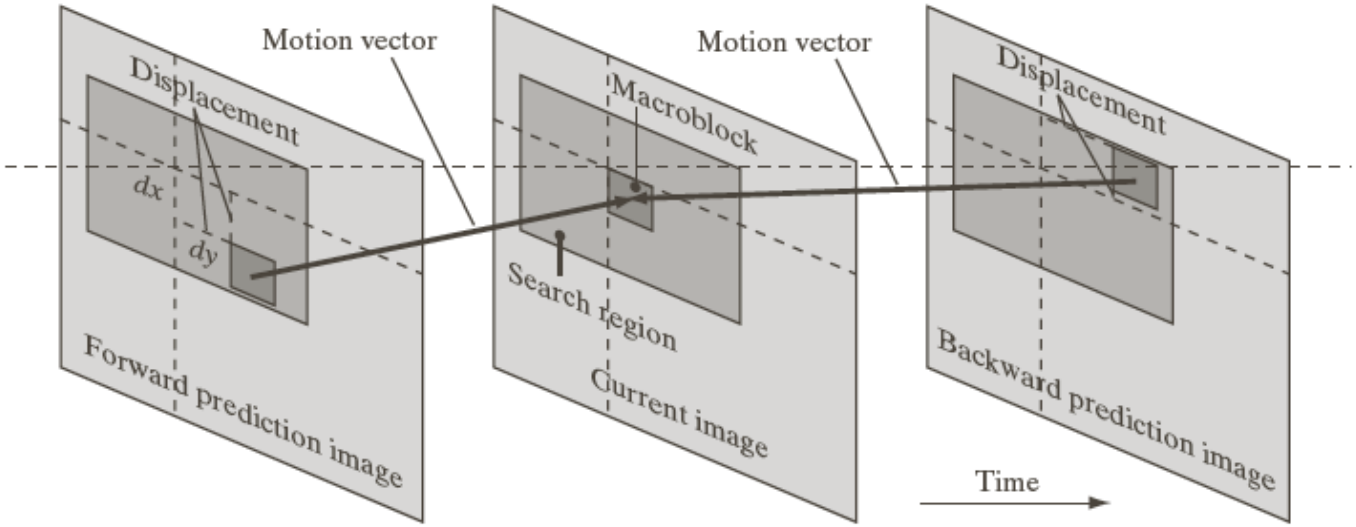


FIGURE 8.36 Macroblock motion specification.

General Image Compression System

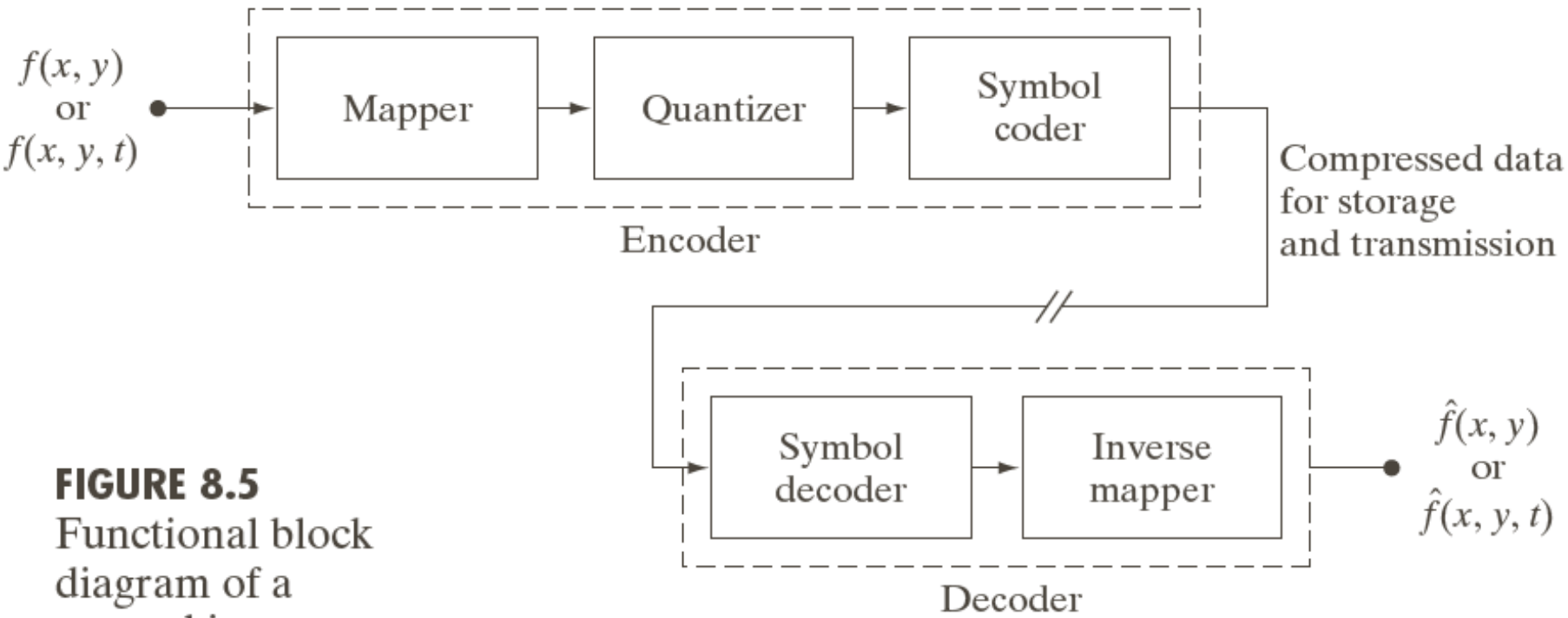


FIGURE 8.5 Functional block diagram of a general image compression system.

References

- Image Compression
  Gonzalez, chapter 8
  Szeliski, section 2.3.3

- SVD
  James Demmel
  Applied Numerical Linear Algebra
  Chapter 5