What is JPEG?

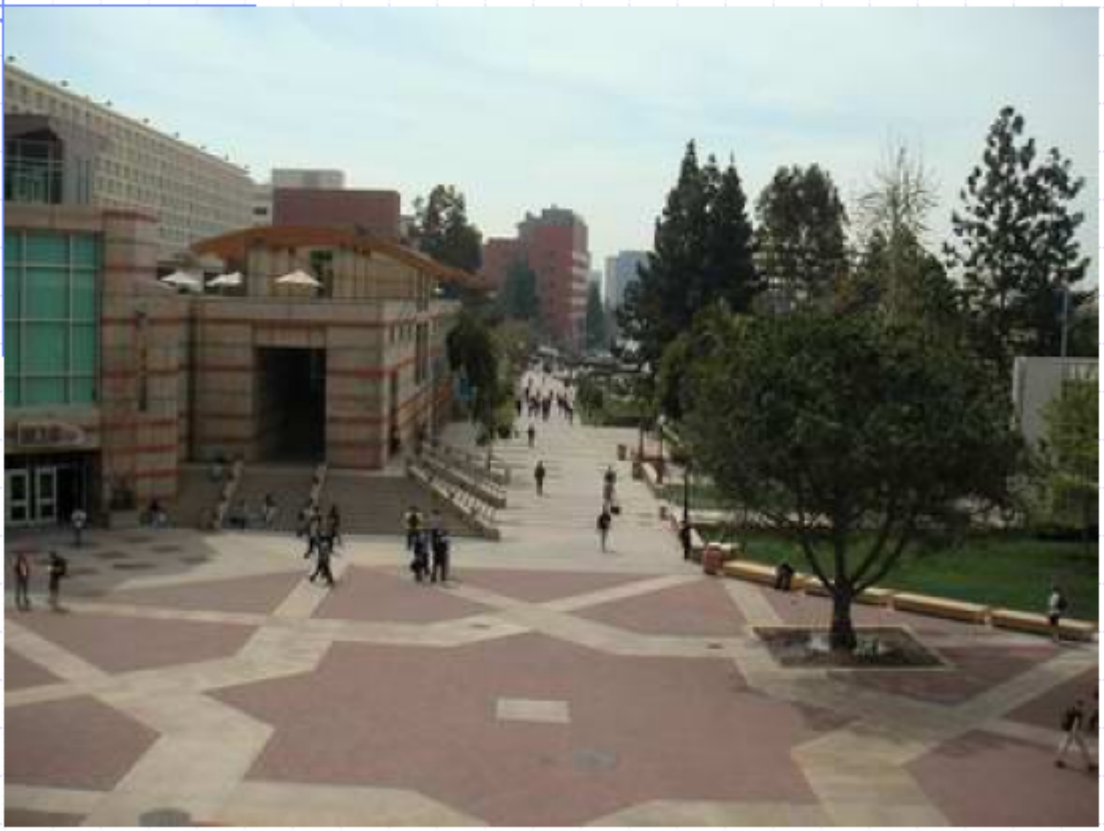
- JPEG is short for the 'Joint Photographic Experts Group'.
- The JPEG standard is fairly complex because, rather than defining an image file format, it defines a number of related image compression techniques.

JPEG Characteristics

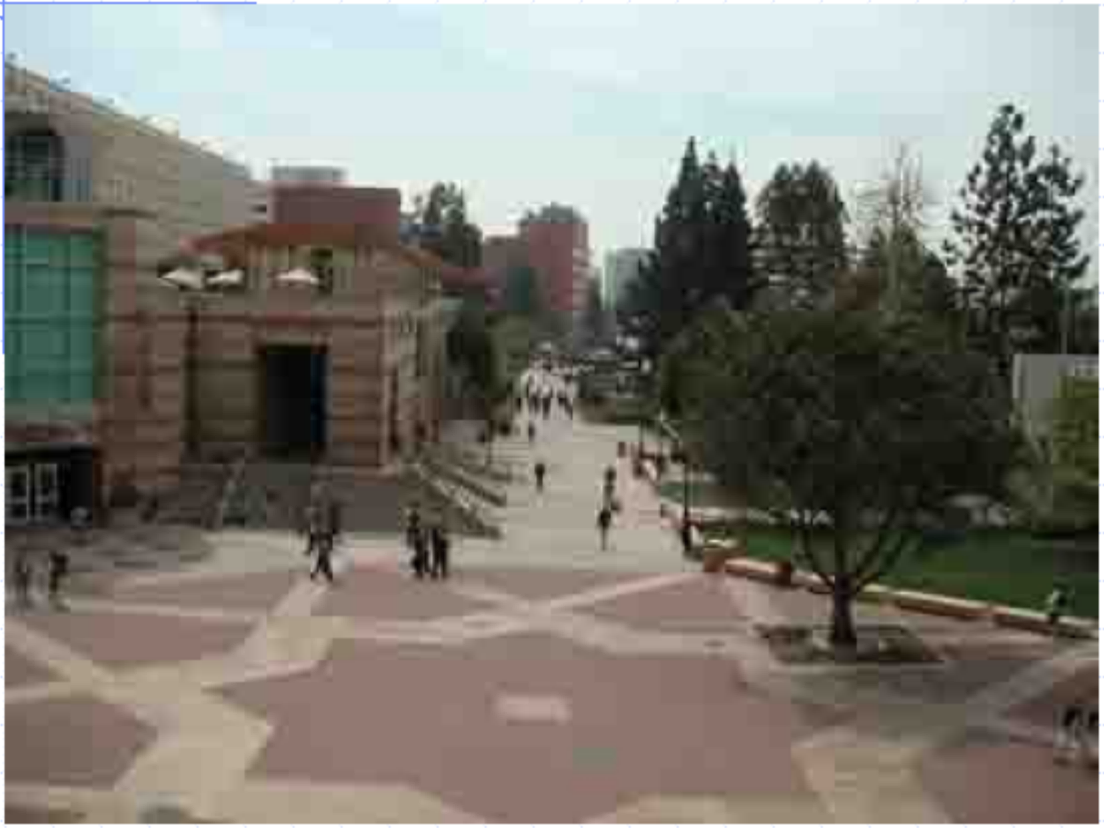
- Always Lossy Compression
- True 24-bit color (16 million colors)
- Compression ration of 2-100 : 1
- Good performance for pictures that are smooth with a lot of colors.
- Bad performance for pictures with sharp edges.

Uncompressed TIFF (400 x 300 x 24bpp - 360KB)

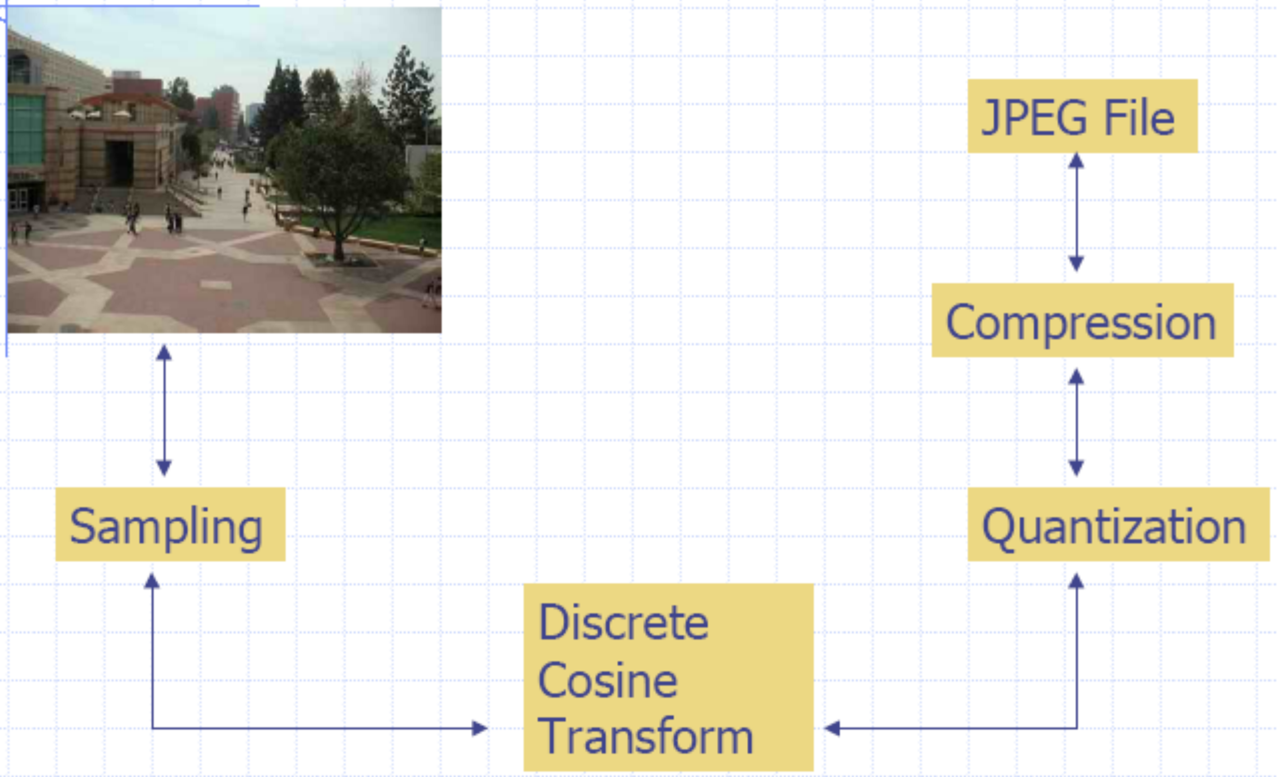
JPEG (19 KB – 5.28% of original image)



JPEG (11 KB – 3% of original image)



Dataflow of JPEG Compression Algorithm



Sampling: RGB Color System

- Three component representation of the color of a pixel
- Represents the intensities of the red, green, and blue components
- 24 bit "True Color"
- Each component represented with 8 bits of precision
- The components each contain roughly the same amount of information

Human Visual System

- The human eye has a tendency to notice variations of brightness intensity much more than variations of the color in an image
- The human eye is not as sensitive to high-frequency chrominance (color) components as it is to luminance (intensity) components
- We can take advantage of this by transforming the color space of RGB to another representation

YUV (YCrCb) Color Space

- An ideal format for JPEG compression
- The brightness and color information in an image are separated
- Concentrates the most important info into one component, allowing for greater compression
- Y component represents the color intensity of the image (equivalent to a black and white television signal)
- U and V represent the relative redness and blueness of the image

YUV Transformation

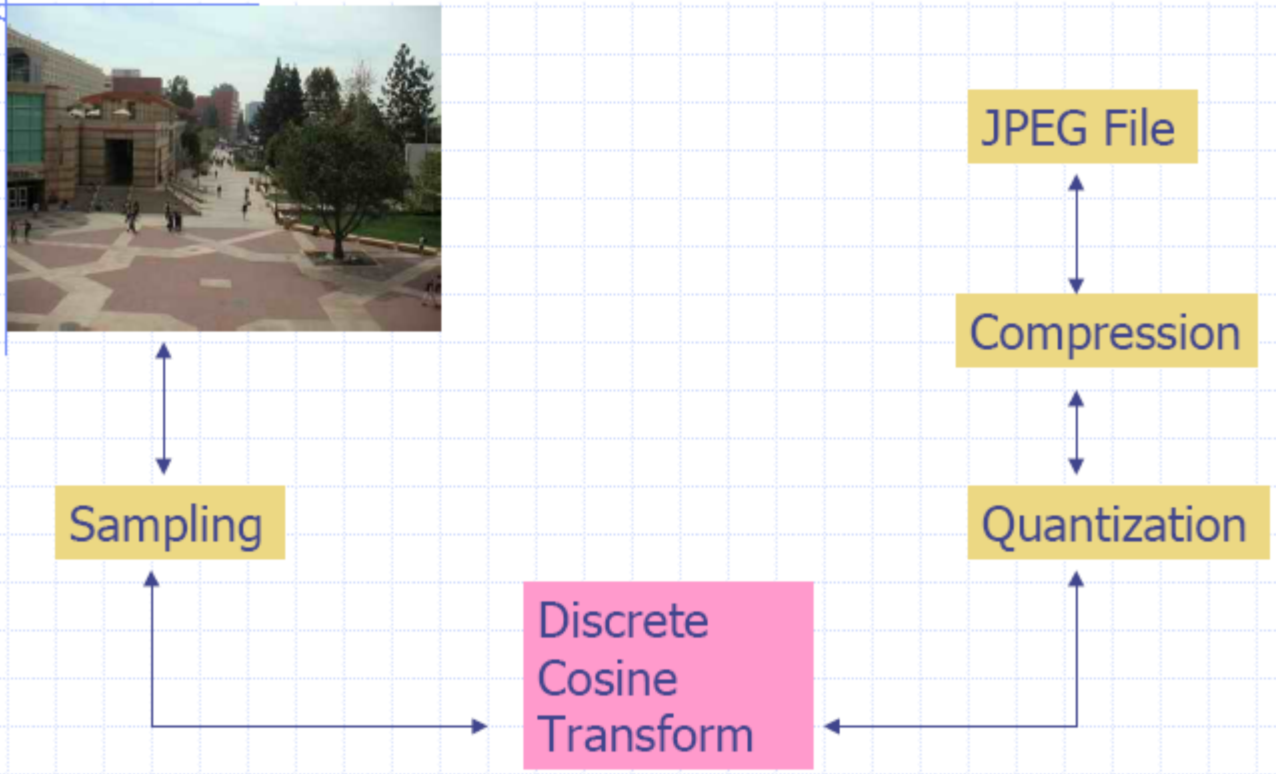
- A linear transformation from RGB to YUV and from YUV to RGB

$$Y = 0.299R + 0.587G + 0.114B$$
$$U = -0.1687R - 0.3313G + 0.5B + 128$$
$$V = 0.5R - 0.4187G - 0.0813B + 128$$

$$R = Y + 1.402V$$
$$G = Y - 0.34414(U - 128) - 0.71414(V - 128)$$
$$B = Y + 1.722(U - 128)$$

$$128 = 2^{\text{Sample Prescision}/2}$$

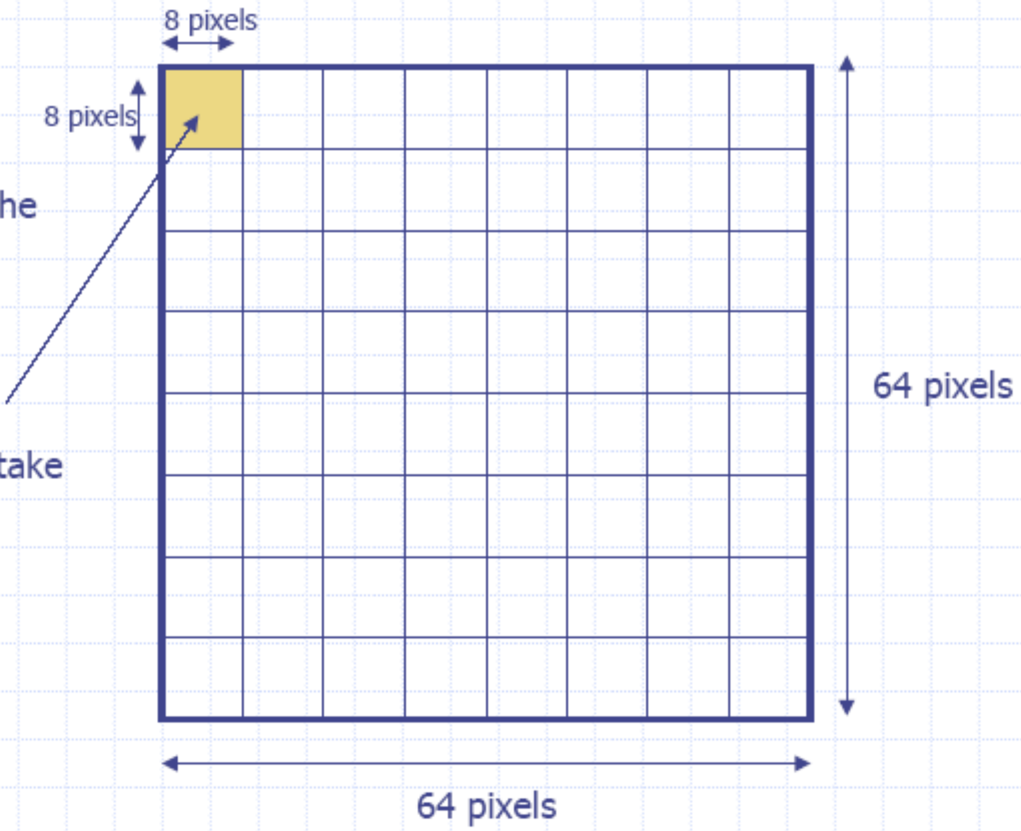
Discrete Cosine Transform



DCT on 8x8 blocks

- We will break the image into non-overlapping 8x8 blocks.

- For each block u(m,n), we will take an 8x8 DCT



DCT – Discrete Cosine Transform

- 8x8 block of Pixels are represented using cosine functions of different frequencies.
- A real unitary Transform
  - Unlike the DFT (FFT) which uses complex basis functions. The result is a transform that will give you real values in the transform domain.

$\underline{v} = \underline{C}\,\underline{u}$

$$\underline{C} = C(k,n) = \begin{cases} \dfrac{1}{\sqrt{N}} & k = 0 \quad 0 \le n \le N-1 \\[2ex] \sqrt{\dfrac{2}{N}} \cos\left(\dfrac{\pi(2n+1)}{2N}\right) & \begin{array}{l} 1 \le k \le N-1 \\ 0 \le n \le N-1 \end{array} \end{cases}$$

Unitary 2D-DCT

- Not surprisingly, it turns out that you can get better compression using the DCT if you take into account the horizontal and vertical correlation between pixels simultaneously.

$$\alpha(k)=\sqrt{\frac{2}{N}} \qquad \alpha(0)=\frac{1}{\sqrt{N}}$$

- Forward DCT

$$V(k,l)=\alpha(k)\alpha(l)\sum_{m=0}^{N-1}\sum_{n=0}^{N-1}u(m,n)\cos\left(\frac{\pi(2m+1)k}{2N}\right)\cos\left(\frac{\pi(2n+1)l}{2N}\right)$$

- Backward DCT

$$u(m,n)=\sum_{k=0}^{N-1}\sum_{l=0}^{N-1}\alpha(k)\alpha(l)V(k,l)\cos\left(\frac{\pi(2m+1)k}{2N}\right)\cos\left(\frac{\pi(2n+1)l}{2N}\right)$$

Example of DCT

🟥	Low Frequencies
🟨	Mid Frequencies
🟦	High Frequencies

u(m,n)

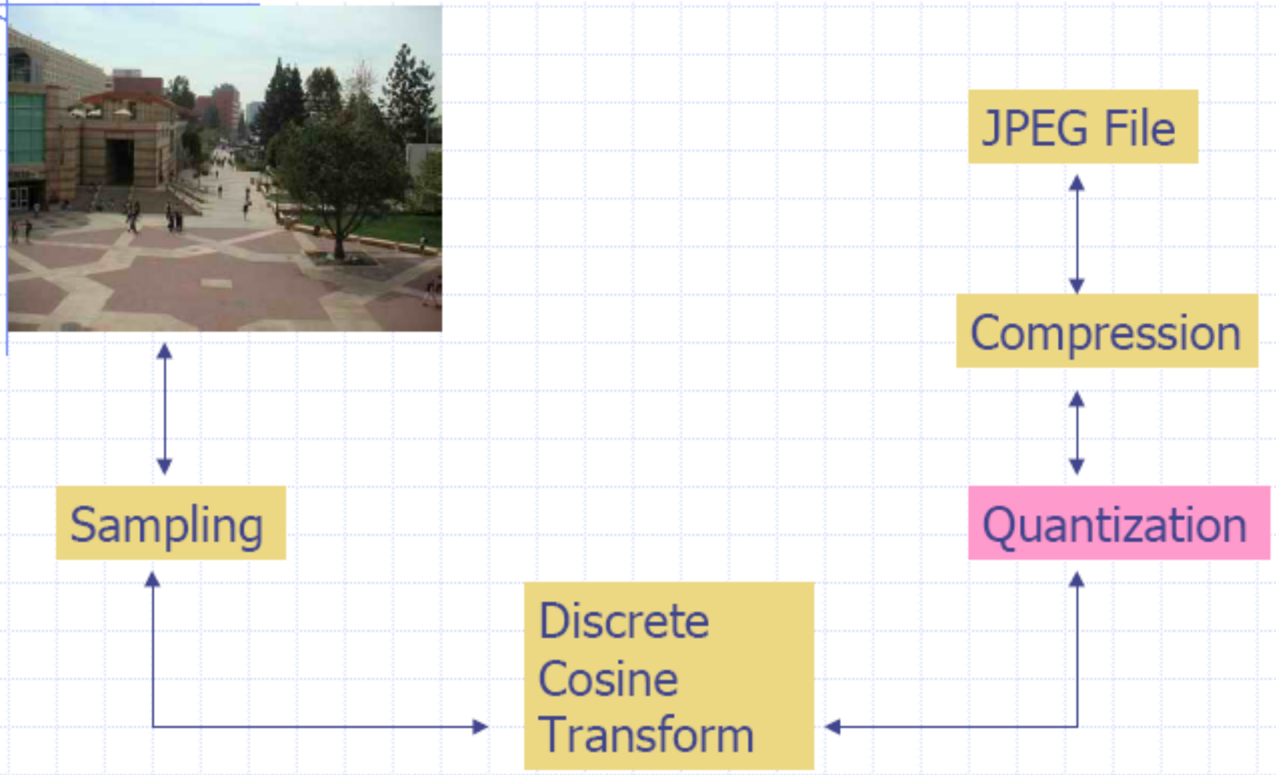
58	45	29	27	24	19	17	20
62	52	42	41	38	30	22	18
48	47	49	44	40	36	31	25
59	78	49	32	28	31	31	31
98	138	116	78	39	24	25	27
115	160	143	97	48	27	24	21
99	137	127	84	42	25	24	20
74	95	82	67	40	25	25	19

v(k,l)

-603	203	11	45	-30	-14	-14	-7
-108	-93	10	49	27	6	8	2
-42	-20	-6	16	17	9	3	3
56	69	7	-25	-10	-5	-2	-2
-33	-21	17	8	3	-4	-5	-3
-16	-14	8	2	-4	-2	1	1
0	-5	-6	-1	2	3	1	1
8	5	-6	-9	0	3	3	2

DCT

Quantization



Quantization

- Quantized Value = Round (coefficient / Quantum Value)

$$v'(k,l) = round\left(\frac{v(k,l)}{q(k,l)}\right)$$

- Choosing a quantum value as small as 20 would convert over half of the coefficients to zeros.
- The JPEG standard does not specify the quantization values to be used. This is left up to the application. However it does provide some quantization tables that have been tested empirically and found to generate good results

Example of Quantization

🟦 = zeros

$\underline{v}(k,l)$

-603	203	11	45	-30	-14	-14	-7
-108	-93	10	49	27	6	8	2
-42	-20	-6	16	17	9	3	3
56	69	7	-25	-10	-5	-2	-2
-33	-21	17	8	3	-4	-5	-3
-16	-14	8	2	-4	-2	1	1
0	-5	-6	-1	2	3	1	1
8	5	-6	-9	0	3	3	2

$\underline{v}'(k,l)$

-38	18	1	-3	-1	0	0	0
-9	-8	1	3	1	0	0	0
-3	-2	0	1	0	0	0	0
4	4	0	-1	0	0	0	0
-2	-1	0	0	0	0	0	0
-1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

$$\underline{v}(k,l) \xrightarrow{\text{Quantization}} \underline{v}'(k,l)$$

Quantization Table

Y Component

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	58	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

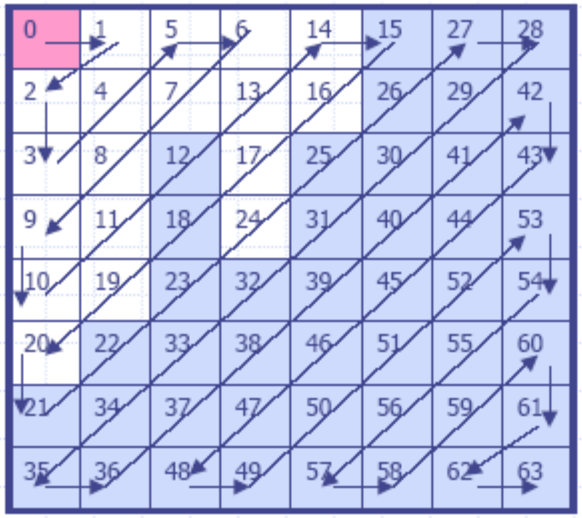
U and V Components

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

Zig-Zag Ordering

🟥 = DC

🟦 = zeros



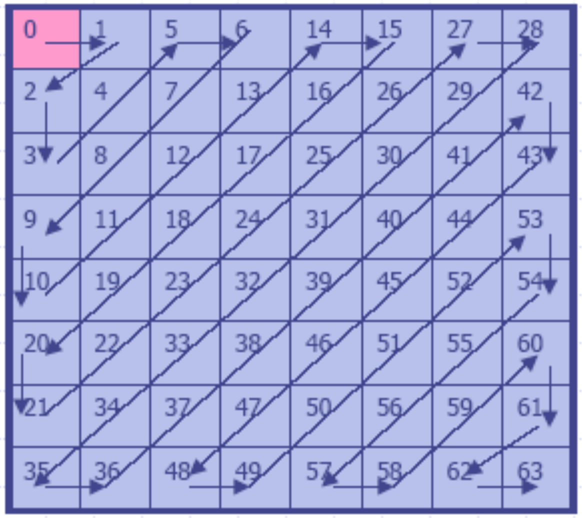
- The goal is to group all the zeros together, to allow compression

18 -9 -3 -8 1 -3 1 -2 4 -2 4 0 3 -1 0 1 1 0 -1 -1 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Run Length Coding

- Simple, lossless compression scheme
- Consecutive pixels with the same value are encoded using a run-length and value pair
  - 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 ->0x0A 0x00
- So in our example on the last page:
  - 18 -9 -3 -8 1 -3 1 -2 4 -2 4 0 3 -1 0 1 1 0 -1 -1 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  - We will take advantage of the fact that there is a long string of coefficients that are 0's. We will run-length code these.

AC/DC



🟥 = DC Coefficients

🟦 = AC Coefficients