http://tutorial9.s3.amazonaws.com/uploads/2008/05/jpeg-comparisons.jpg

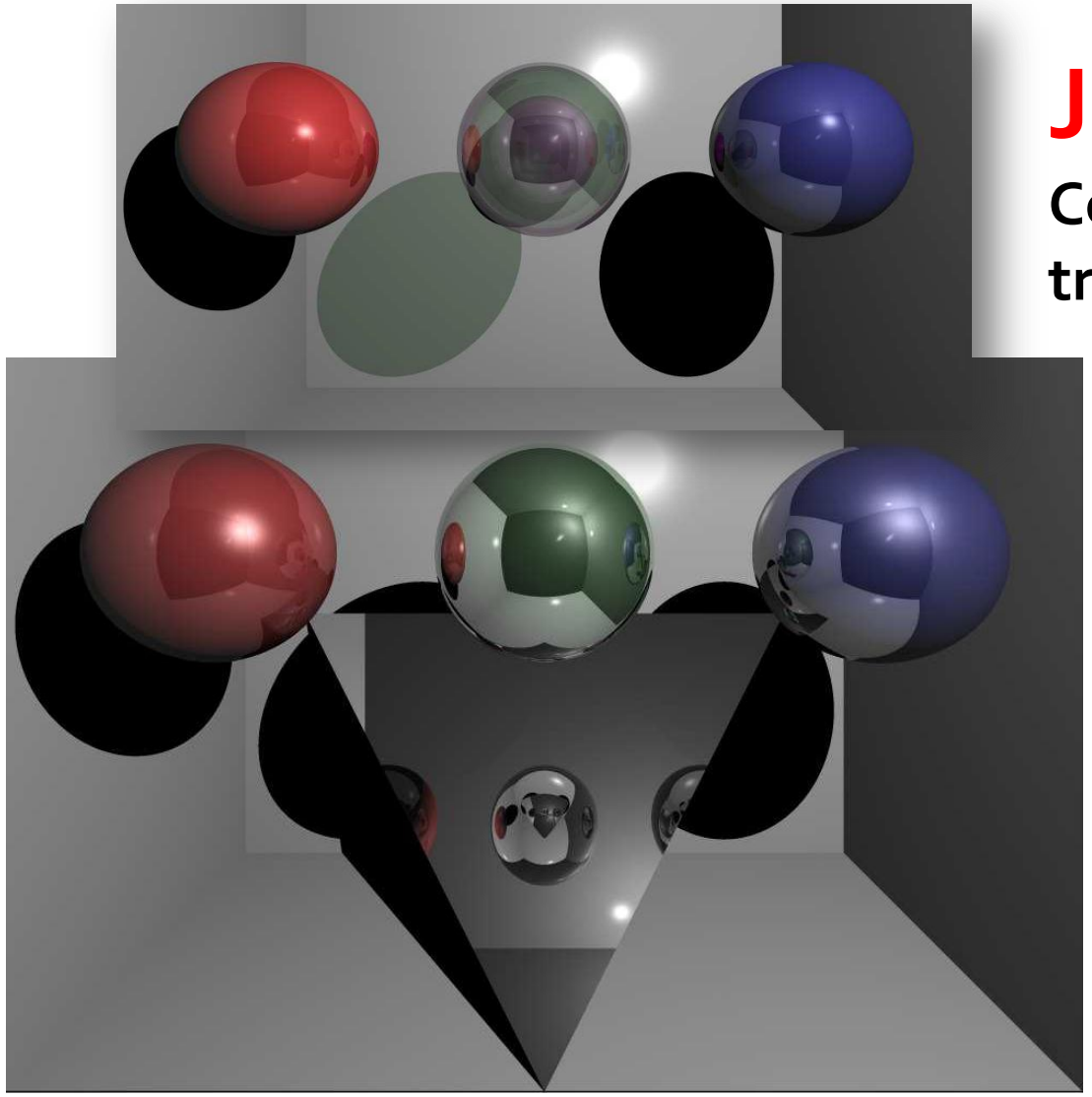
Image Compression

CS 148, Summer 2012
Introduction to Computer Graphics and Imaging

Justin Solomon

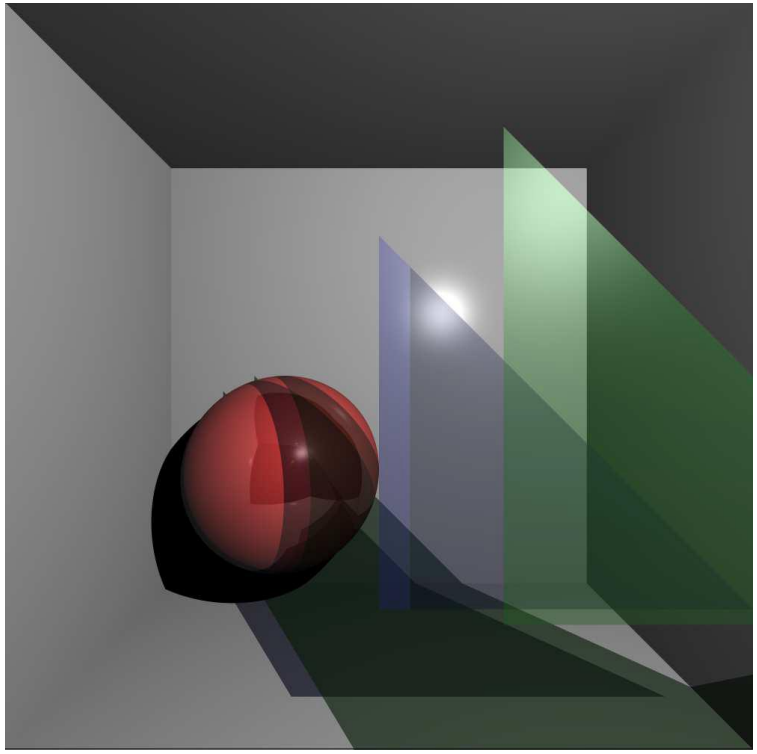
<ray_tracing>

Ray Tracing Homework Results

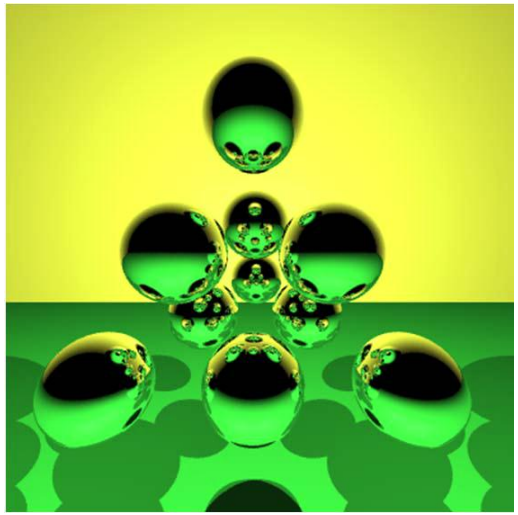


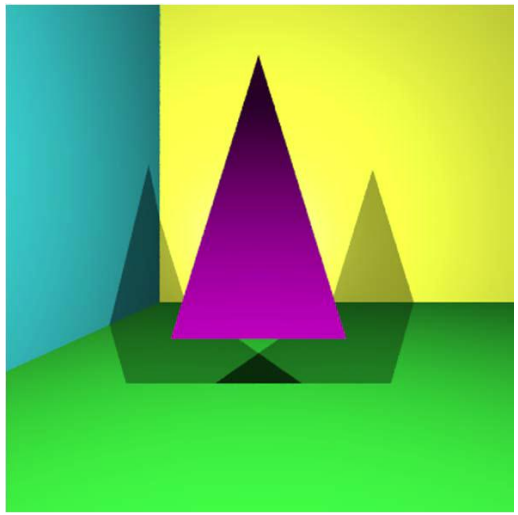
Jack Hessel

Complex materials/shadows, transparency



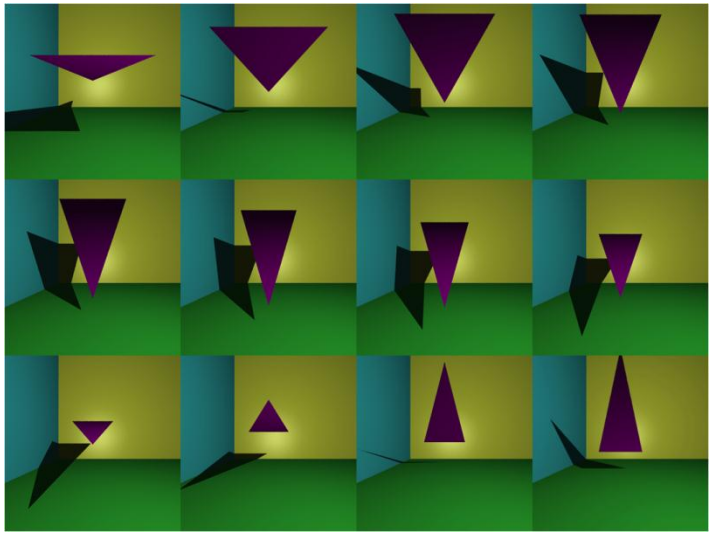
Ray Tracing Homework Results



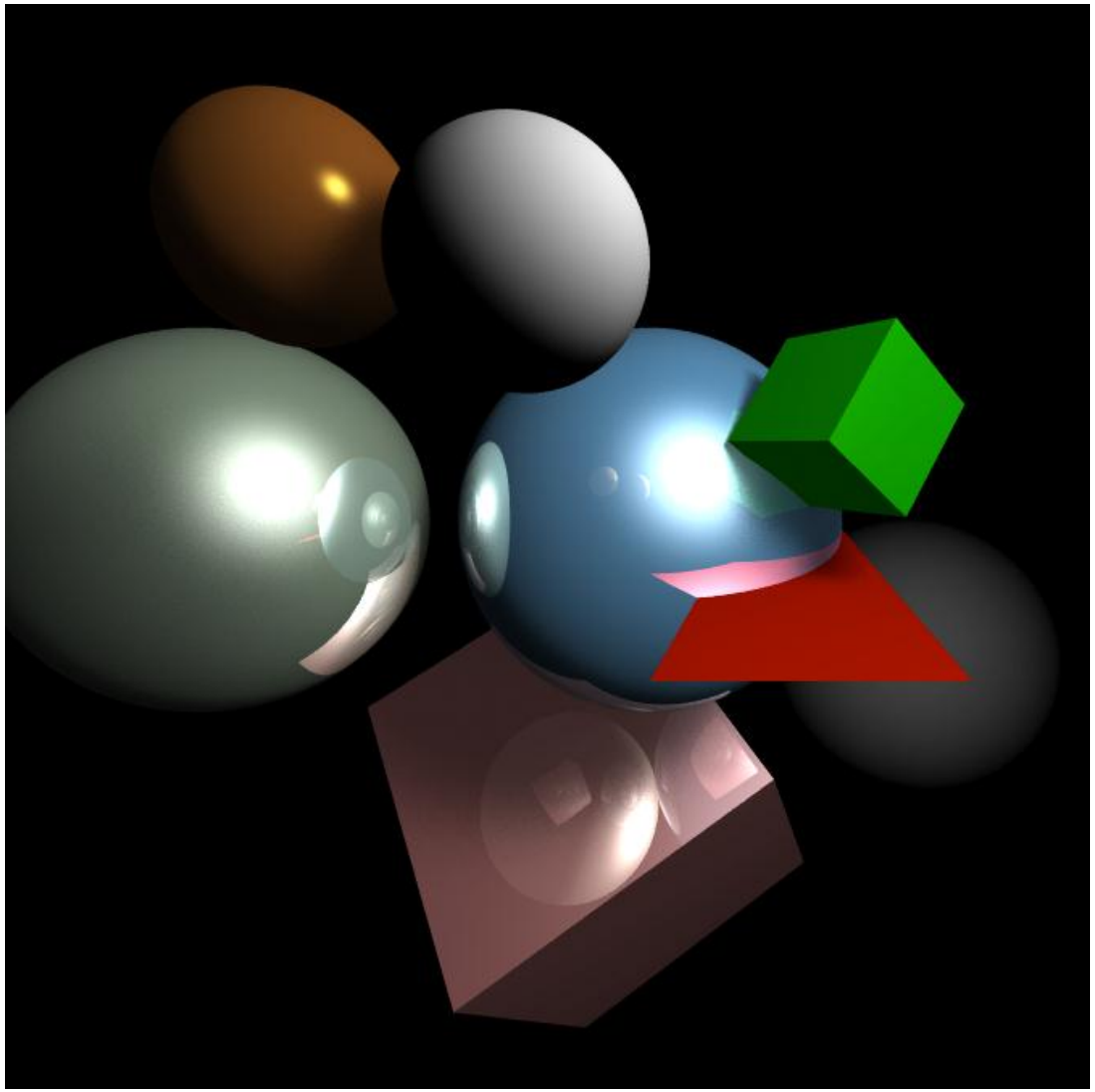


Junjie Liang

Animation, .obj shapes, multiple lights, transformations



Ray Tracing Homework Results



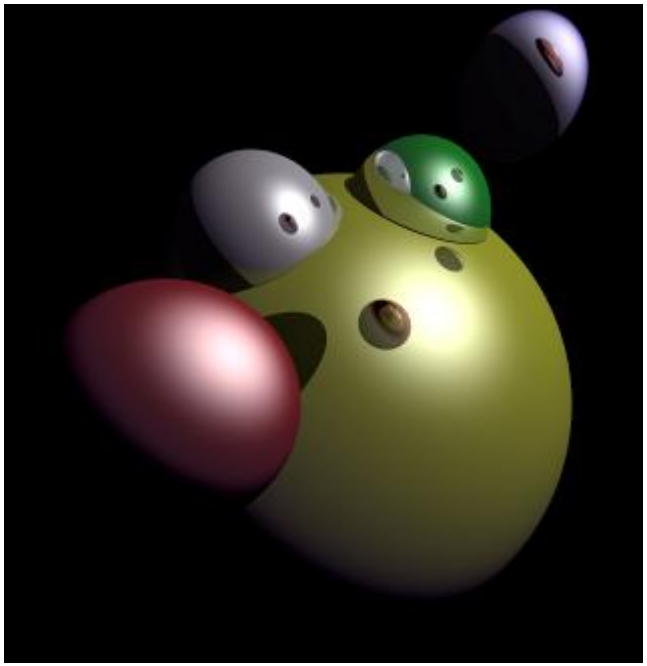
Alex Cope

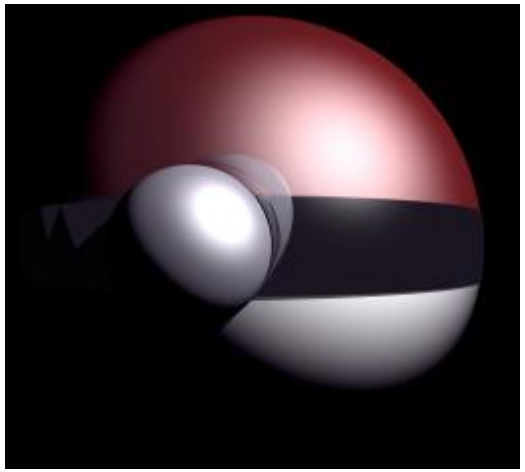
Lighting falloff, soft shadows

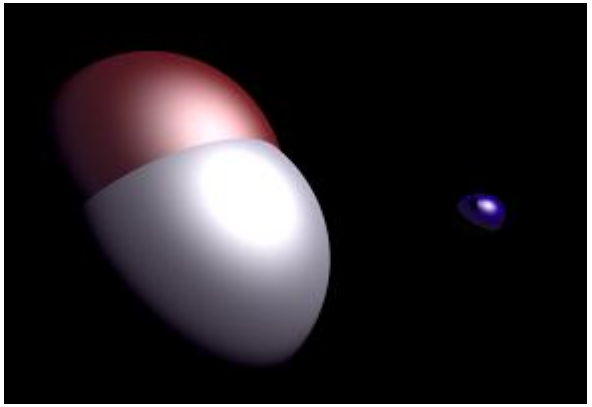
Ray Tracing Homework Results

Kirill Sevastyanenko

Interesting scenes and shapes!



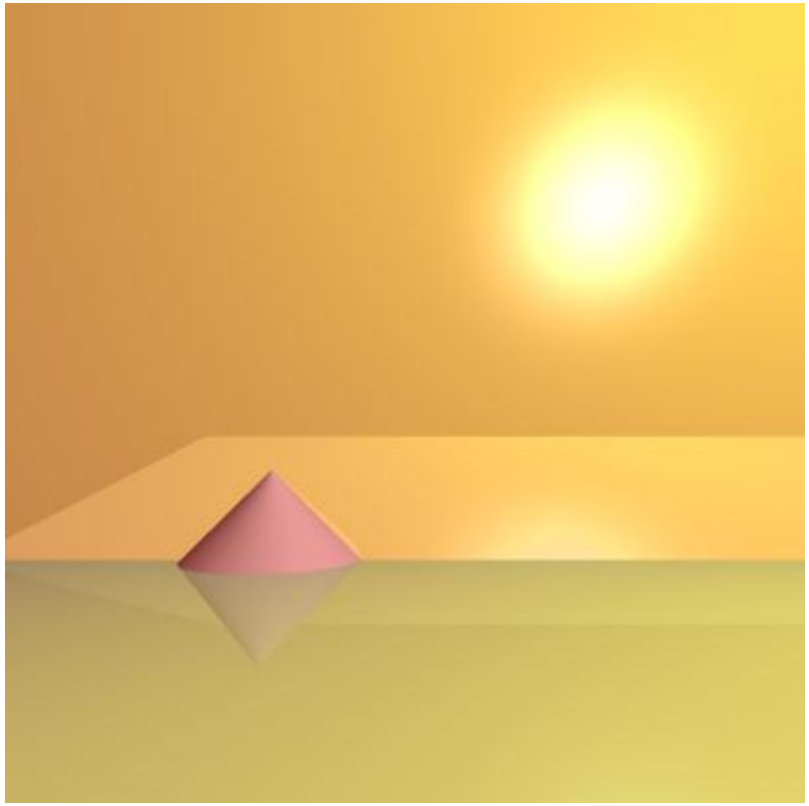


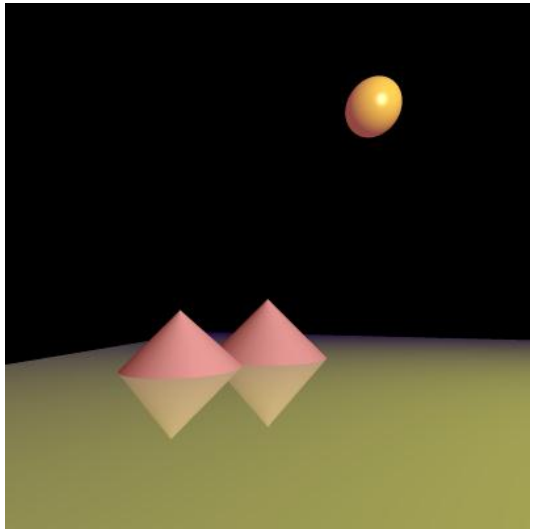


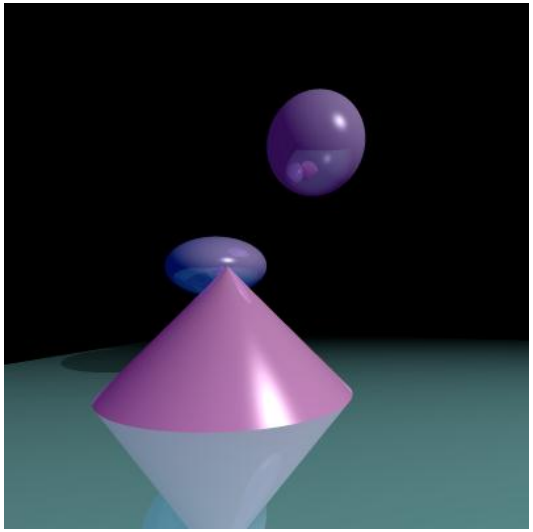
Ray Tracing Homework Results

Lilith Wu

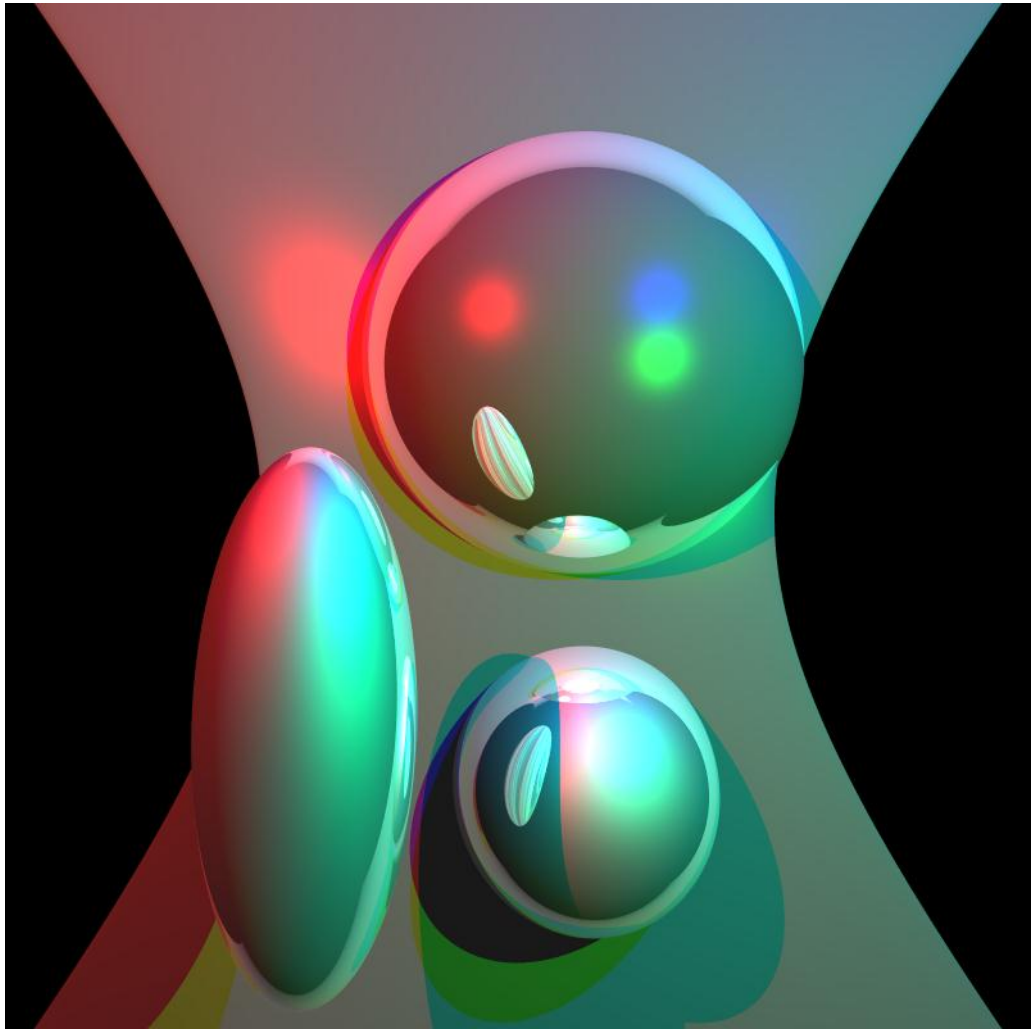
"Two islands in a lake at sunset"





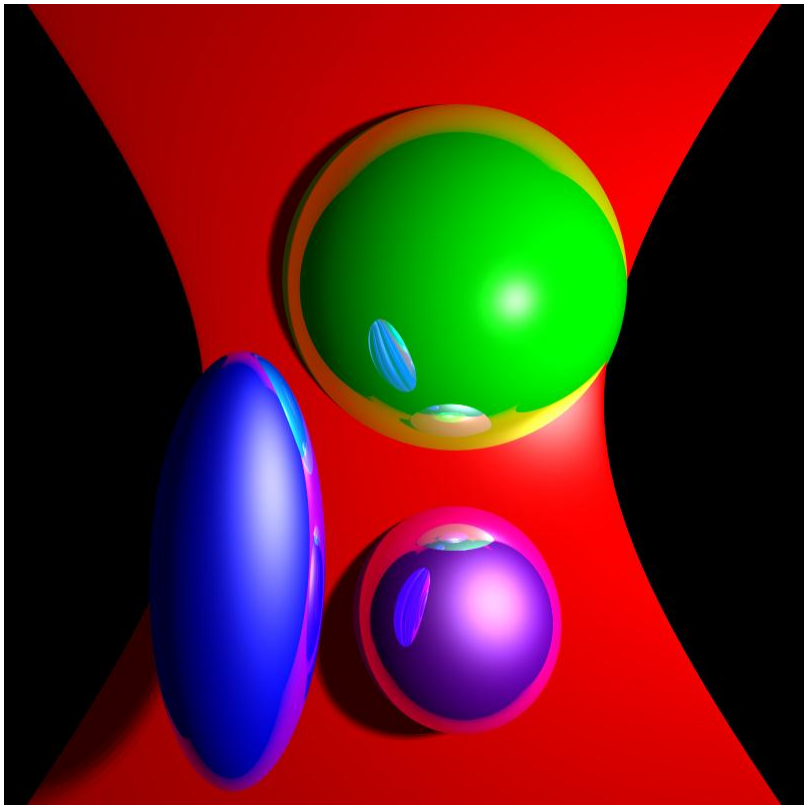


Ray Tracing Homework Results



Nicholas McGee

Multiple lights, soft shadows



</ray_tracing>

```
\begin{homework}
```

Moving Least Squares, SIGGRAPH '06

Image Deformation Using Moving Least Squares

Scott Schaefer*
Texas A&M University

Travis McPhail†
Rice University

Joe Warren‡
Rice University

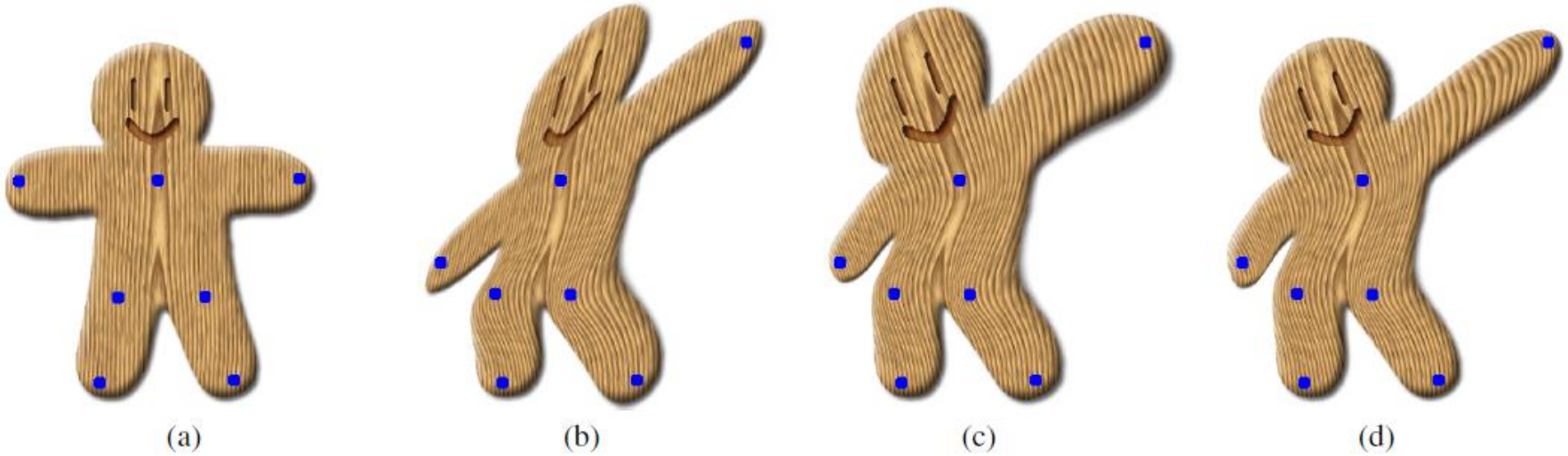


Figure 1: Deformation using Moving Least Squares. Original image with control points shown in blue (a). Moving Least Squares deformations using affine transformations (b), similarity transformations (c) and rigid transformations (d).

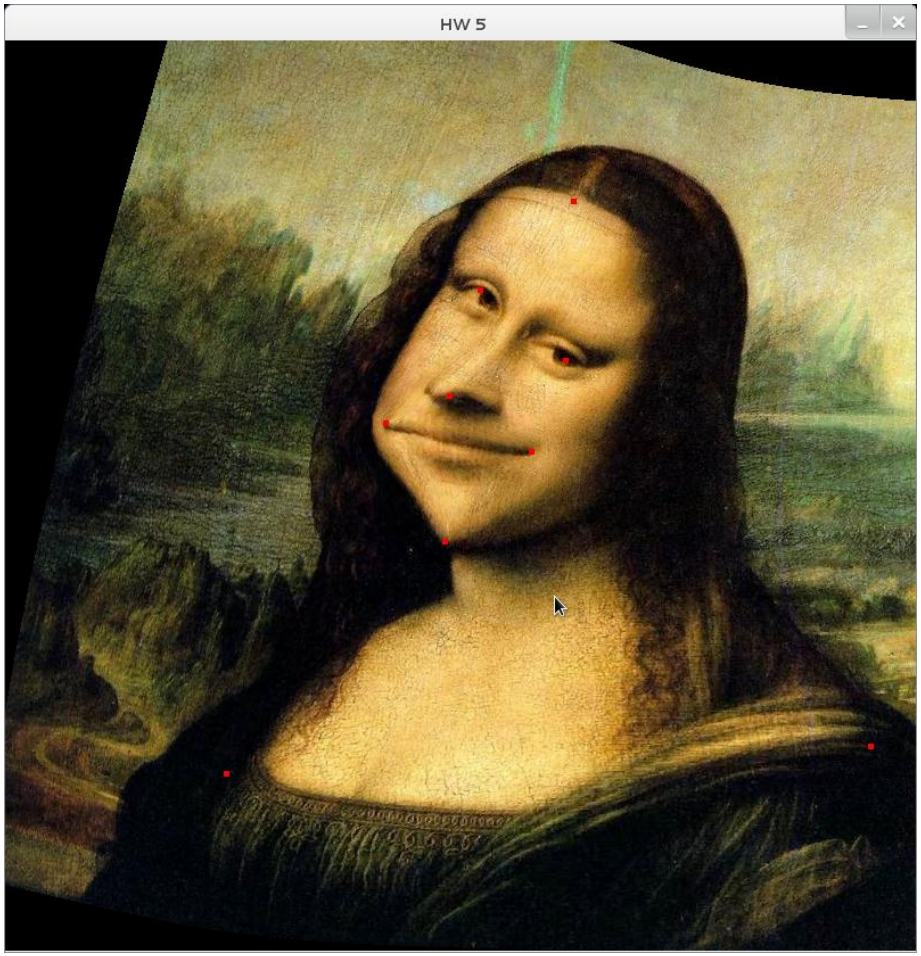
Abstract

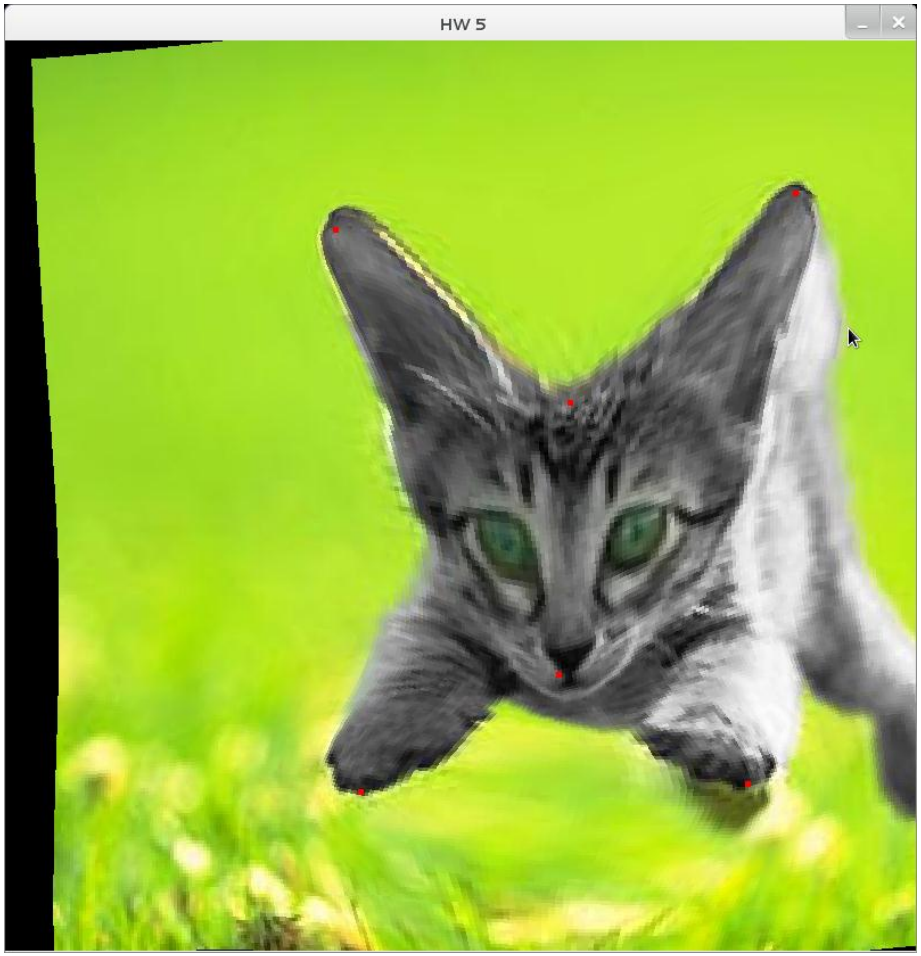
We provide an image deformation method based on Moving Least

the position and orientation of these handles, the image should deform in an intuitive fashion.

We view this deformation as a function f that maps points in the

Moving Least Squares, CS 148





Some Simplifications

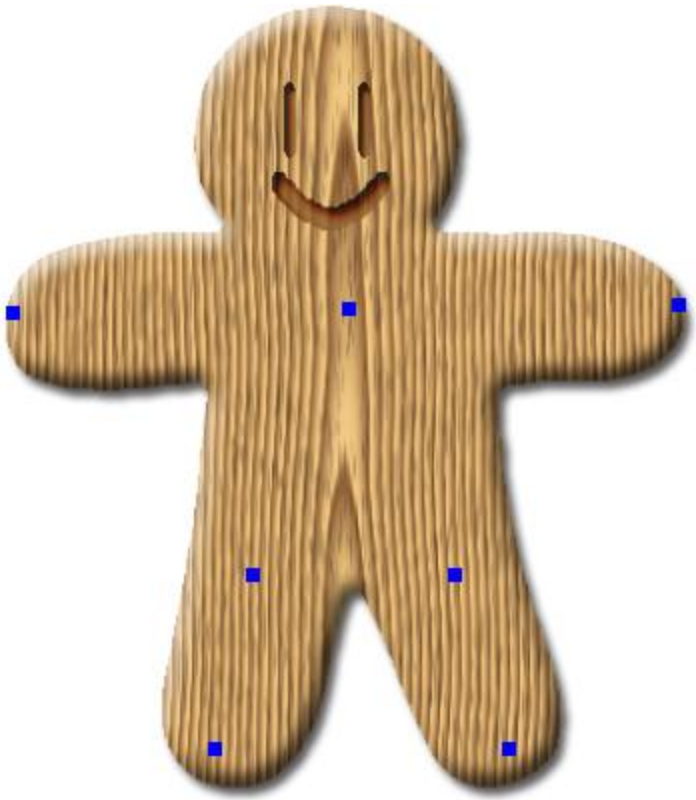
- **Square images**

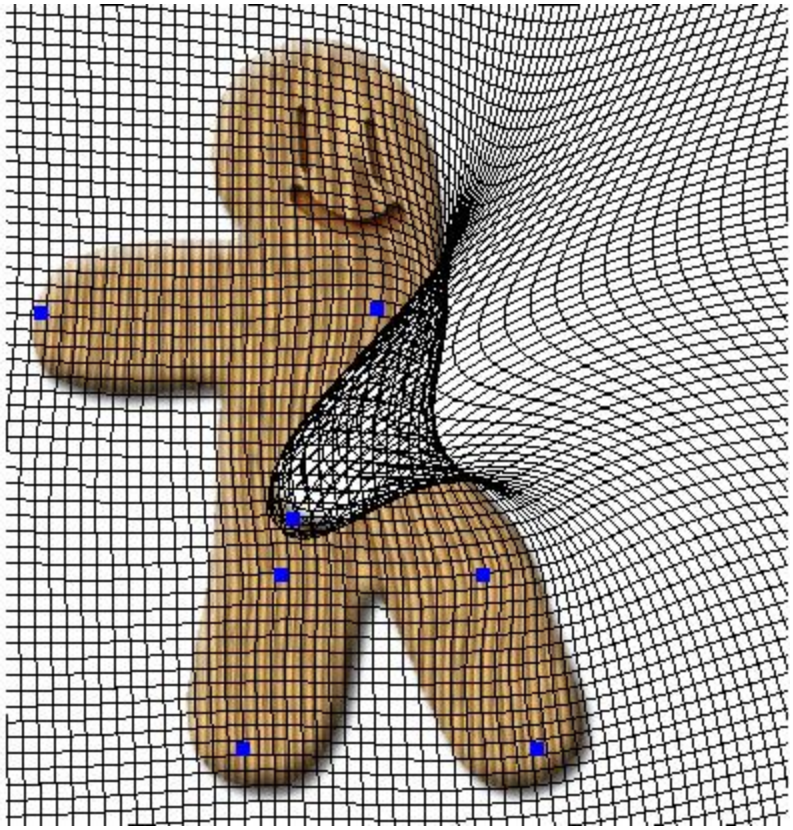
  Simple meshing, distances

- **Only pointwise constraints**

  Cannot fix lines

MLS Interface

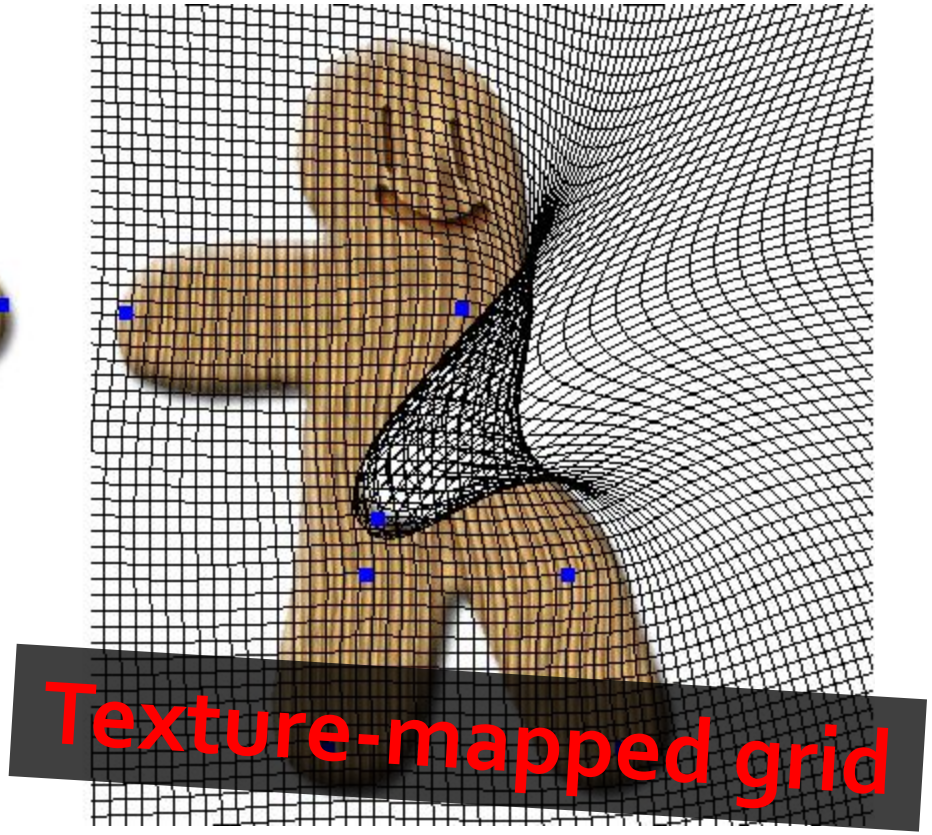




http://faculty.cs.tamu.edu/schaefer/research/mls.pdf

Grab points and move them

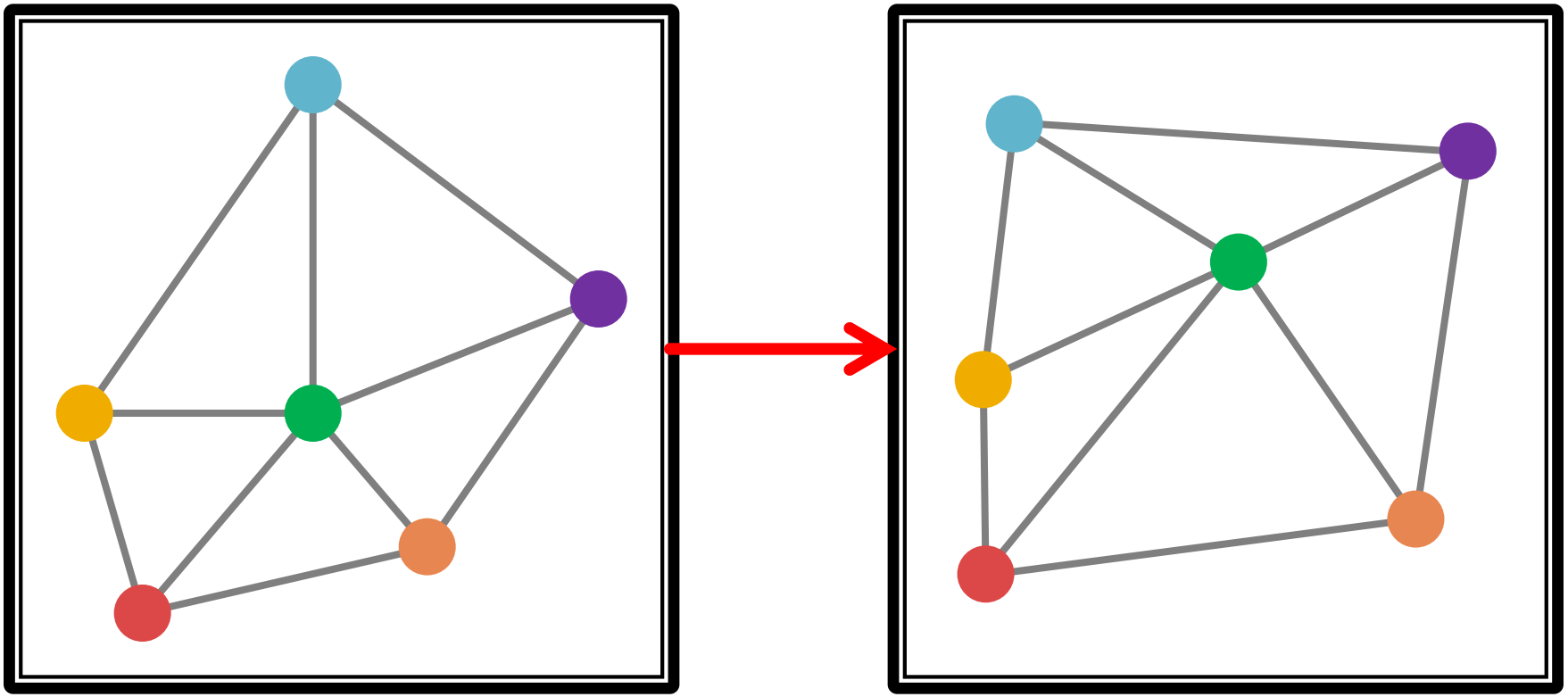
MLS Interface



http://faculty.cs.tamu.edu/schaefer/research/mls.pdf

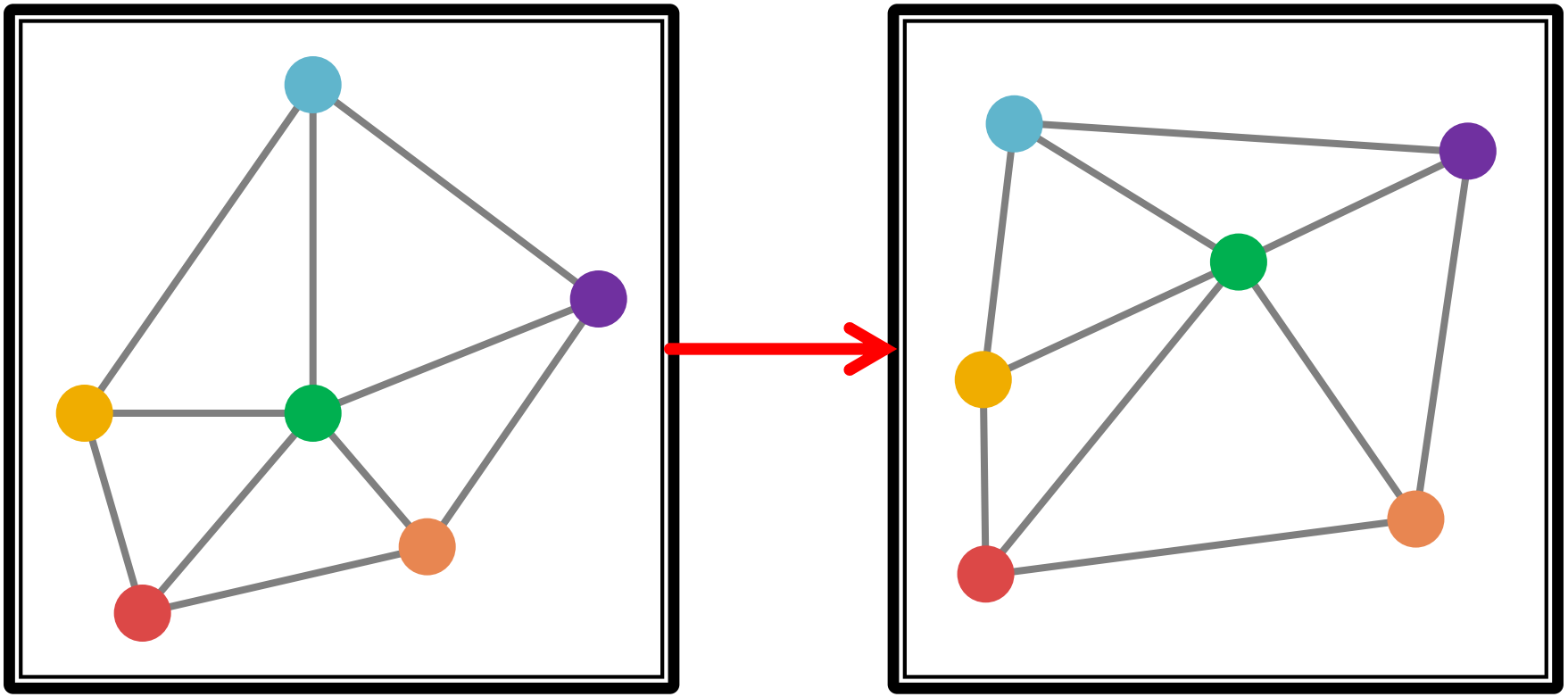
Grab points and move them

Motivating Problem



Want: $\vec{p} \mapsto M\vec{p} + \vec{t}$

Motivating Problem



Want: $\vec{p} \mapsto M\vec{p} + \vec{t}$ *Approximate!*

Affine Maps

$$\vec{p} \mapsto M\vec{p} + \vec{t}$$

$$M \in \mathbb{R}^{2\times 2}$$

$$\vec{t} \in \mathbb{R}^2$$

Alternative to projective math

Affine Maps

4 DOFs

2 DOFs

$$\vec{p} \mapsto M\vec{p} + \vec{t}$$

$$M \in \mathbb{R}^{2\times 2}$$

$$\vec{t} \in \mathbb{R}^2$$

Alternative to projective math

Least-Squares Affine Maps

Constraints

$$\vec{p}_1 \mapsto \vec{q}_1$$

$$\vec{p}_2 \mapsto \vec{q}_2$$

$$\cdots \mapsto \cdots$$

$$\vec{p}_n \mapsto \vec{q}_n$$

$$E(M, \vec{t}) = \sum_{i=1}^{n} w_i \|M\vec{p}_i + \vec{t} - \vec{q}_i\|^2$$

Least-Squares Affine Maps

Constraints

$$\vec{p}_1 \mapsto \vec{q}_1$$

$$\vec{p}_2 \mapsto \vec{q}_2$$

$$\cdots \mapsto \cdots$$

$$\vec{p}_n \mapsto \vec{q}_n$$

$$E(M, \vec{t}) = \sum_{i=1}^{n} w_i \|M\vec{p}_i + \vec{t} - \vec{q}_i\|^2$$

Match quality

Least-Squares Affine Maps

Constraints

$$\vec{p}_1 \mapsto \vec{q}_1$$

$$\vec{p}_2 \mapsto \vec{q}_2$$

$$\cdots \mapsto \cdots$$

$$\vec{p}_n \mapsto \vec{q}_n$$

$$E(M, \vec{t}) = \sum_{i=1}^{n} w_i \|M\vec{p}_i + \vec{t} - \vec{q}_i\|^2$$

How much we care about point *i*

Least-Squares Affine Maps

Constraints

$$\vec{p}_1 \mapsto \vec{q}_1$$

$$\vec{p}_2 \mapsto \vec{q}_2$$

$$\cdots \mapsto \cdots$$

$$\vec{p}_n \mapsto \vec{q}_n$$

$$E(M, \vec{t}) = \sum_{i=1}^{n} w_i \|M\vec{p}_i + \vec{t} - \vec{q}_i\|^2$$

Minimize

Least-Squares Affine Maps

Constraints

$$\vec{p}_1 \mapsto \vec{q}_1$$

$$\vec{p}_2 \mapsto \vec{q}_2$$

$$\cdots \mapsto \cdots$$

$$\vec{p}_n \mapsto \vec{q}_n$$

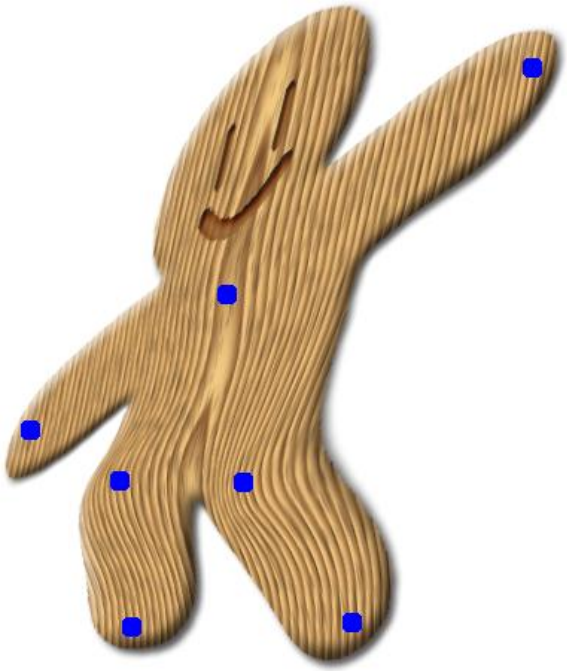
$$E(M, \vec{t}) = \sum_{i=1}^{n} w_i \|M\vec{p}_i + \vec{t} - \vec{q}_i\|^2$$

Minimize

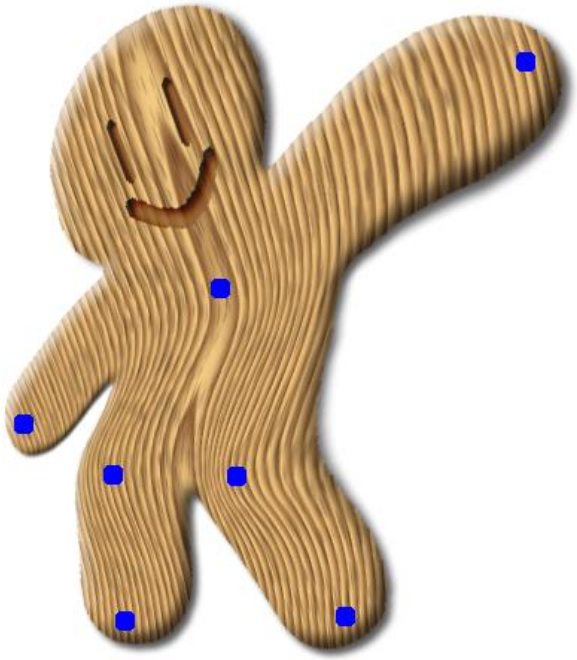
Closed form formula in (5)

Additional Constraints

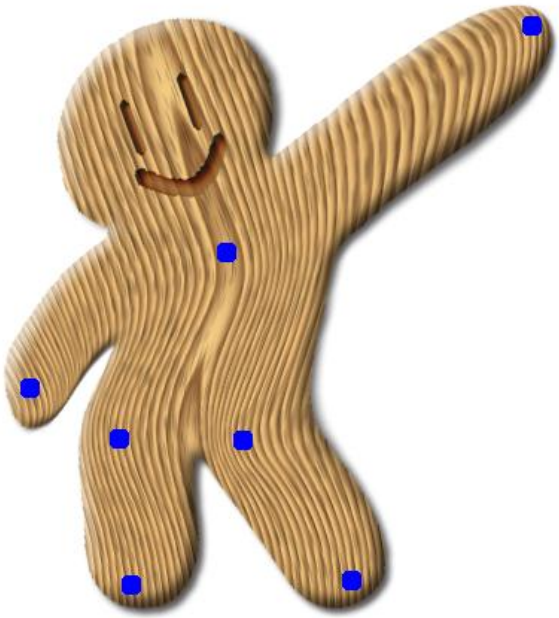
$$\vec{p} \mapsto M\vec{p} + \vec{t}$$



Affine



Similarity



Rigid

http://faculty.cs.tamu.edu/schaefer/research/mls.pdf

Additional Constraints

$$\vec{p} \mapsto M\vec{p} + \vec{t}$$

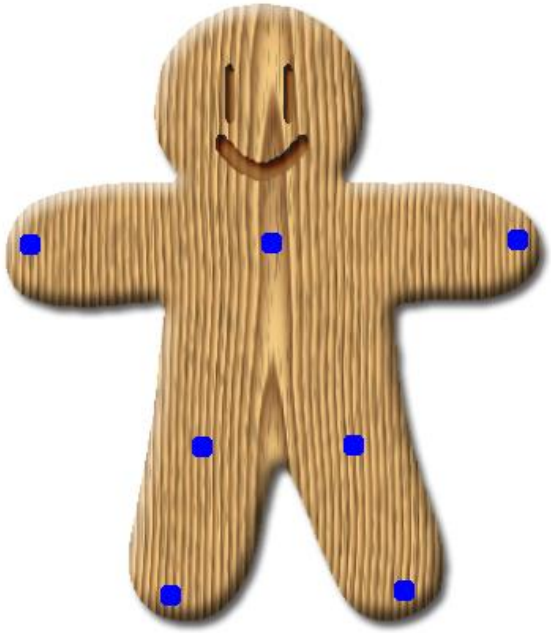
Different closed form in each case

Affine Similarity Rigid

http://faculty.cs.tamu.edu/schaefer/research/mls.pdf

The Trick

$$E(M, \vec{t}) = \sum_{i=1}^{n} w_i \|M\vec{p}_i + \vec{t} - \vec{q}_i\|^2$$

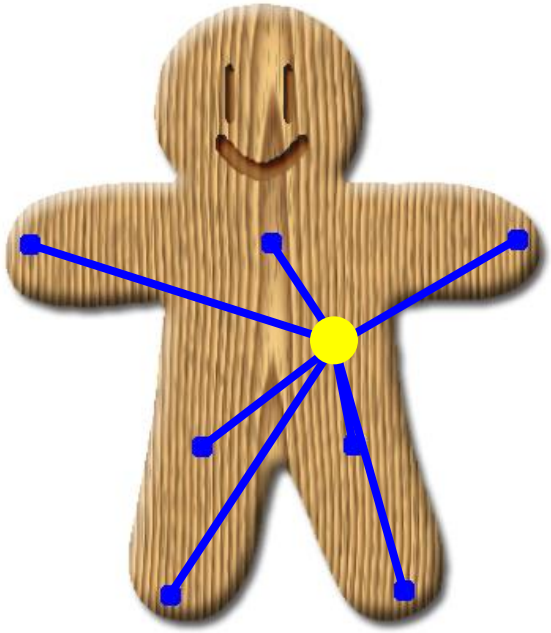


http://faculty.cs.tamu.edu/schaefer/research/mls.pdf

Choose weights w_i cleverly

The Trick

$$E(M,\vec{t}) = \sum_{i=1}^{n} w_i \|M\vec{p}_i + \vec{t} - \vec{q}_i\|^2$$



To find where yellow point *p* goes use weights inversely proportional to distance:

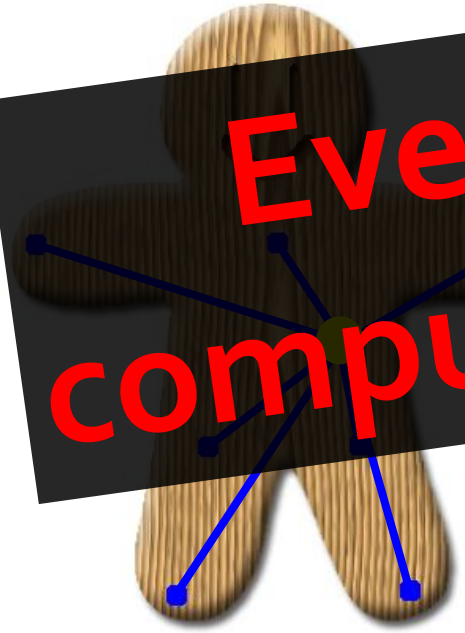
$$w_i = \|\vec{p}_i - \vec{p}\|^{-2\alpha}$$

http://faculty.cs.tamu.edu/schaefer/research/mls.pdf

Choose weights w_i cleverly

The Trick

$$E(M, \vec{t}) = \sum_{i=1}^{n} w_i \|M\vec{p}_i + \vec{t} - \vec{q}_i\|^2$$



Every source point p computes its own M and t.

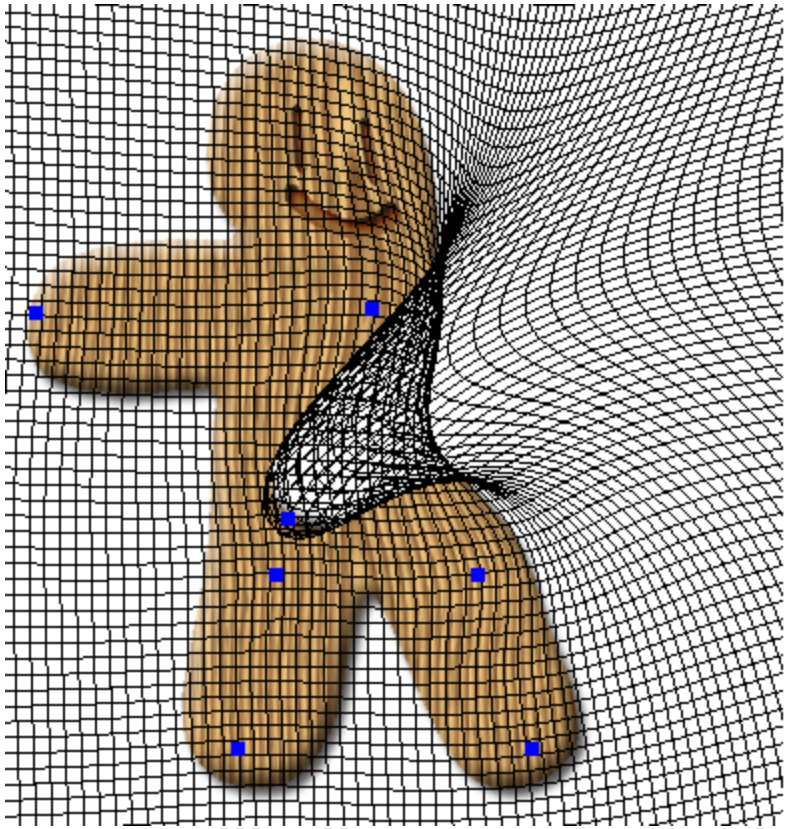
To find where yellow point p goes use weights inversely proportional to distance

$$w_i = \|\vec{p}_i - \vec{p}\|^{-2\alpha}$$

http://faculty.cs.tamu.edu/schaefer/research/mls.pdf

Choose weights w_i cleverly

MLS Algorithm



For each source vertex p:

-If p is a constrained vertex, goes to q.

-Otherwise:

-Compute w_i for p

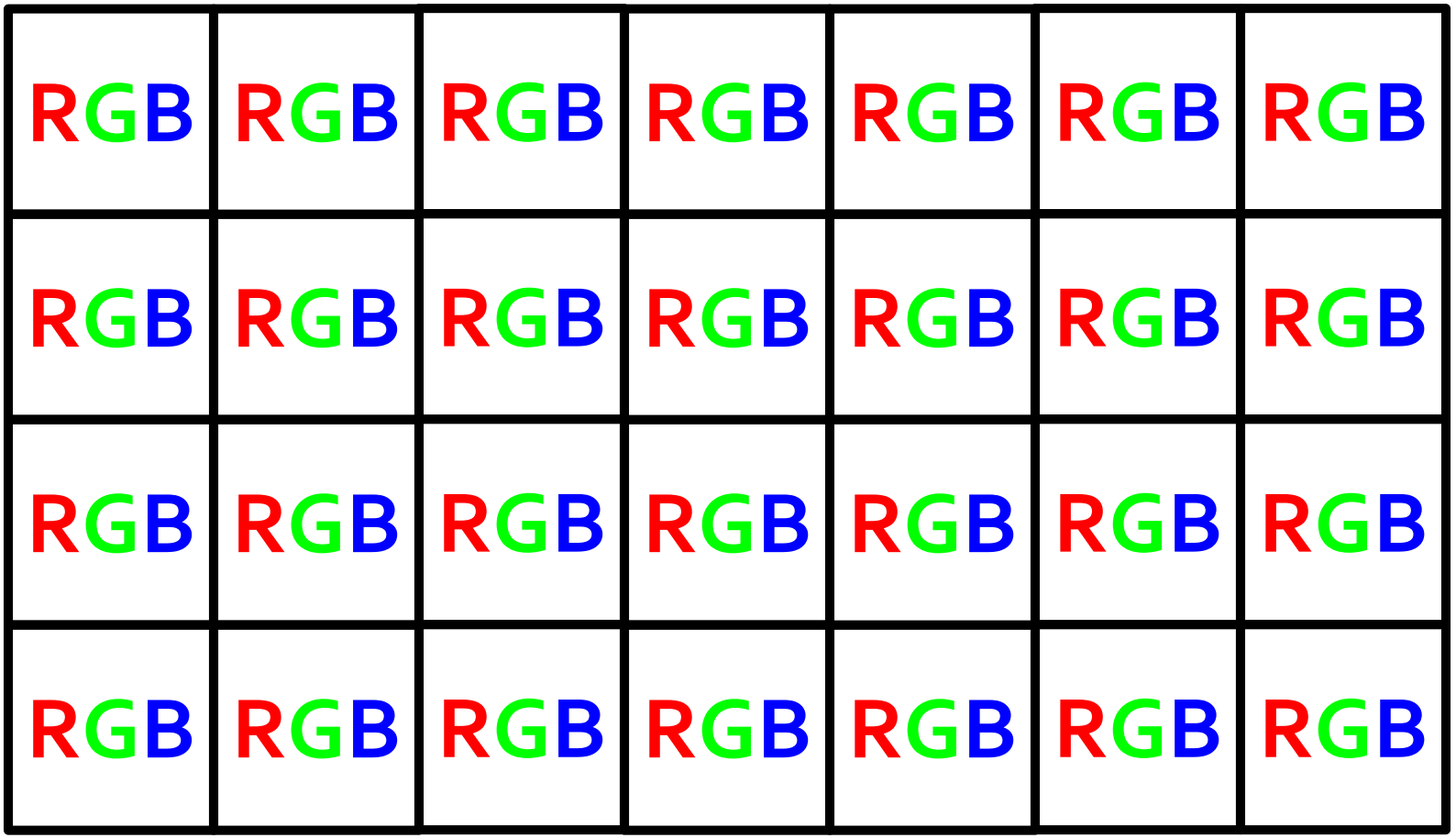
-Compute M and t using these w_i

-Transform p using M and t

http://faculty.cs.tamu.edu/schaefer/research/mls.pdf

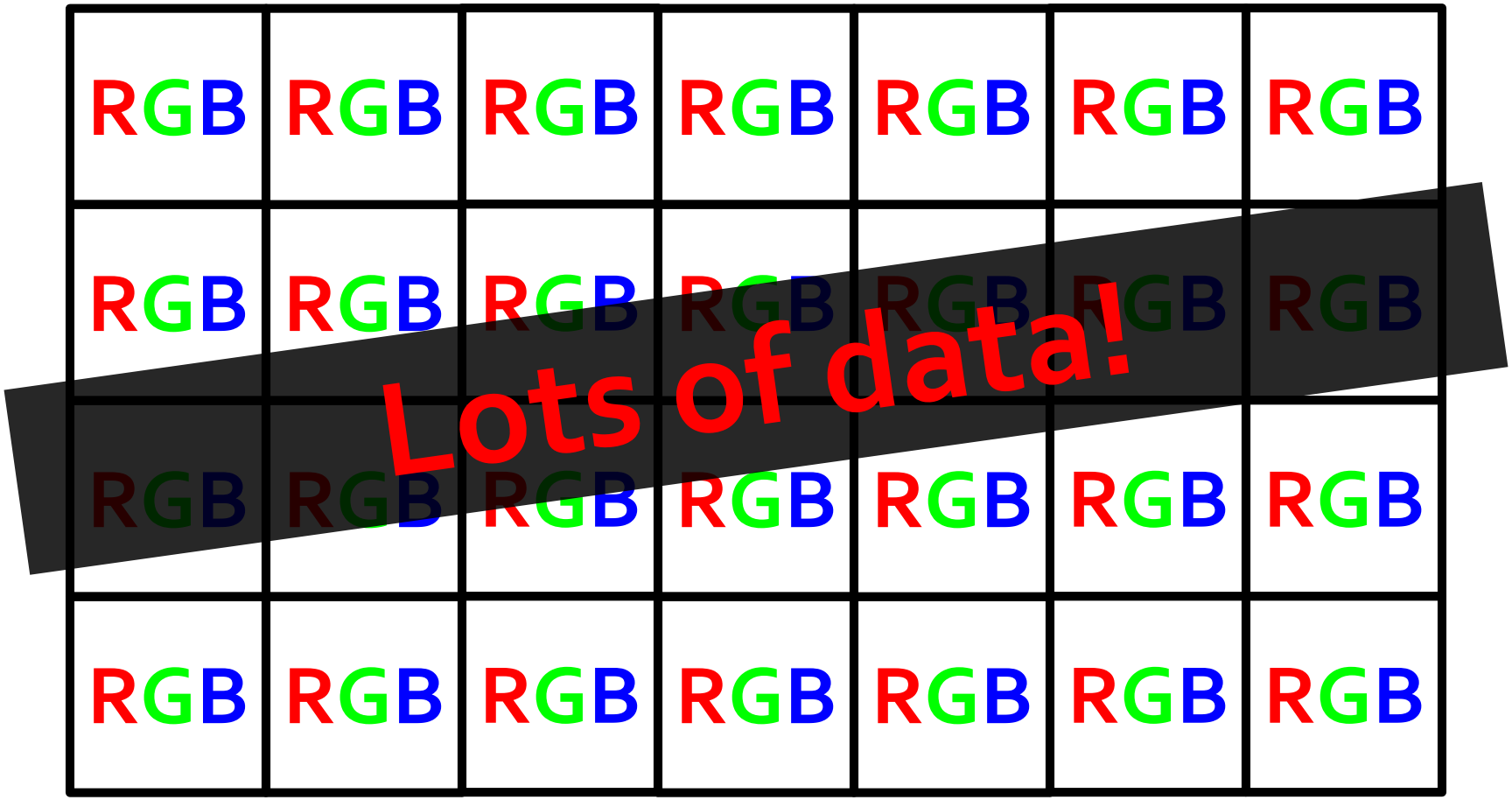
```
\end{homework}
```

Recall: Arrays of What?



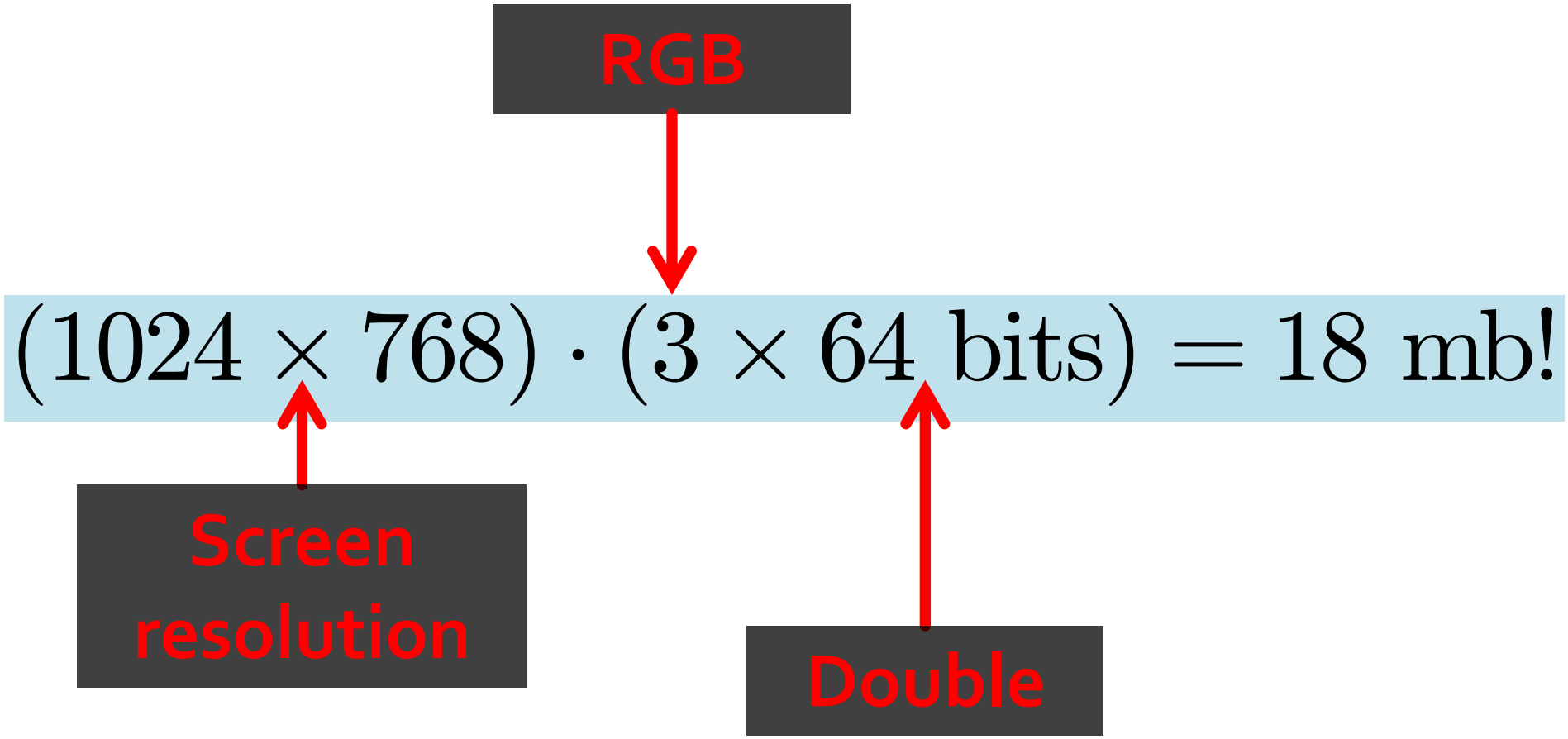
Red/Green/Blue in [0,1]

Recall: Arrays of What?



Red/Green/Blue in [0,1]

Amount of Data in an Image



More Typical Data Rates

Image and Video Data Rates

One full screen Image

- 1024 x 768 x 24b = ~2.5MB

DVD

- 720 x 480 x 24b x 30f/s = ~30 MB/s

High Definition DVD

- 1920 x1080 x 24b x 30f/s = ~178MB/s

Film

- 4000 x 3000 x 36b x 30f/s = ~1.5GB/s
- 8 TB for a 90 minute movie!

Storage is Cheap!

Yes, but we must

- Display at frame rate
- Transfer over the internet
- Move between devices
- Send to mobile devices

Storage is Cheap!

Yes, but we must

- Display at frame rate
- Transfer over the internet
- Move between devices
- Send to mobile devices

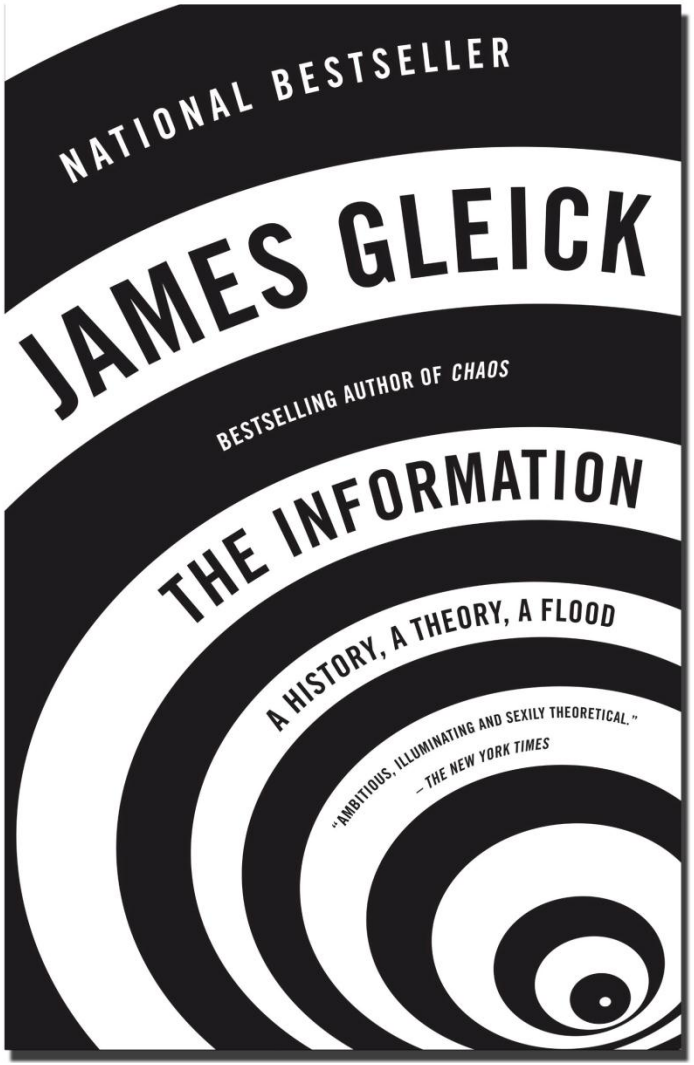
Compression won't be irrelevant any time soon.

Lossless [los-lis]:

No information is lost during compression; the original can be reconstructed exactly.

Lossy [los-ee]:
Some information is lost during compression to achieve a smaller size.

Information Theory



Amount of information in an image isn't necessarily the same as the number of pixels!

Entropy [**en**-tr*uh*-pee]:
Average missing information content when one does not know the value of a random variable.

Entropy in English Language

Four score and seven years ago our fathers brought forth on this continent a new nation, conceived in liberty, and dedicated to the proposition that all men are created equal.

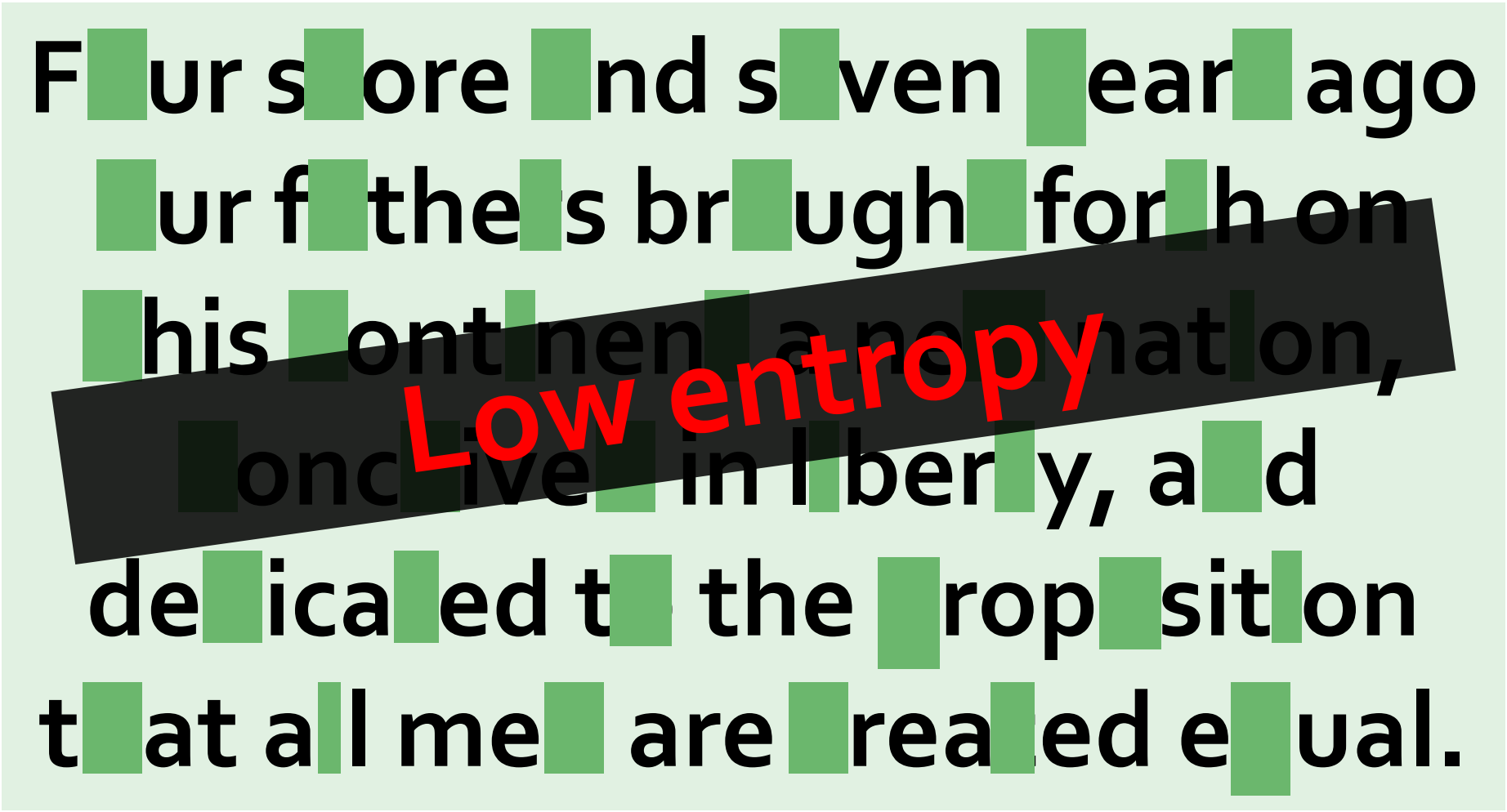
Gettysburg Address, Abraham Lincoln (1863)

Entropy in English Language

F ur s ore nd s ven ear ago
ur f the s br ugh for h on
his ont nen a ne nat on,
onc ive in l ber y, a d
de ica ed t the rop sit on
t at a l me are rea ed e ual.

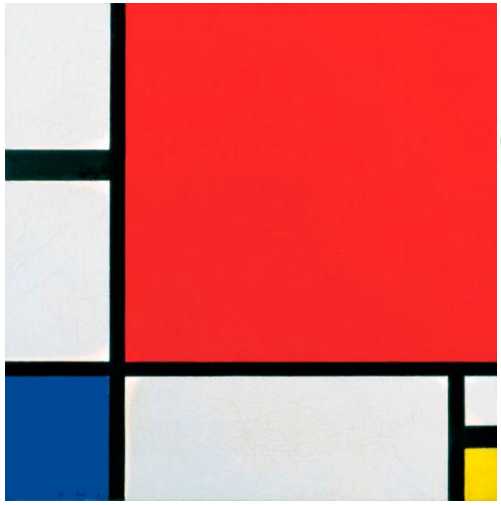
Gettysburg Address, Abraham Lincoln (1863)

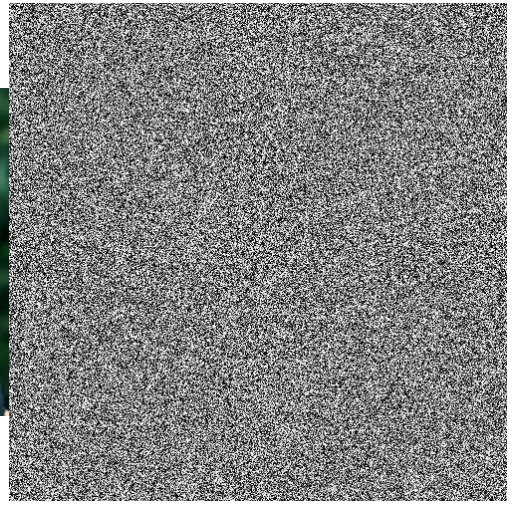
Entropy in English Language



Gettysburg Address, Abraham Lincoln (1863)

Image Entropy





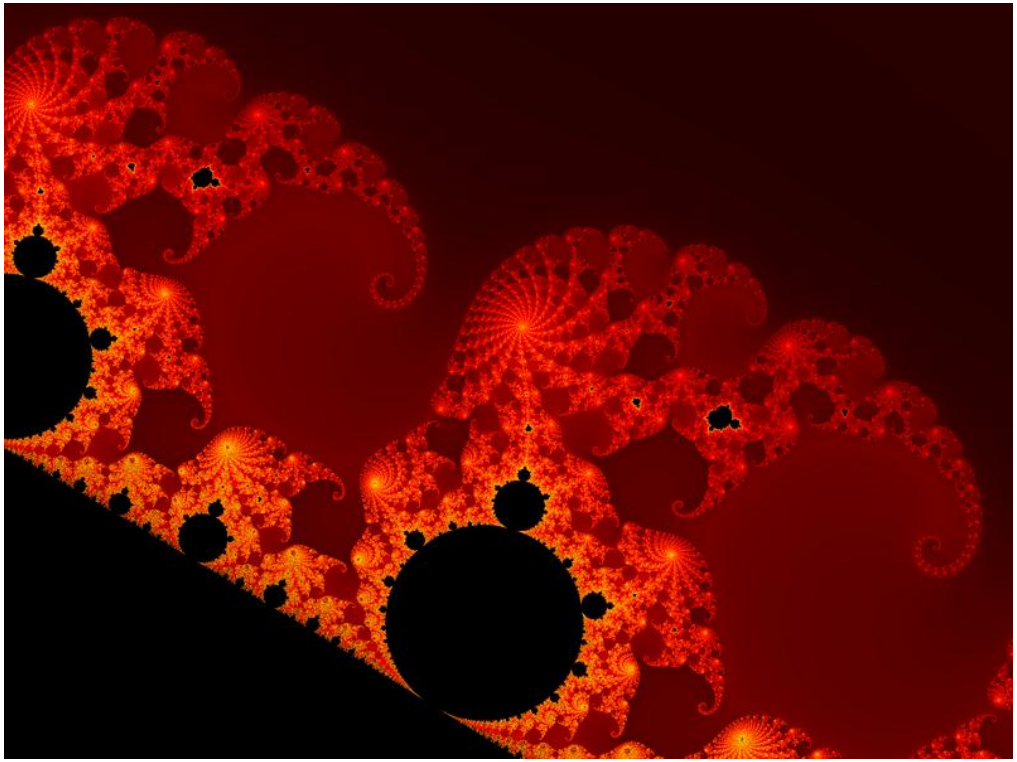
Increasing entropy

http://www.almanart.com/IMG/Image/permanentes4/mondrian-piet_composition_1930_Holtzman-Trust.jpg
http://media.photobucket.com/image/recent/phoenixeous/SmurfIceCream.jpg
http://cdn.smosh.com/sites/default/files/bloguploads/rebecca-black-hipster.png
http://www.quasimondo.com/hydra/sineNoise1.jpg

Aside: Kolmogorov Complexity

What is the shortest program to generate a given piece of data?

Aside: Kolmogorov Complexity



```
z = z0 = Complex(x,y)
for(i=0;i<1000;i++)
    z = z*z + c
setpixel(x,y, |z-z0|)
```

http://graphics.stanford.edu/courses/cs148-11-fall/lectures/compression.pdf

Idea of Lossless Compression

Exchange more entropy (information per bit) for less space.

Also known as:

- Source coding
- Bit-rate reduction

Eliminate statistical redundancy

Huffman Coding

Letter	Frequency
A	0.1
B	0.15
C	0.3
D	0.16
E	0.29

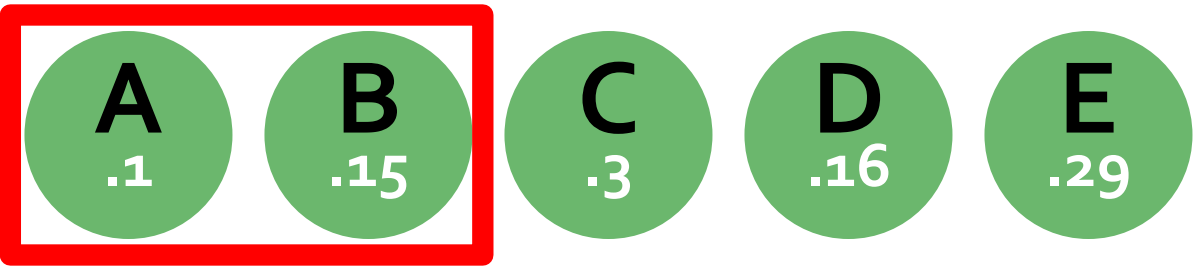
Huffman Coding

Letter	Frequency
A	0.1
B	0.15
C	0.3
D	0.16
E	0.29

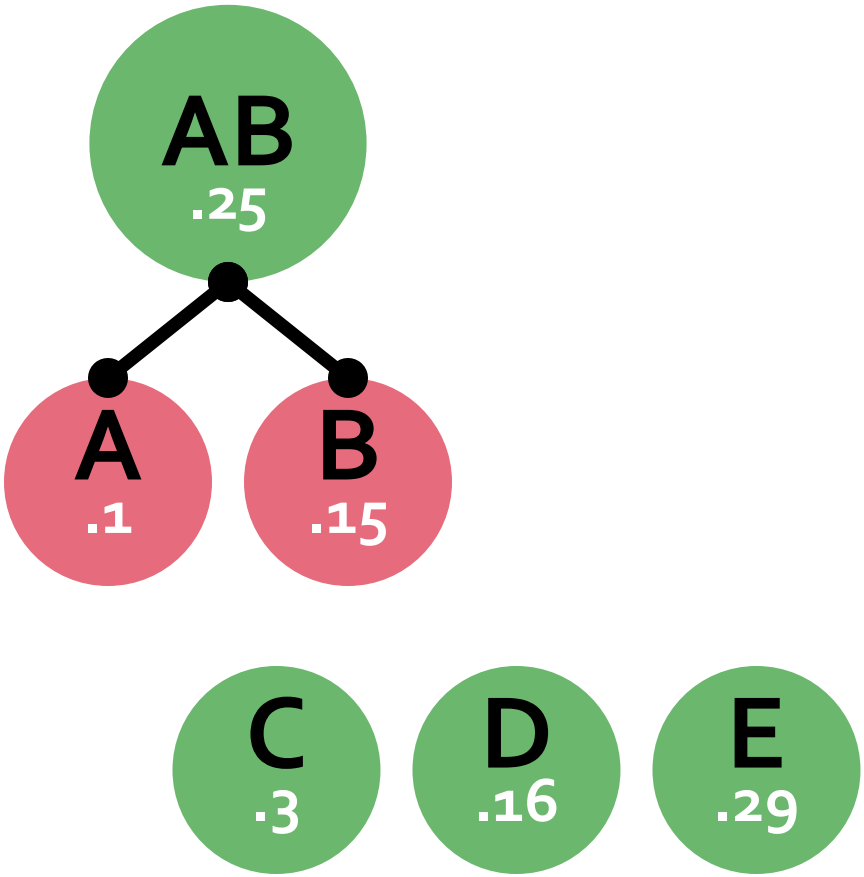
Huffman Coding

Letter	Frequency
A	0.1
B	0.15
C	0.3
D	0.16
E	0.29

Smallest values

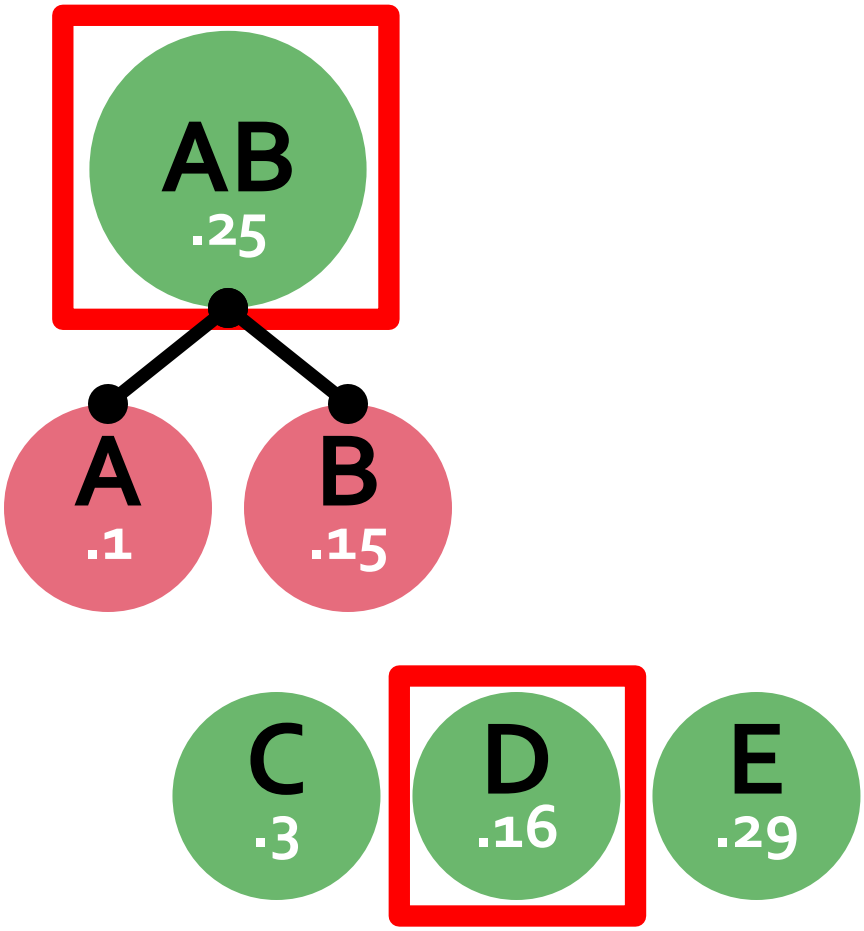


Huffman Coding



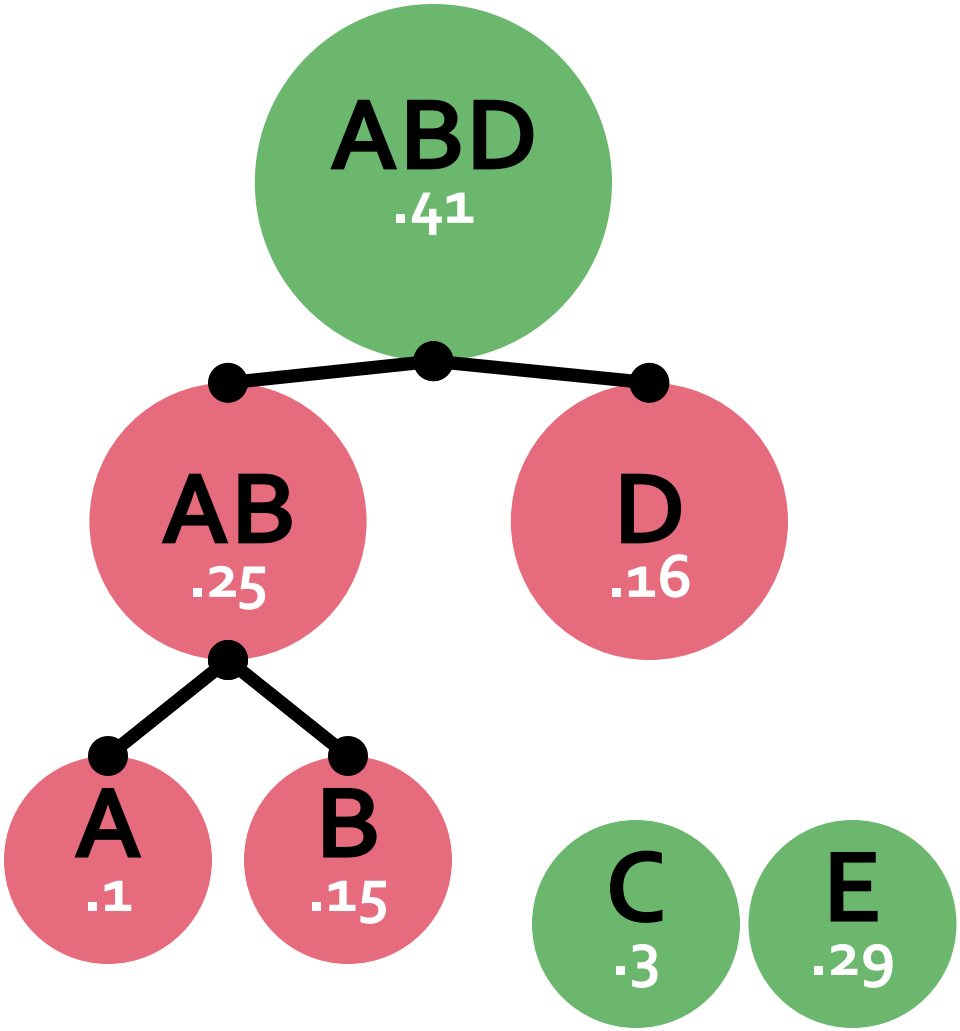
Letter	Frequency
AB	0.25
C	0.3
D	0.16
E	0.29

Huffman Coding



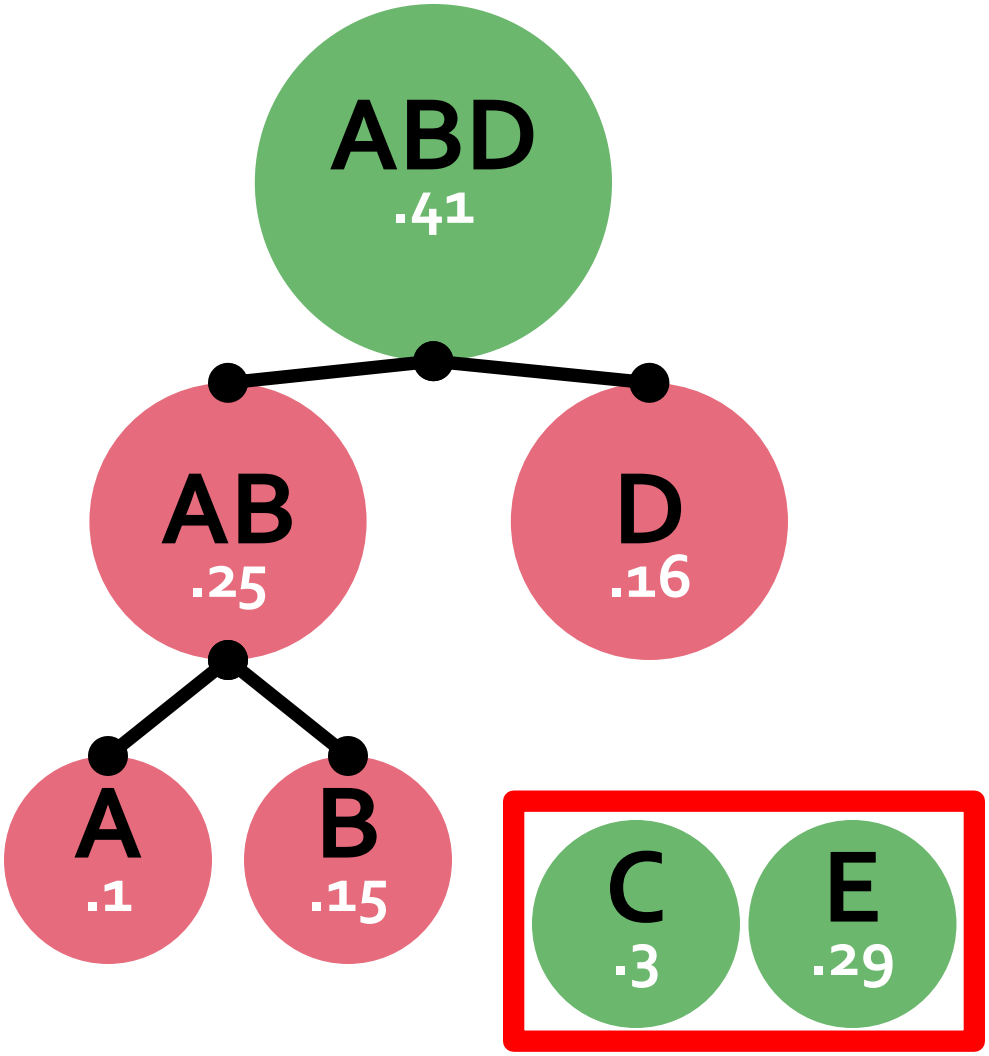
Letter	Frequency
AB	0.25
C	0.3
D	0.16
E	0.29

Huffman Coding



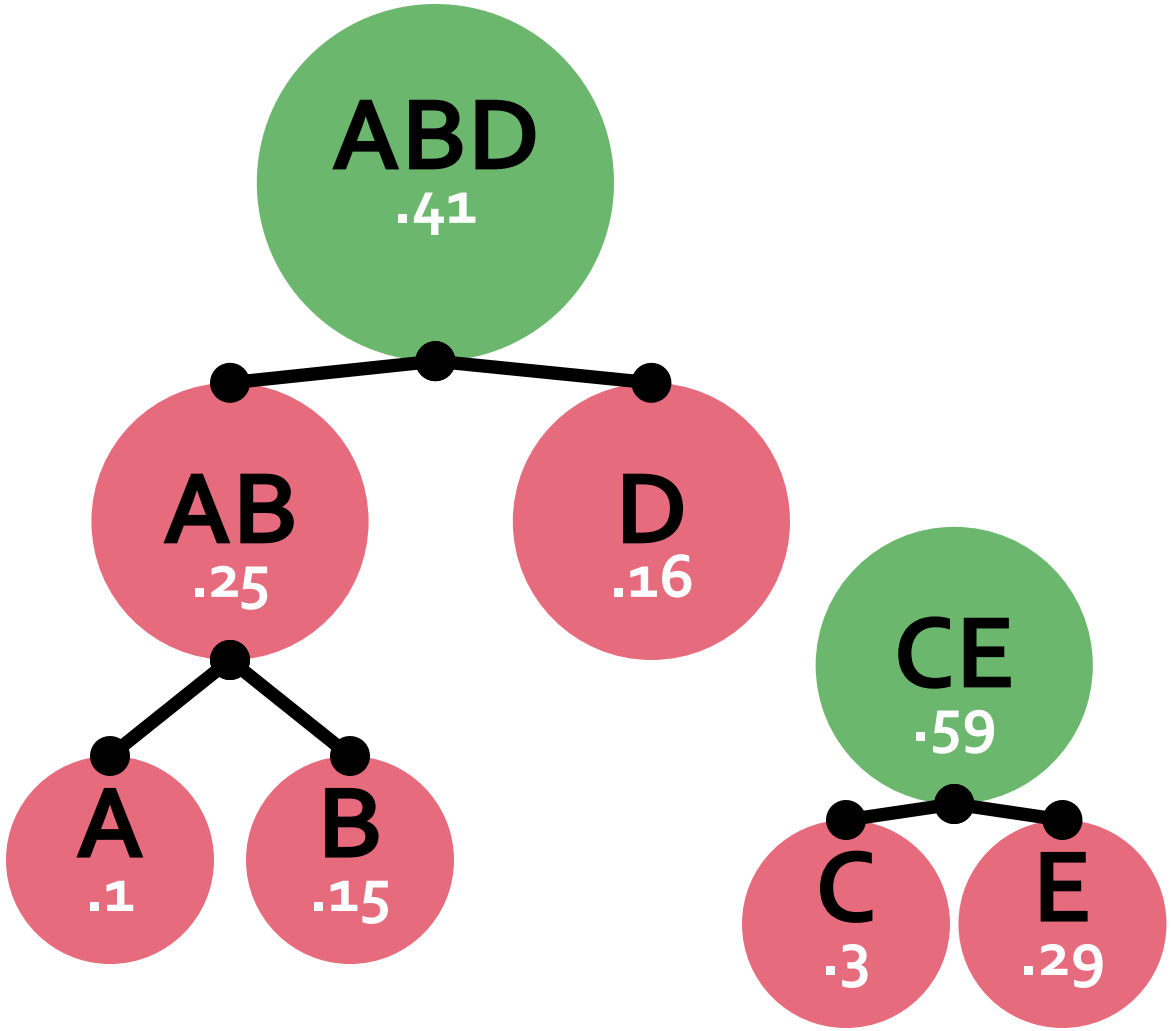
Letter	Frequency
ABD	0.41
C	0.3
E	0.29

Huffman Coding



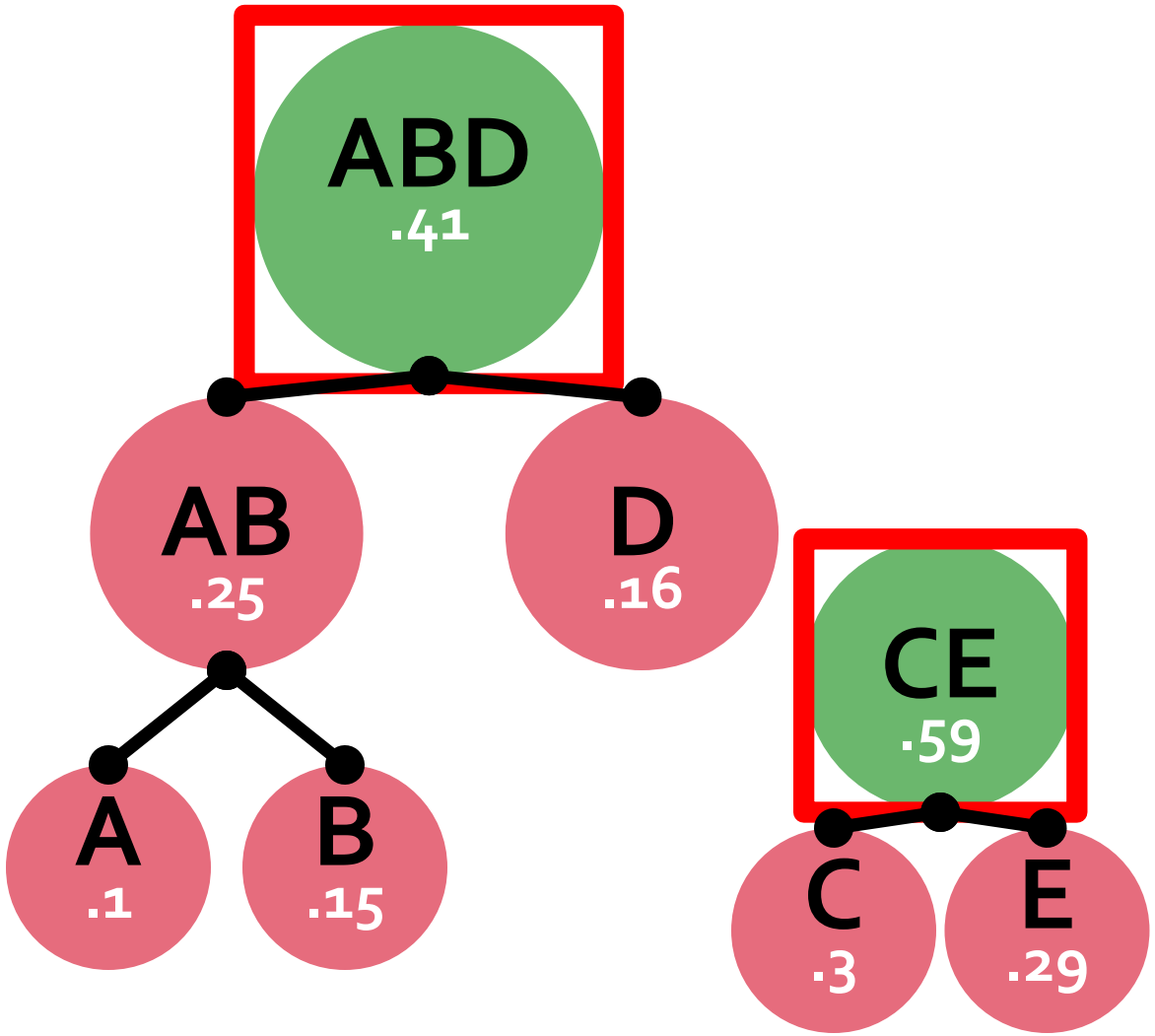
Letter	Frequency
ABD	0.41
C	0.3
E	0.29

Huffman Coding



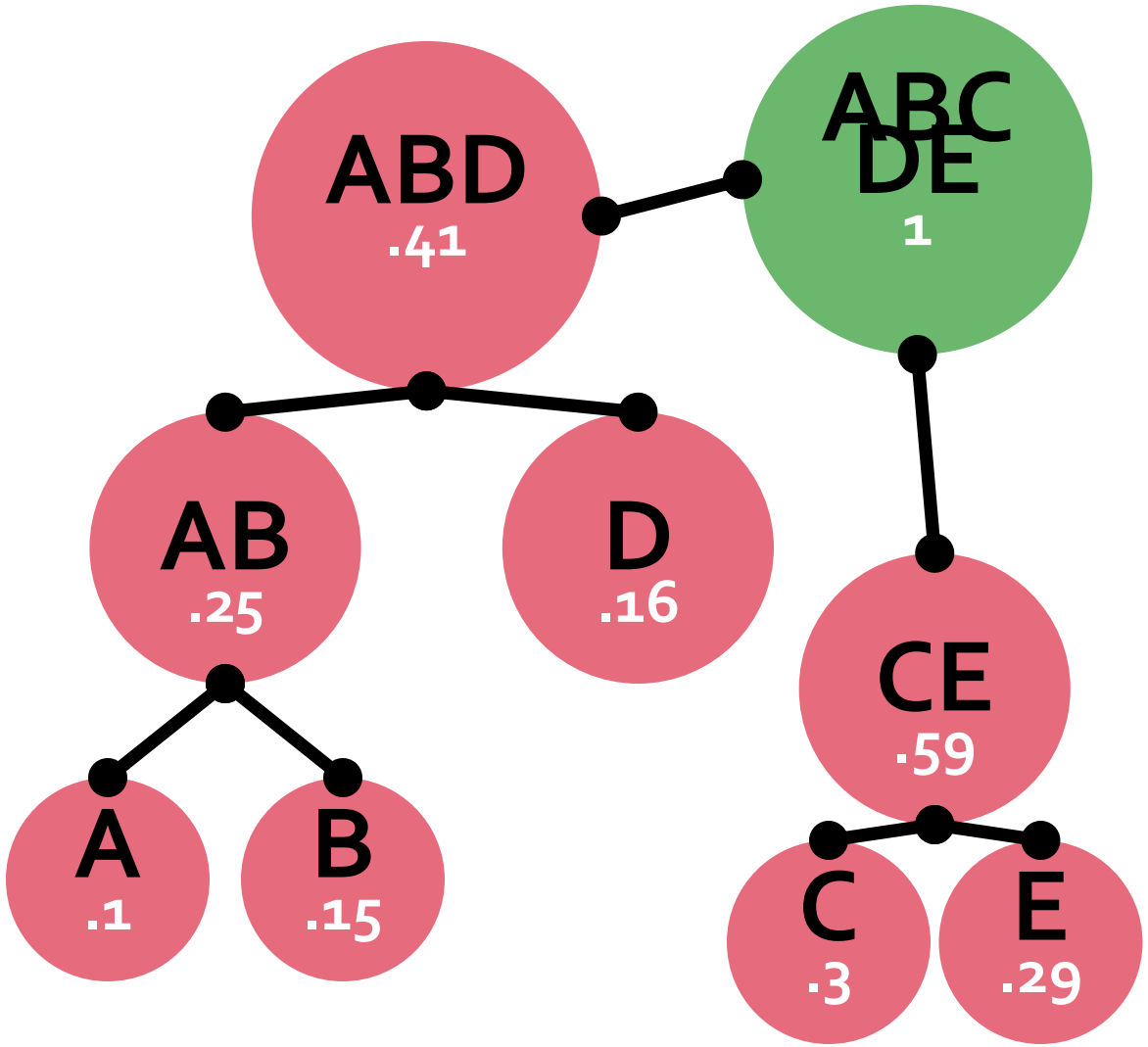
Letter	Frequency
ABD	0.41
CE	0.59

Huffman Coding



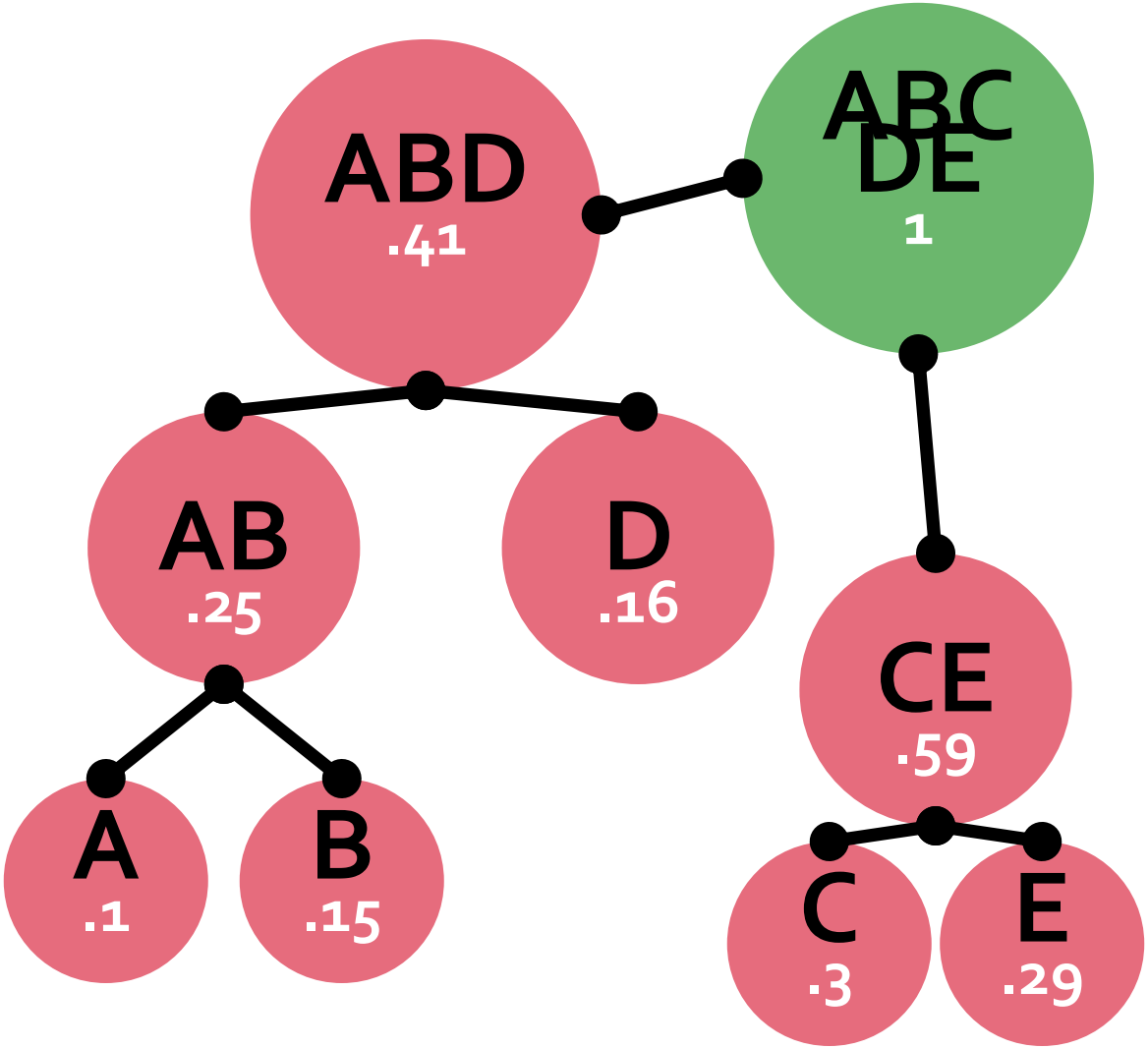
Letter	Frequency
ABD	0.41
CE	0.59

Huffman Coding

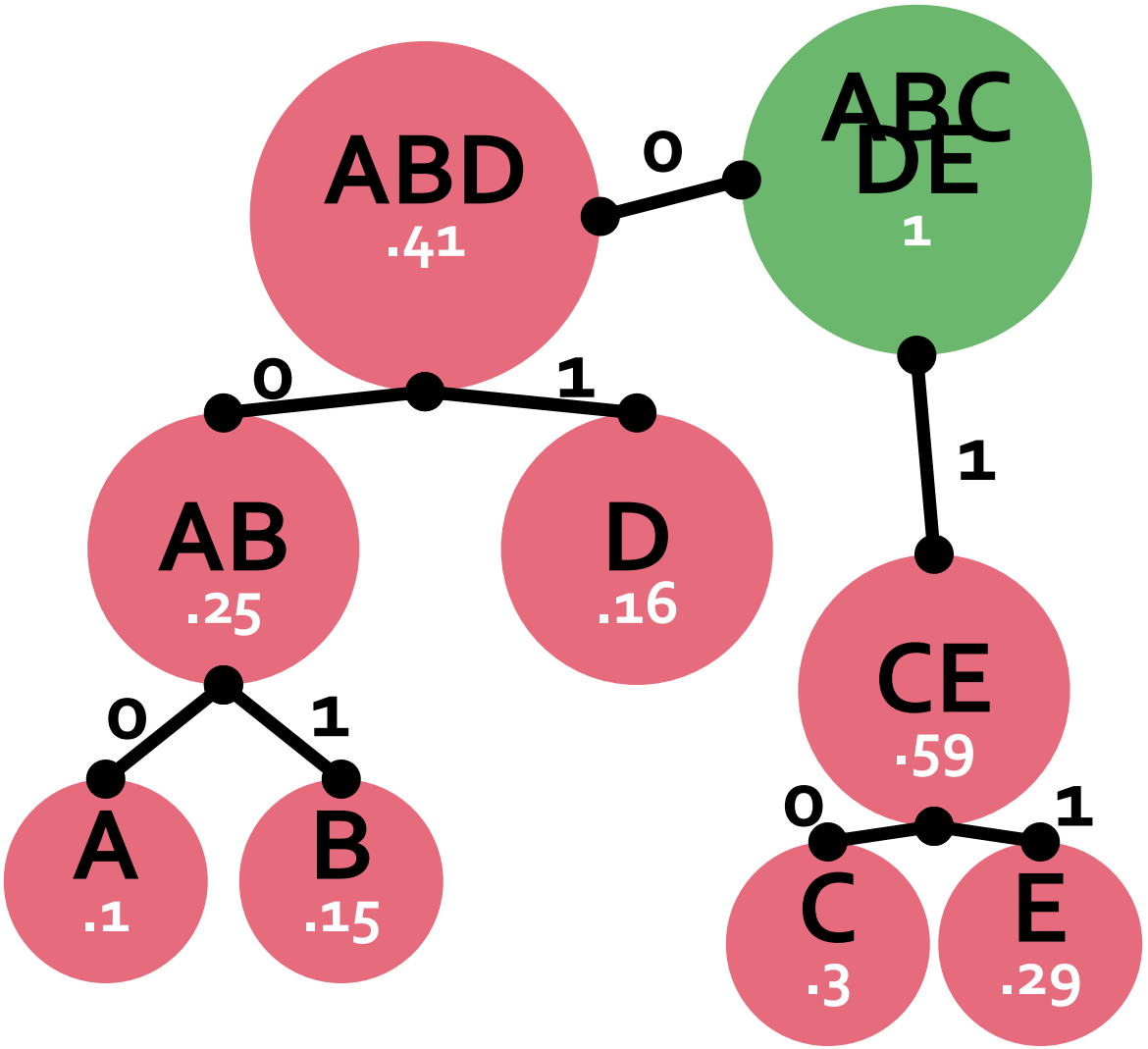


Letter	Frequency
ABCDE	1

Huffman Coding

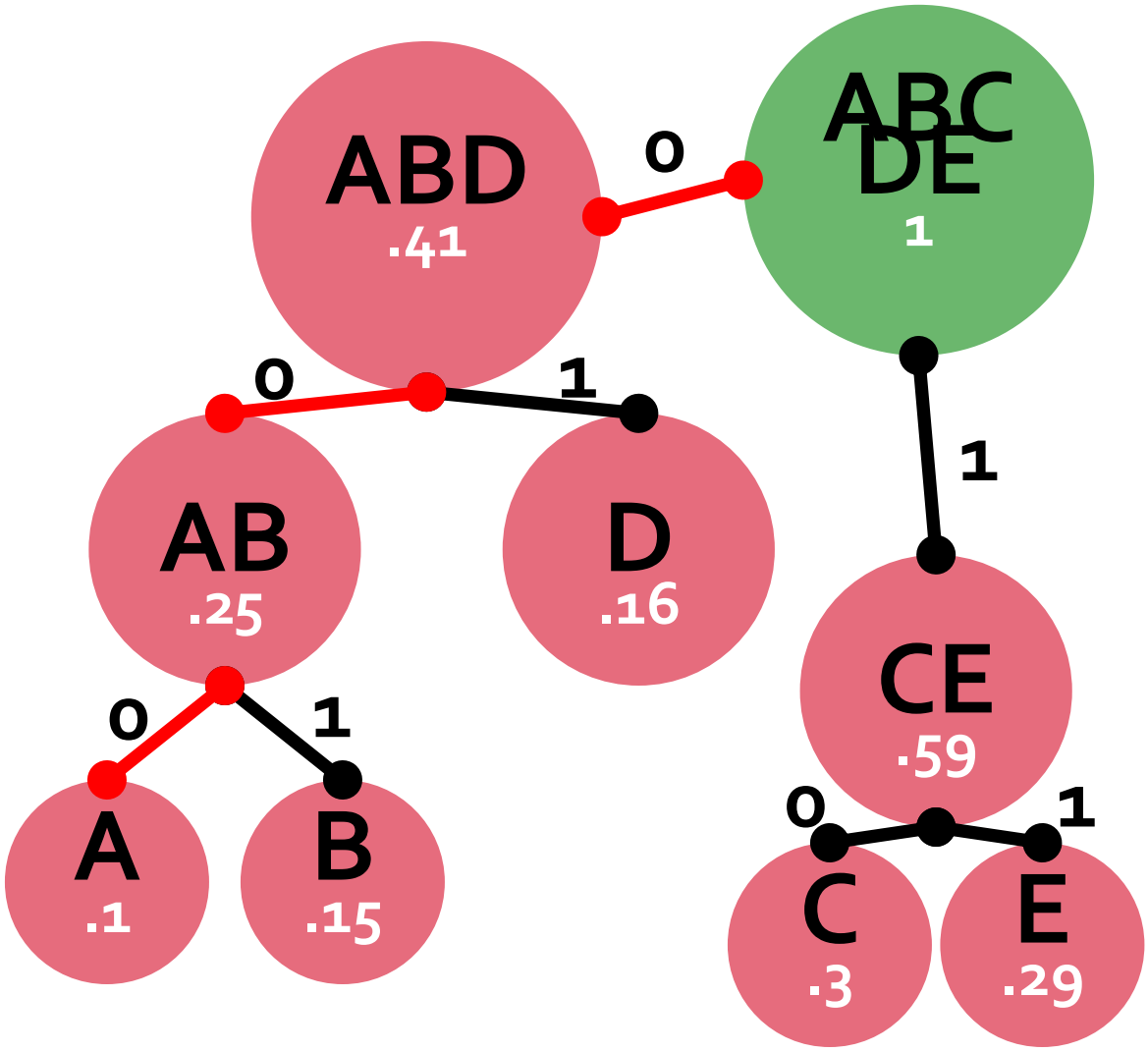


Huffman Coding



Huffman Coding

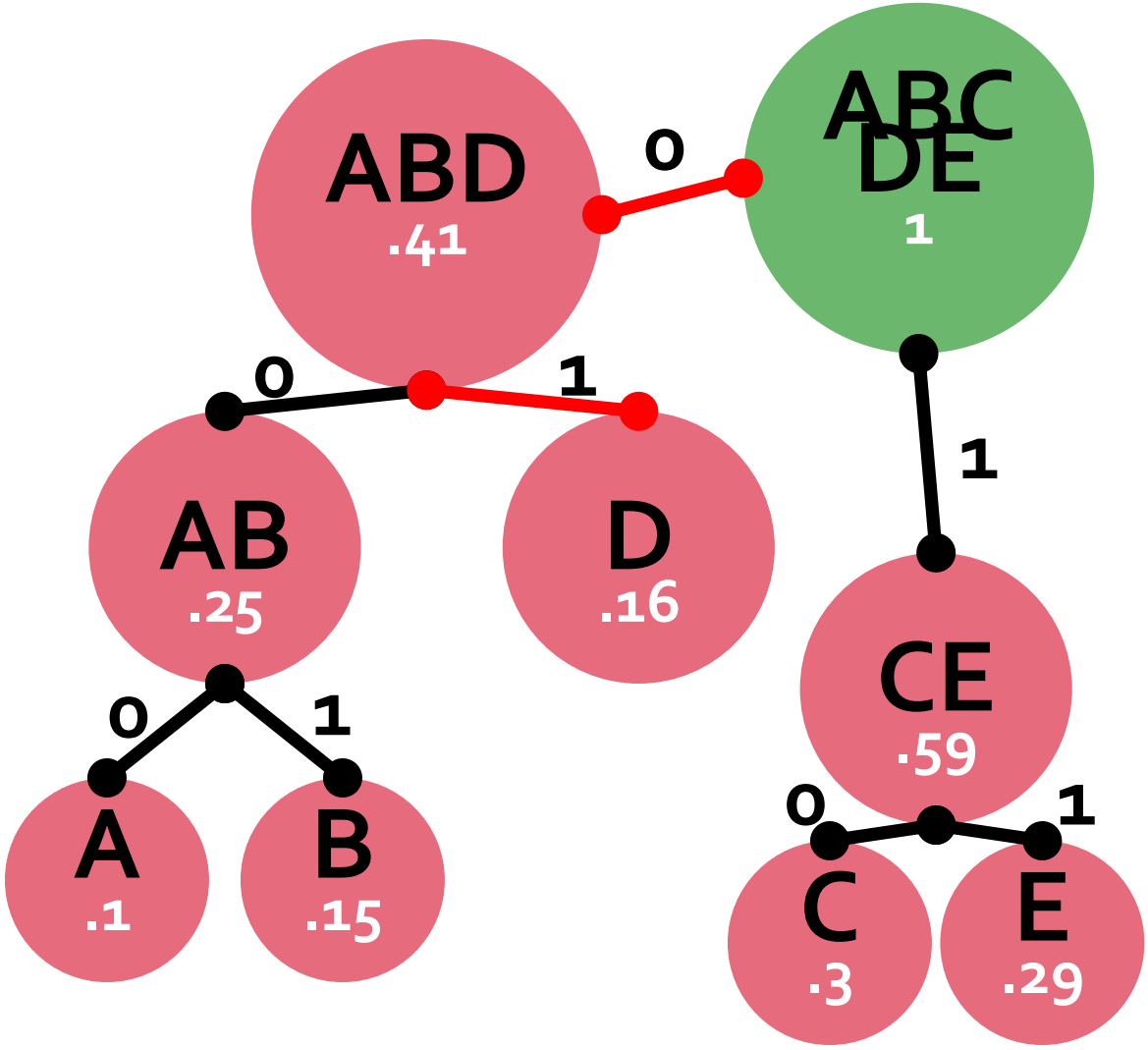
A=000



Huffman Coding

A=000

D=01

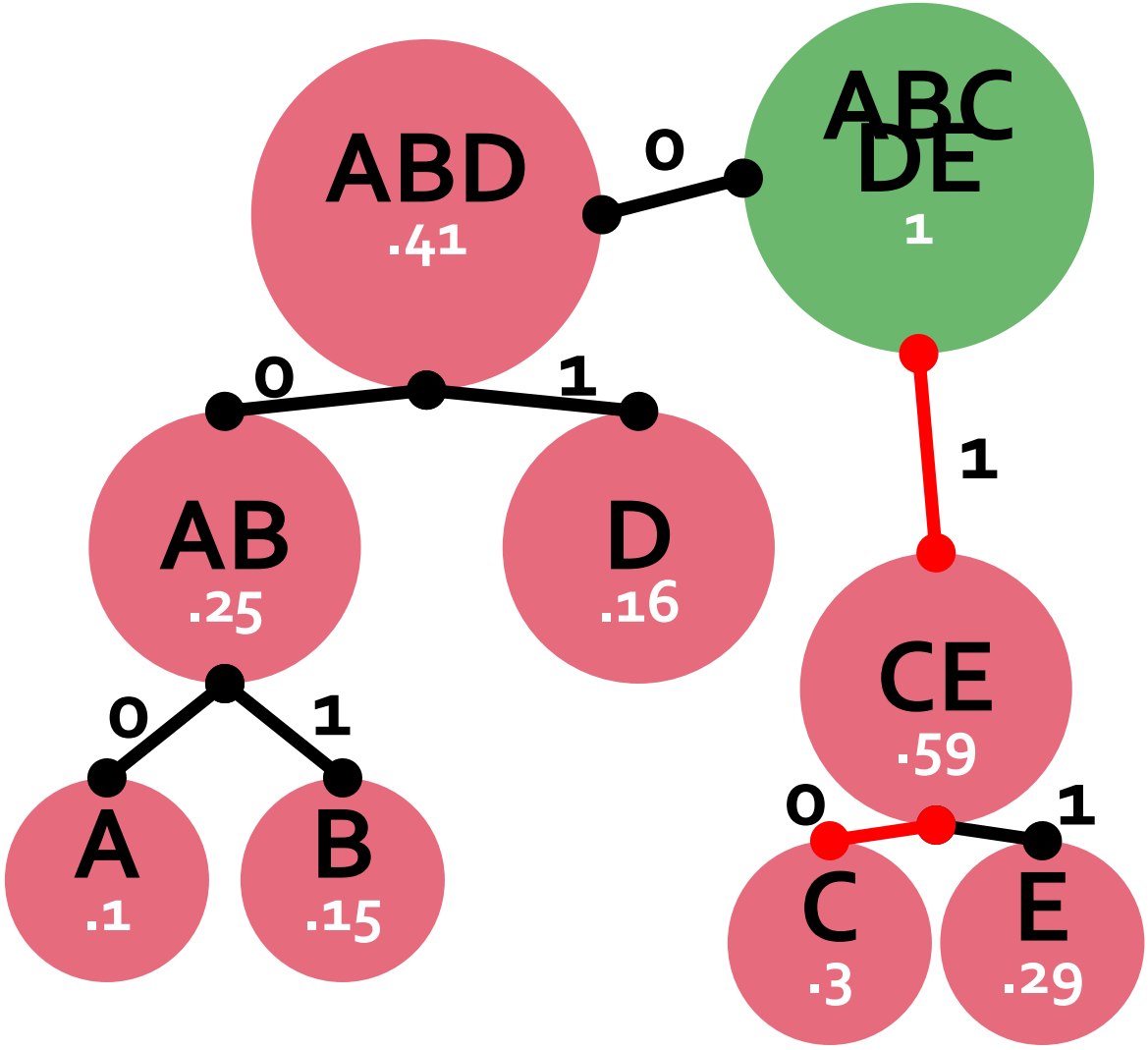


Huffman Coding

A=000

D=01

C=10



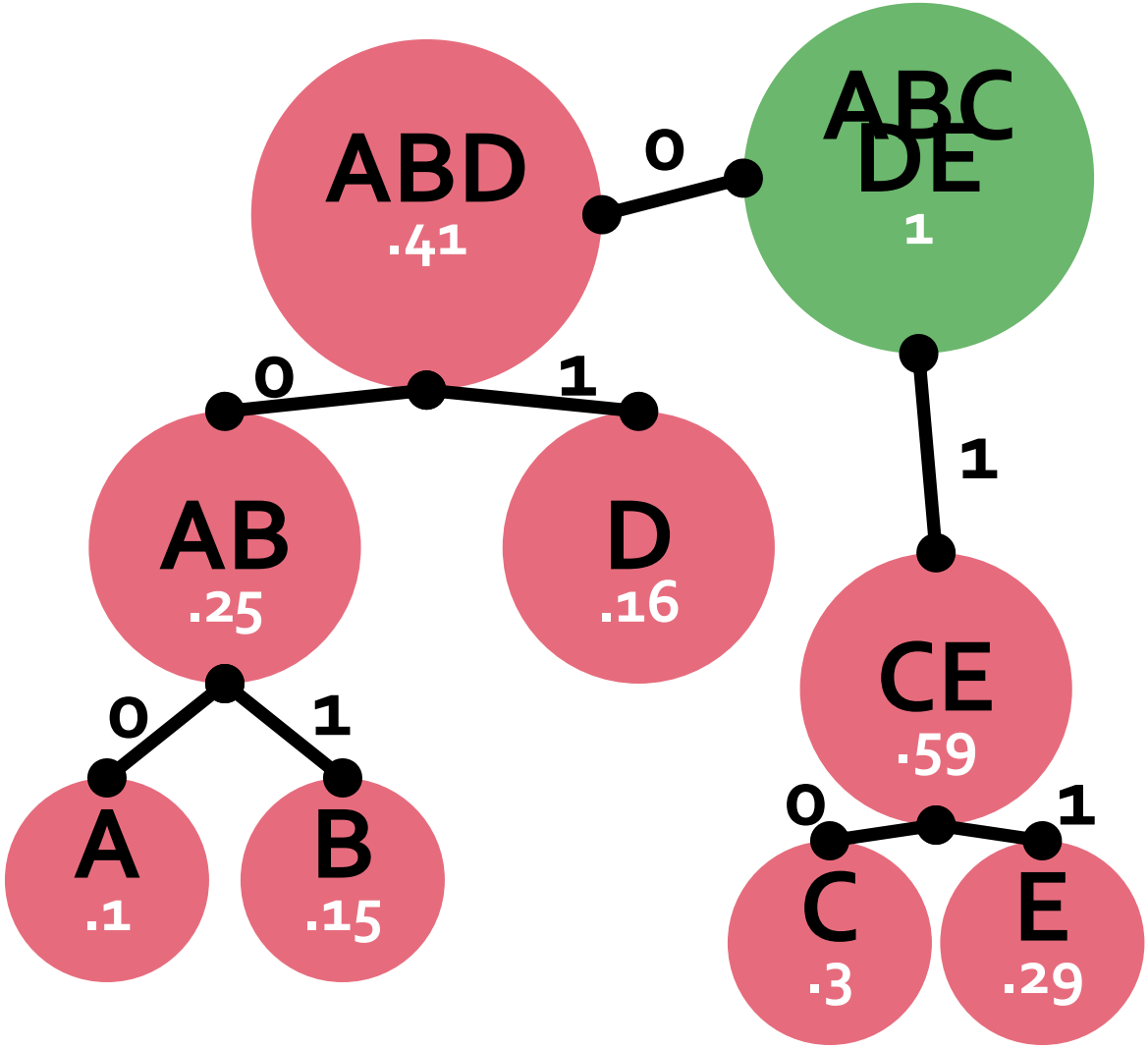
Huffman Coding

Encoding

CAD

=

1000001

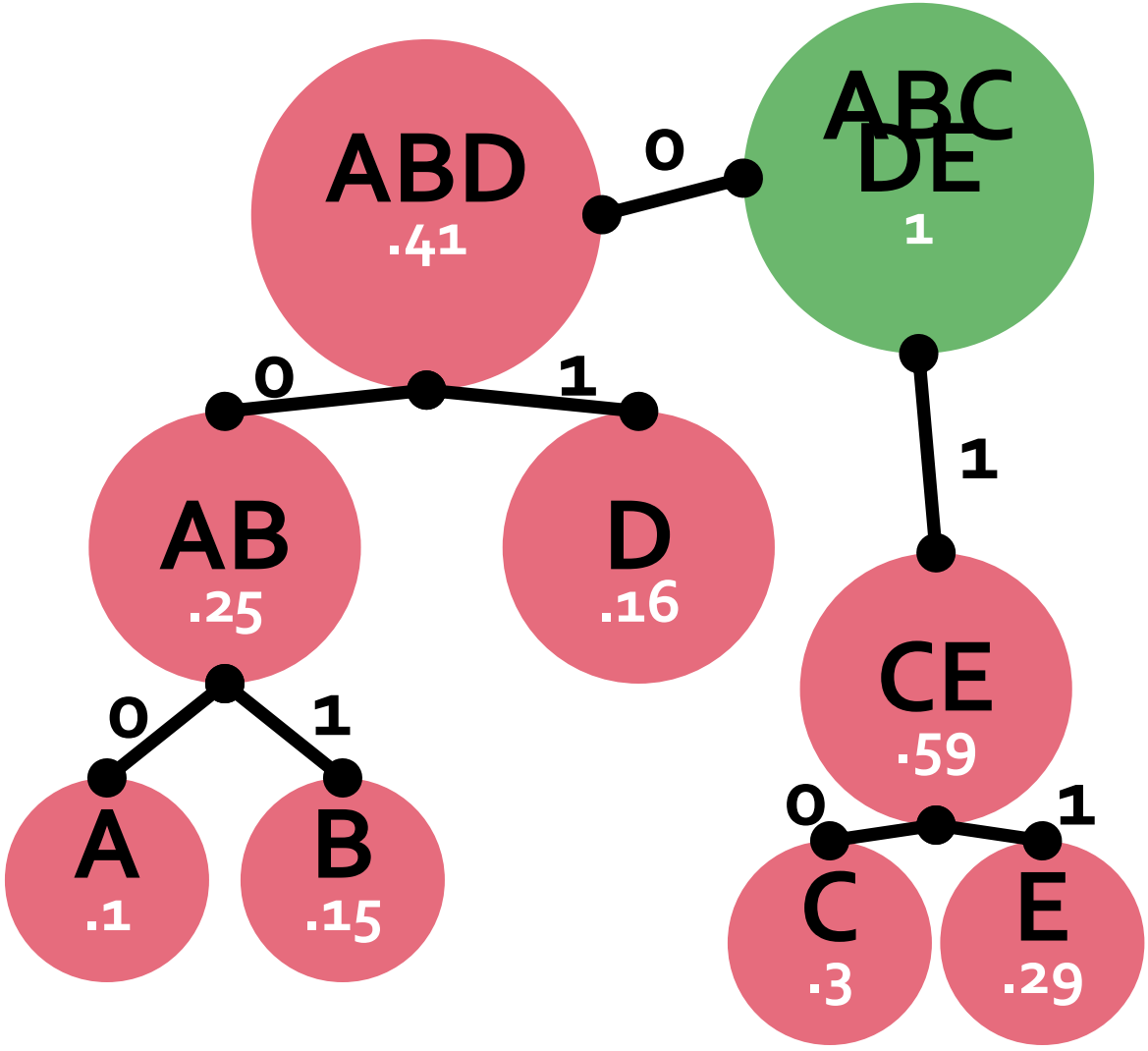


Huffman Coding

Decoding

0011101

=

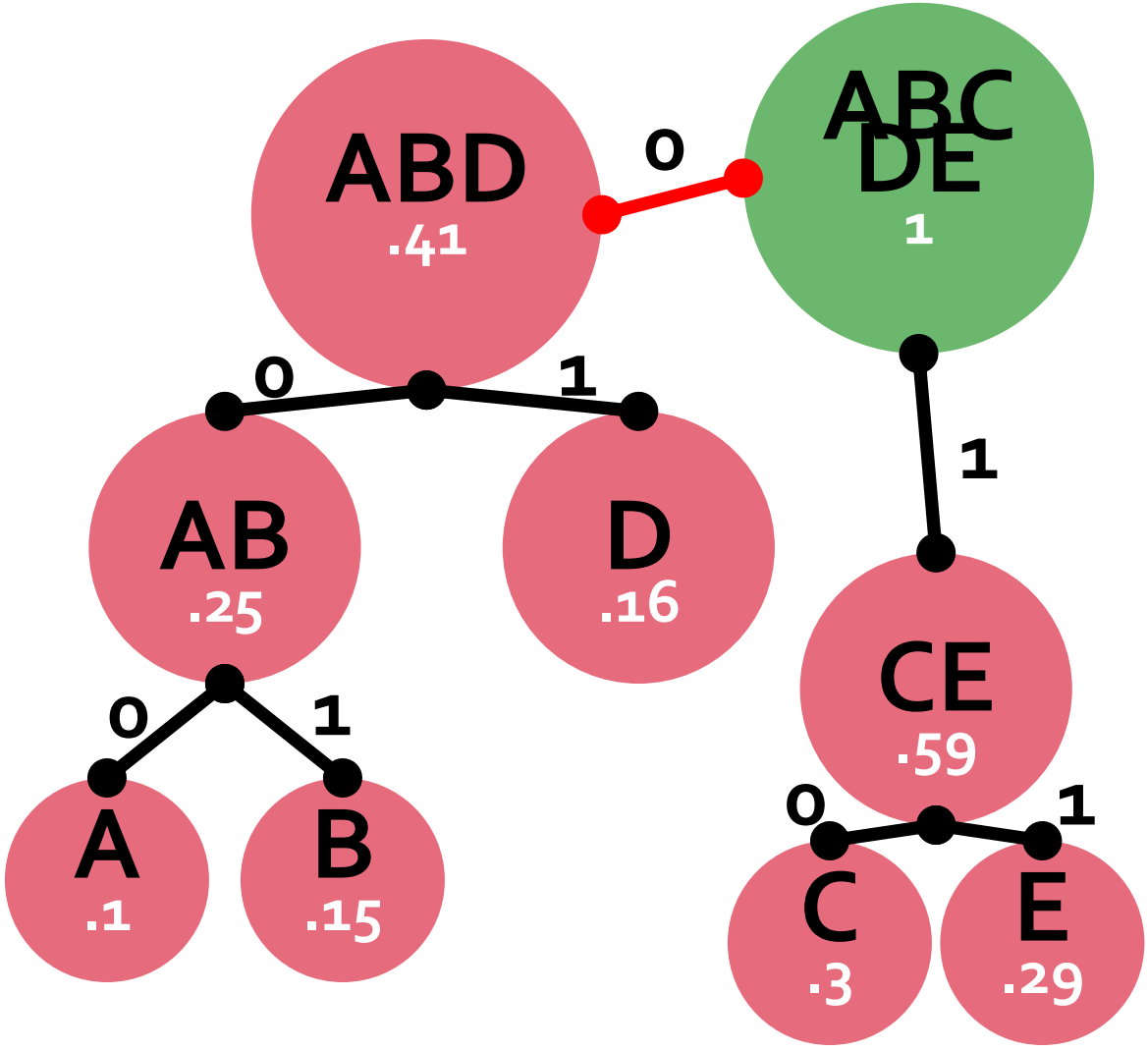


Huffman Coding

Decoding

0011101

=

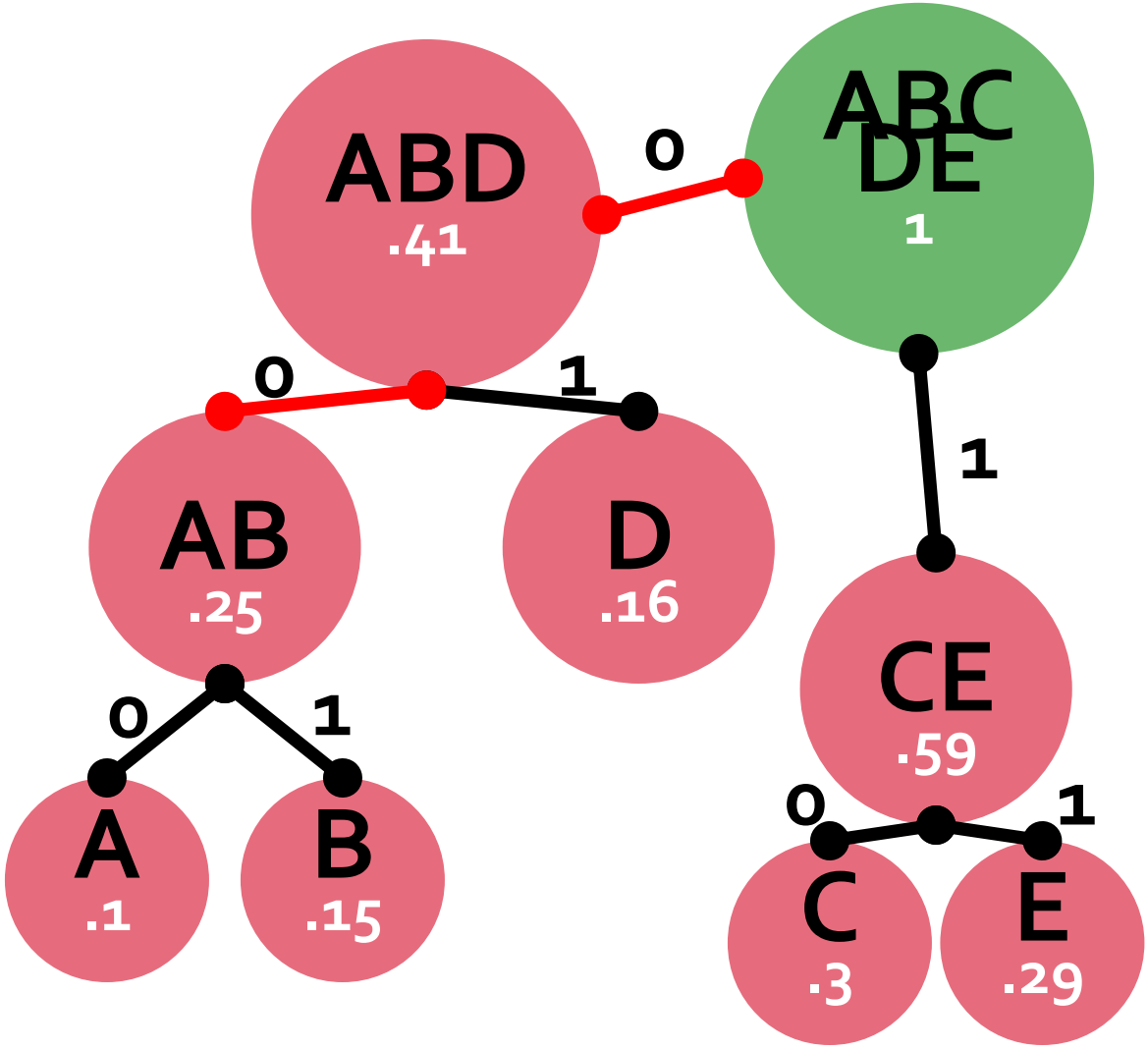


Huffman Coding

Decoding

0011101

=



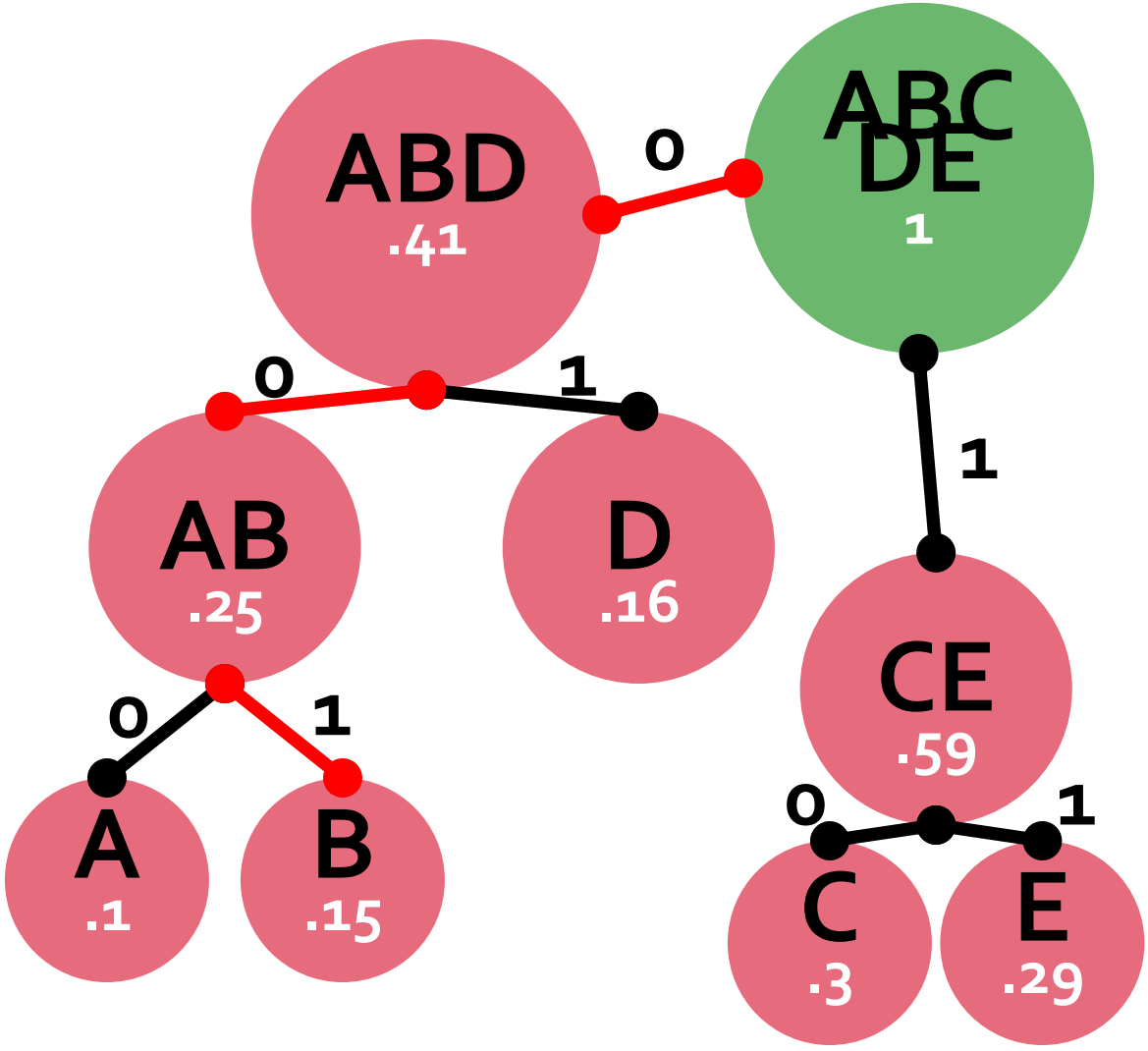
Huffman Coding

Decoding

0011101

=

B



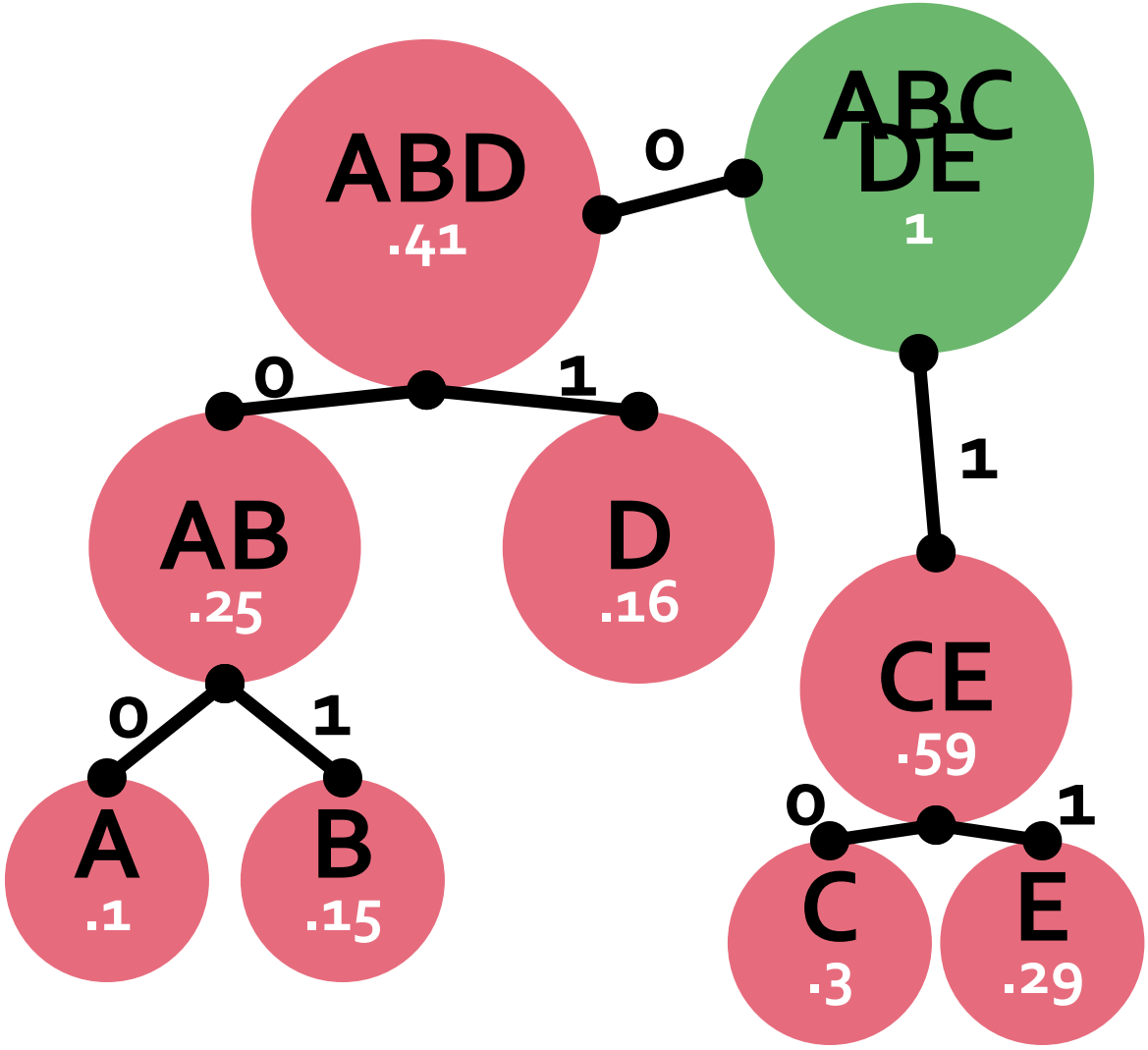
Huffman Coding

Decoding

0011101

=

B



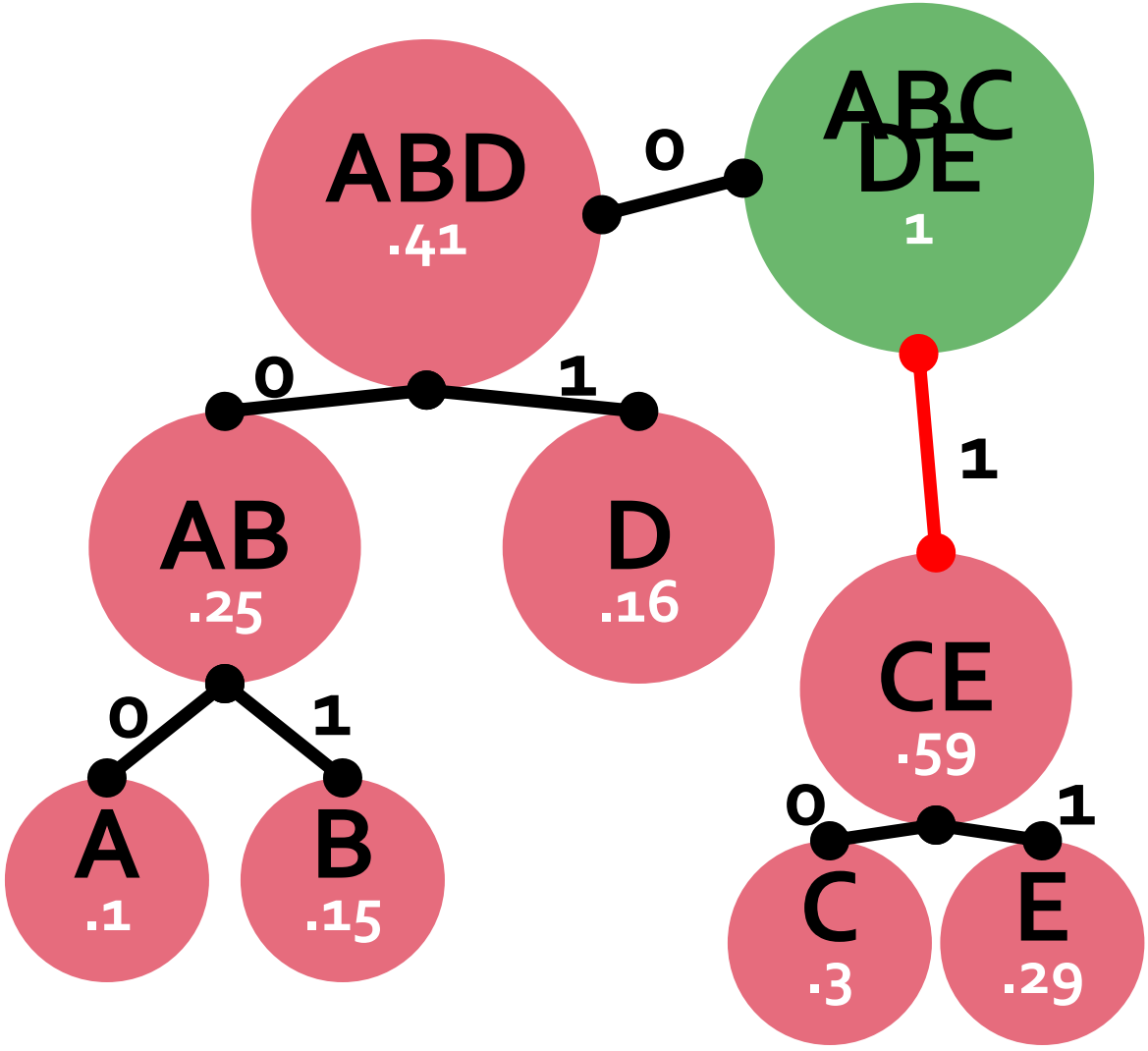
Huffman Coding

Decoding

00111101

=

B



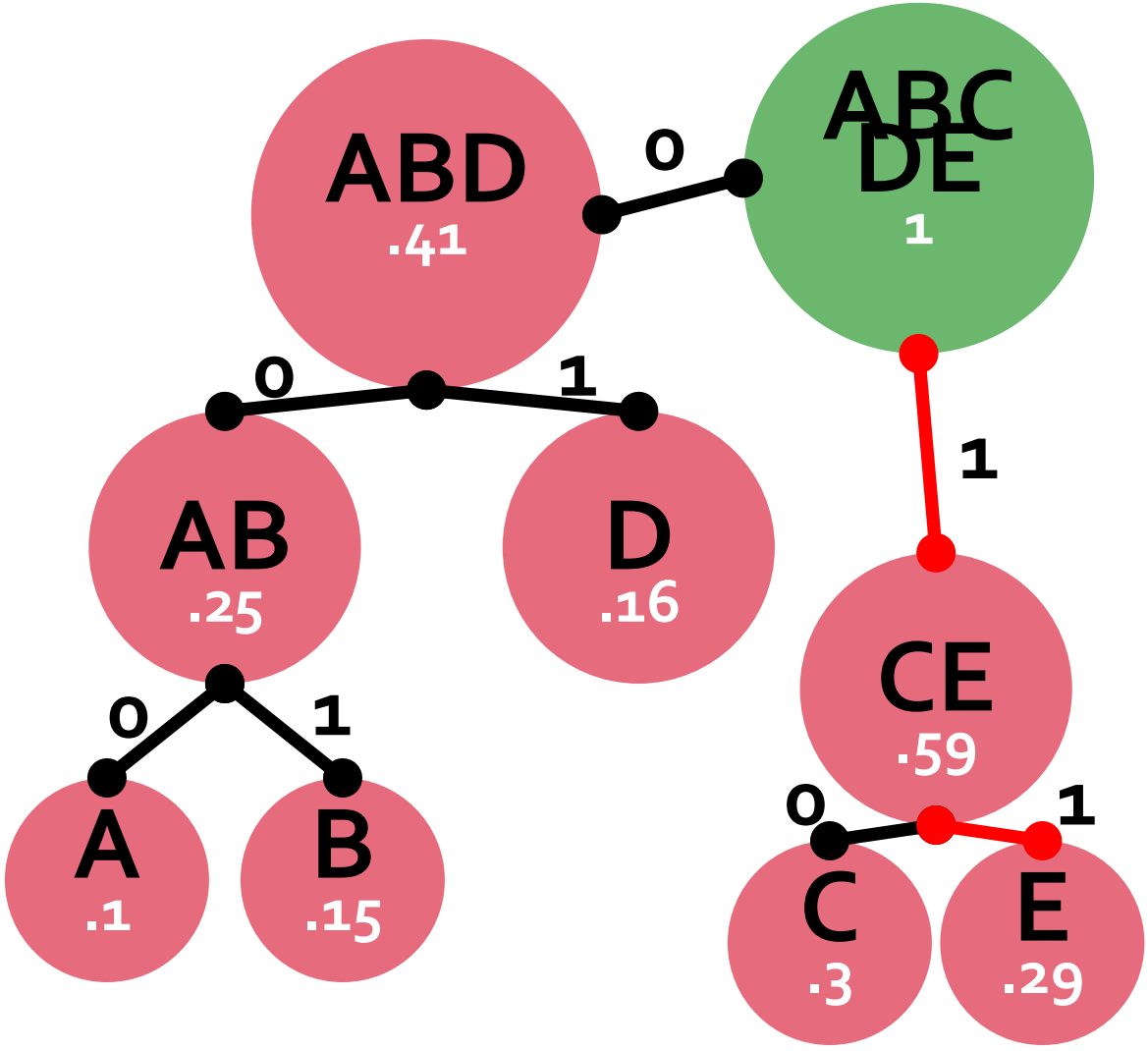
Huffman Coding

Decoding

0011101

=

BE



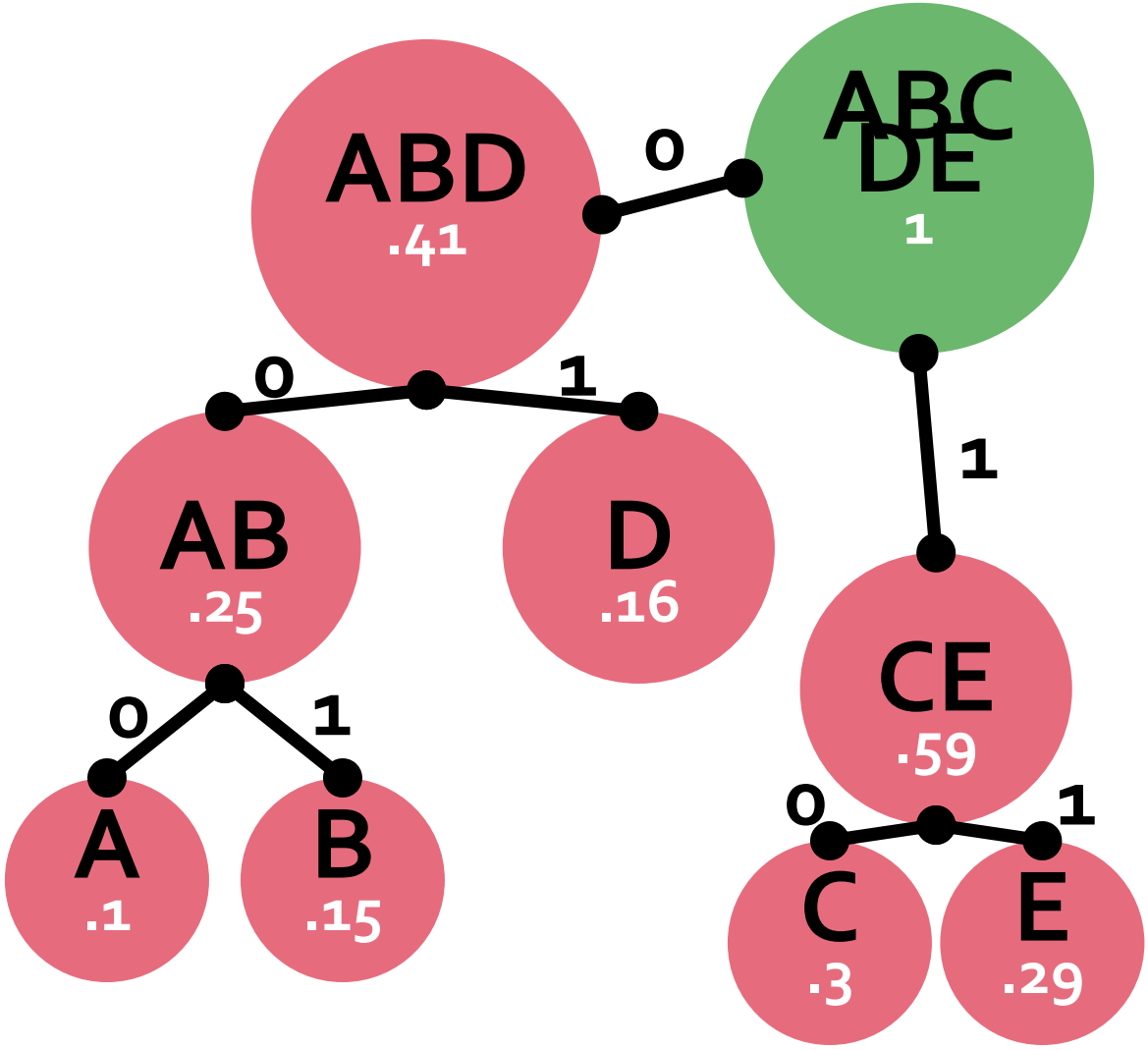
Huffman Coding

Decoding

0011101

=

BE



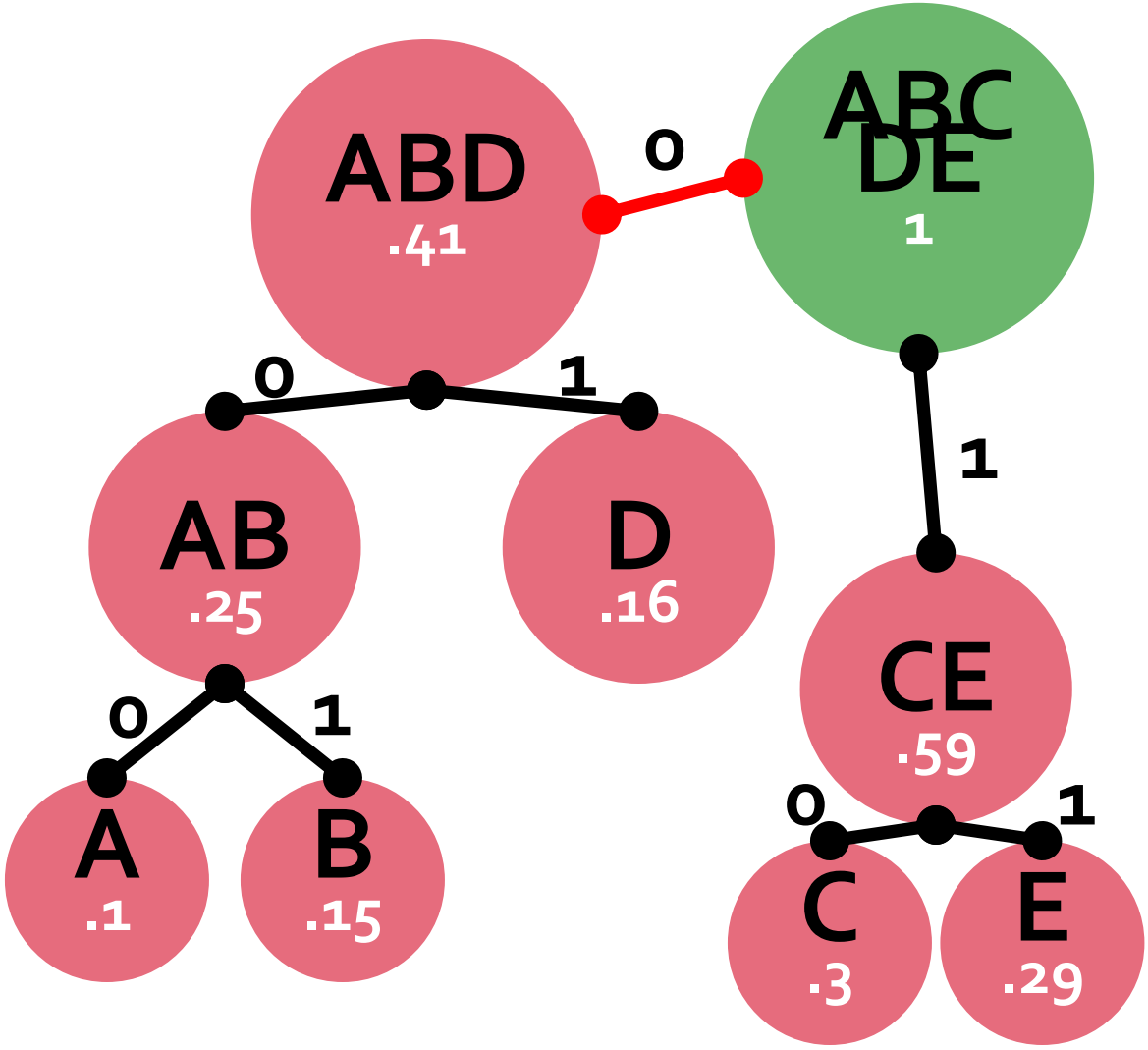
Huffman Coding

Decoding

0011101

=

BE



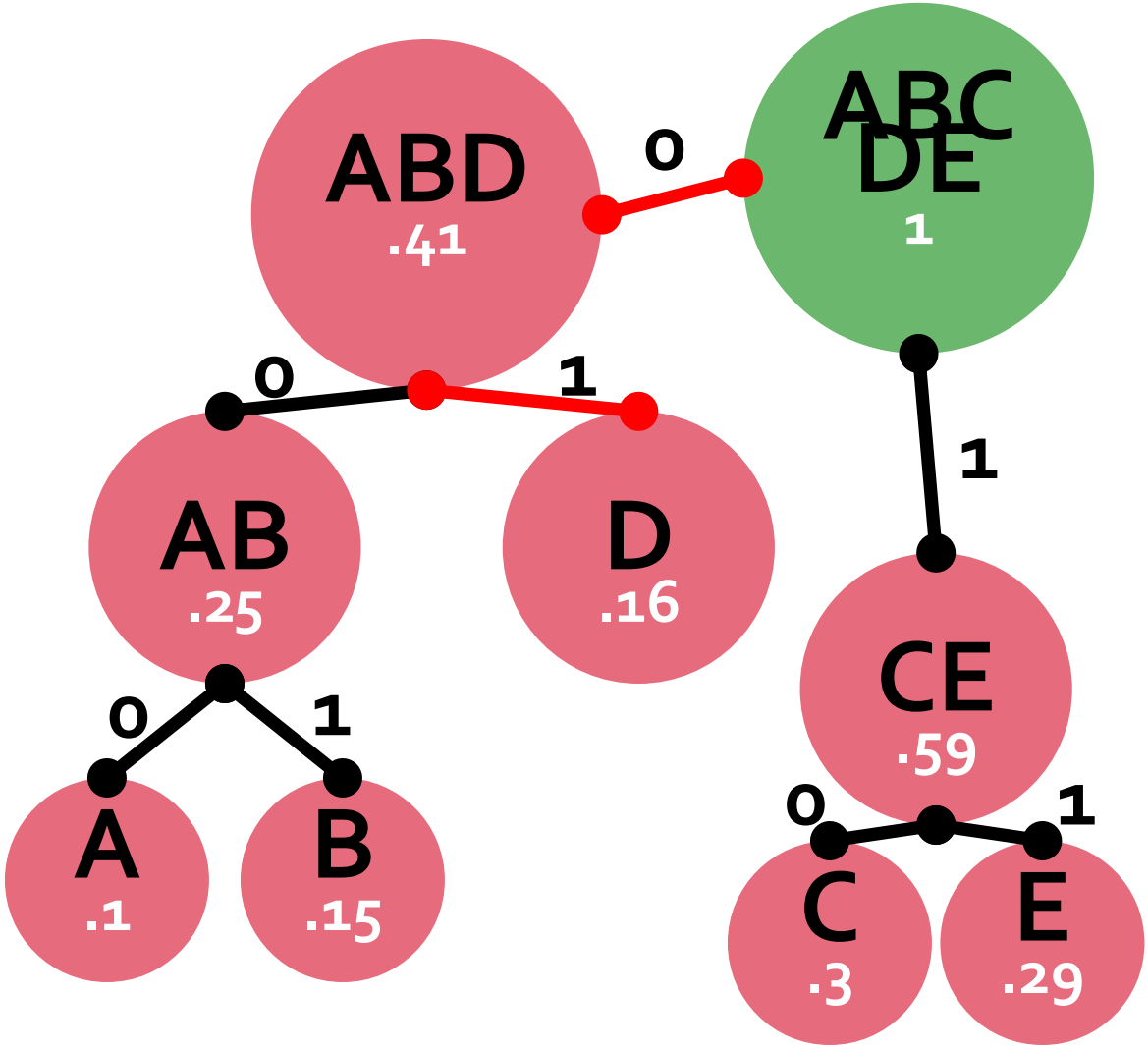
Huffman Coding

Decoding

0011101

=

BED



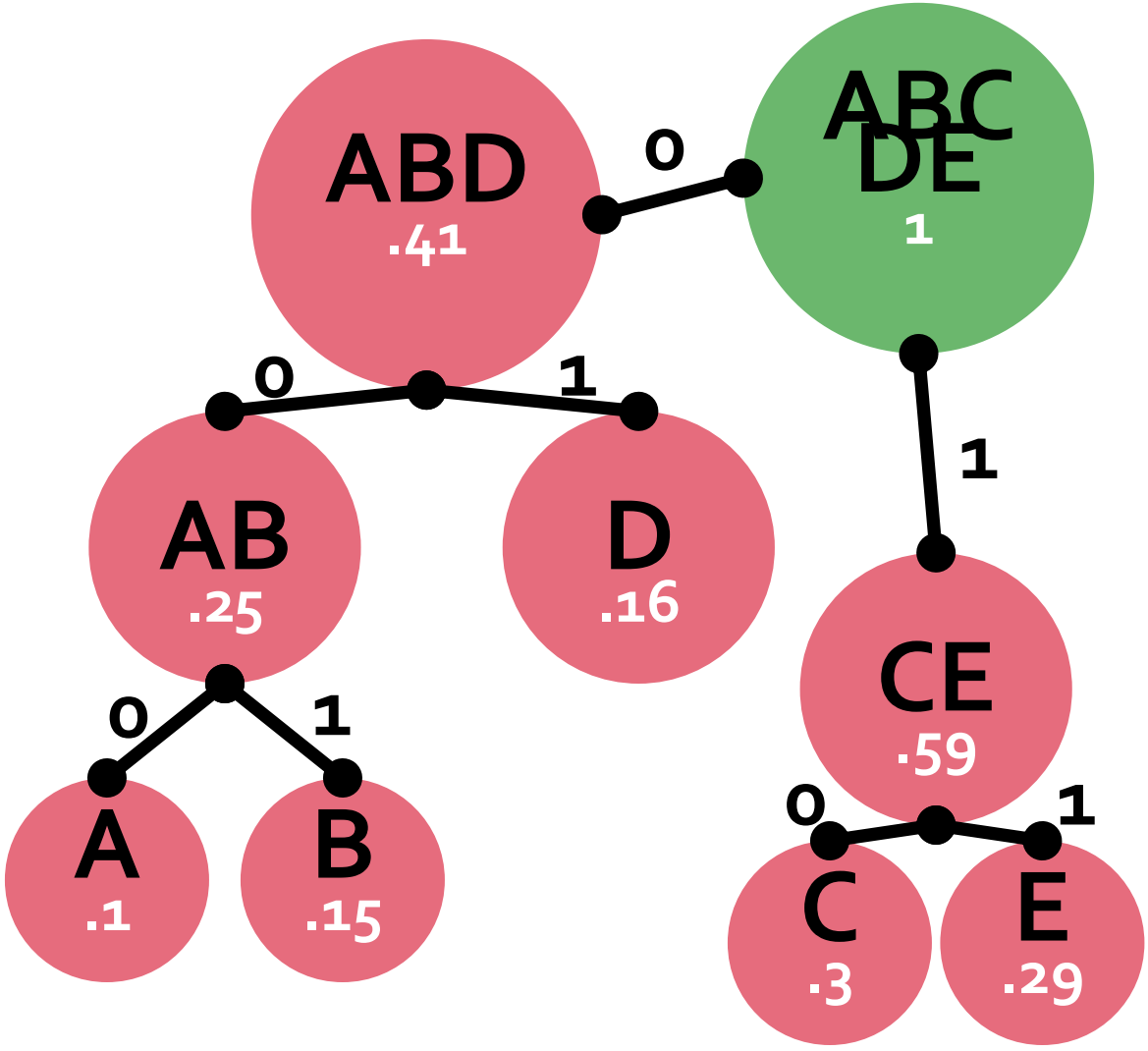
Huffman Coding

Decoding

0011101

=

BED



Huffman Coding

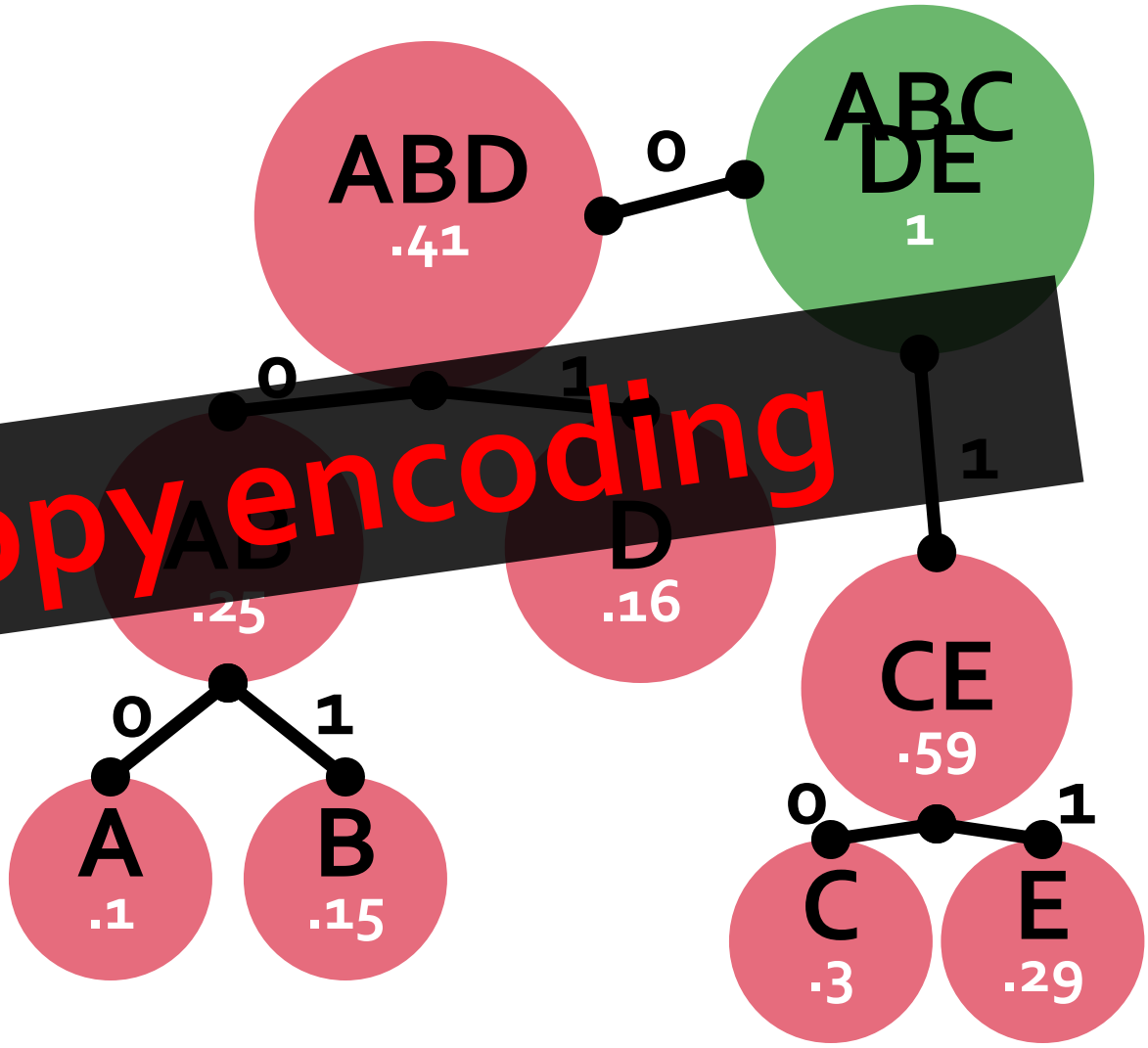
Decoding

0011101

=

BED

Entropy encoding



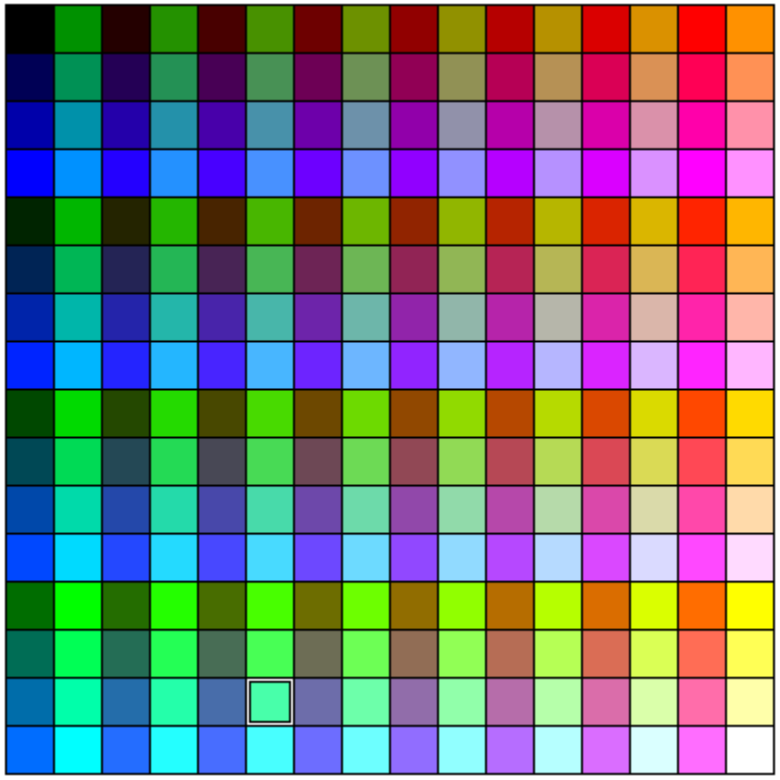
Run-Length Encoding

```
WWWWWWWWWWWWBWWWWWWWWWWWWBBBWWWWW
WWWWWWWWWWWWWWWWWWWWWWWBWWWWWWWWWWWWWW
```

```
12W1B12W3B24W1B13W
```

Encode number of copies

Color Space Reduction



GIF:

Stores 256 colors in a palette; colors are 8 bits per primary (2^{24})

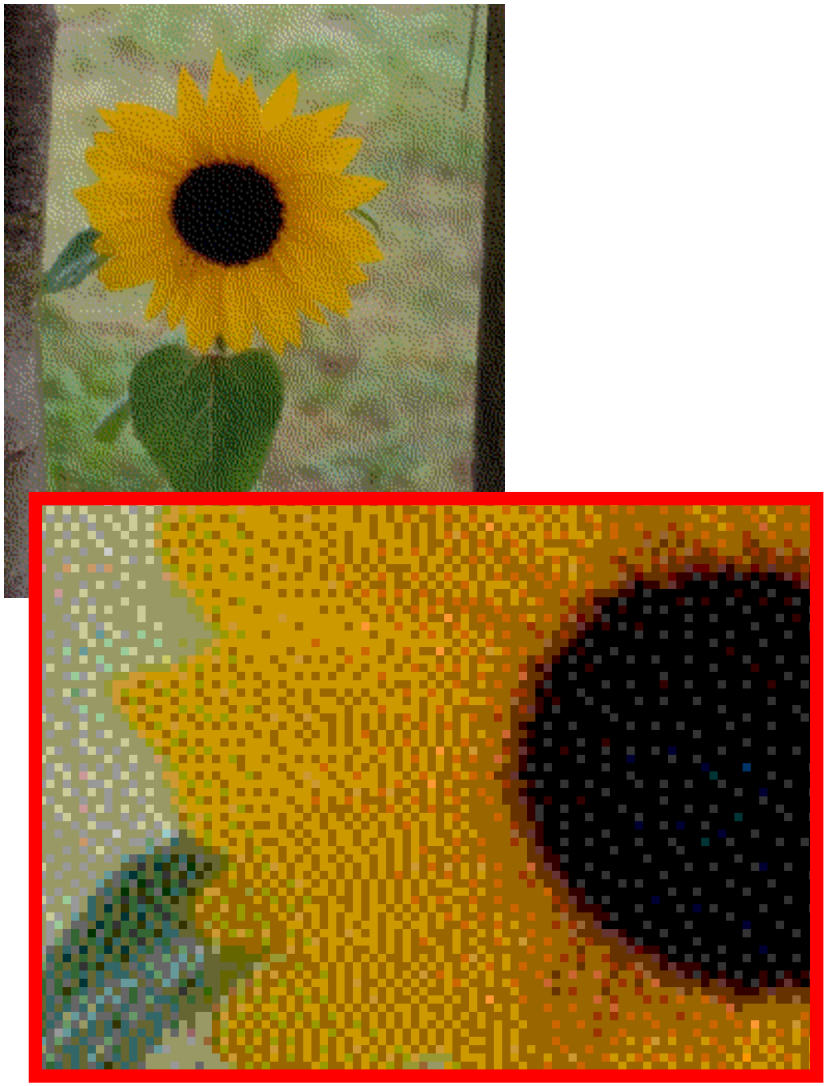
http://developer.sifteo.com/palette.gif

Dithering

Approximate a wider range of colors with a small palette using pixels of two or more colors to approximate in-between colors.

http://en.wikipedia.org/wiki/Graphics_Interchange_Format

Dithering

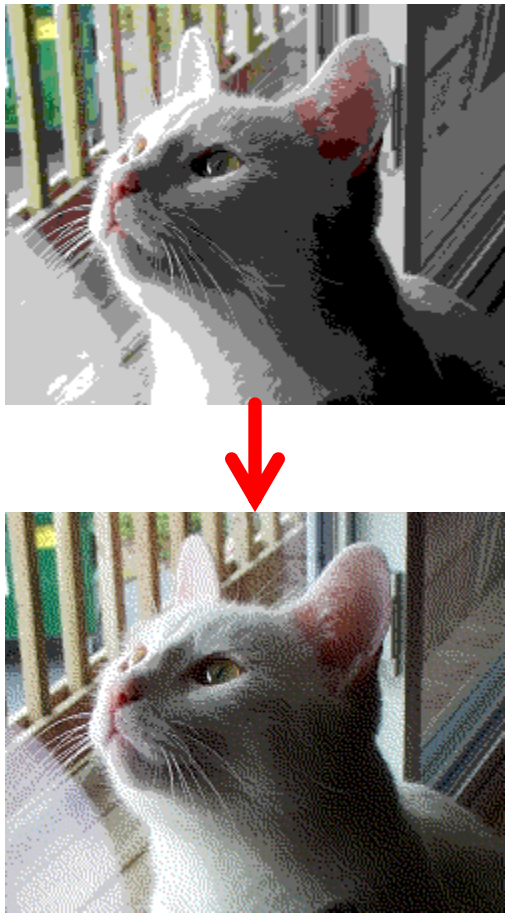


Error diffusion:

Choose a color from the palate for a pixel, compute error, move it to pixels that haven't been visited yet

http://en.wikipedia.org/wiki/Graphics_Interchange_Format

Dithering

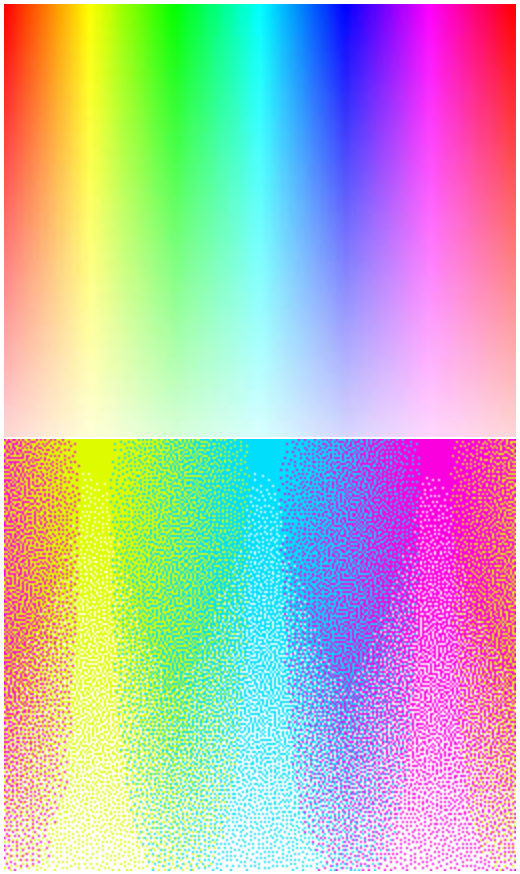


Error diffusion:

Choose a color from the palate for a pixel, compute error, move it to pixels that haven't been visited yet

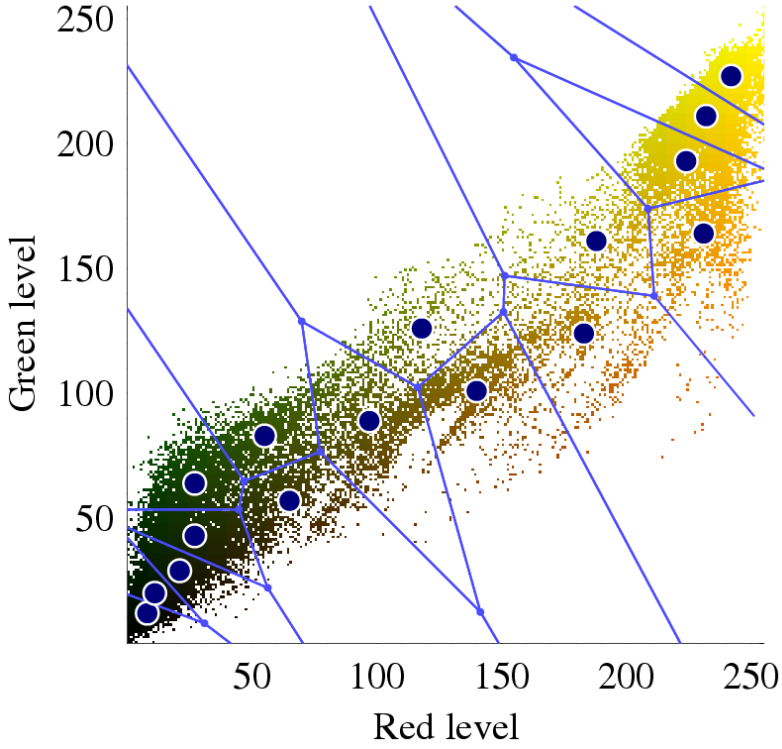
http://en.wikipedia.org/wiki/Dithering

Optimized Palettes



Thousands of colors

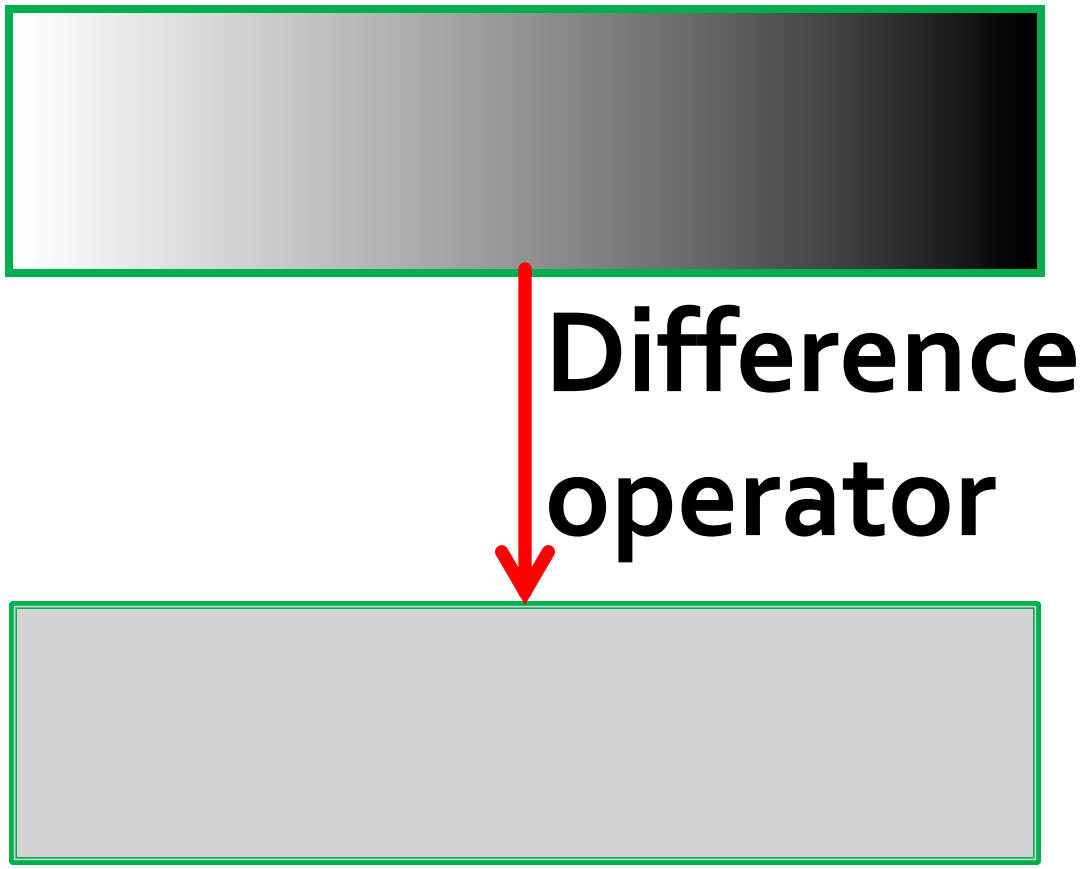
Four colors



http://upload.wikimedia.org/wikipedia/commons/c/c5/Spatial_color_quantization_-_rainbow%2C_4_colors.png
http://upload.wikimedia.org/wikipedia/commons/3/3d/Rosa_Gold_Glow_2_small_noblue_color_space.png

Use color distribution to choose

Transform Coding



Apply transformation to simplify

Transform Coding

Apply transformation to simplify

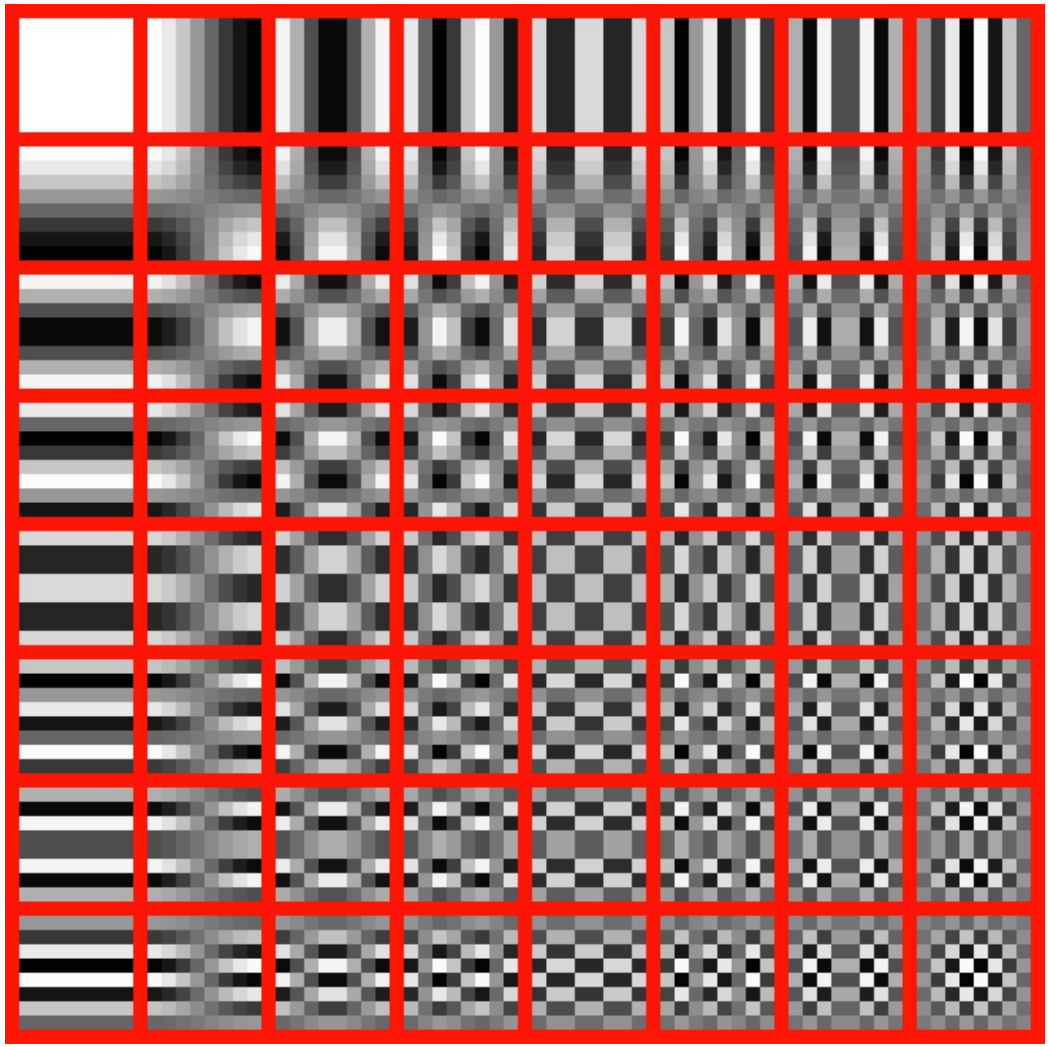
Recall: Fourier Transform

Joseph Fourier
1768-1830

THEOREM(-ISH).

Any function can be written as a combination of simple wiggles.

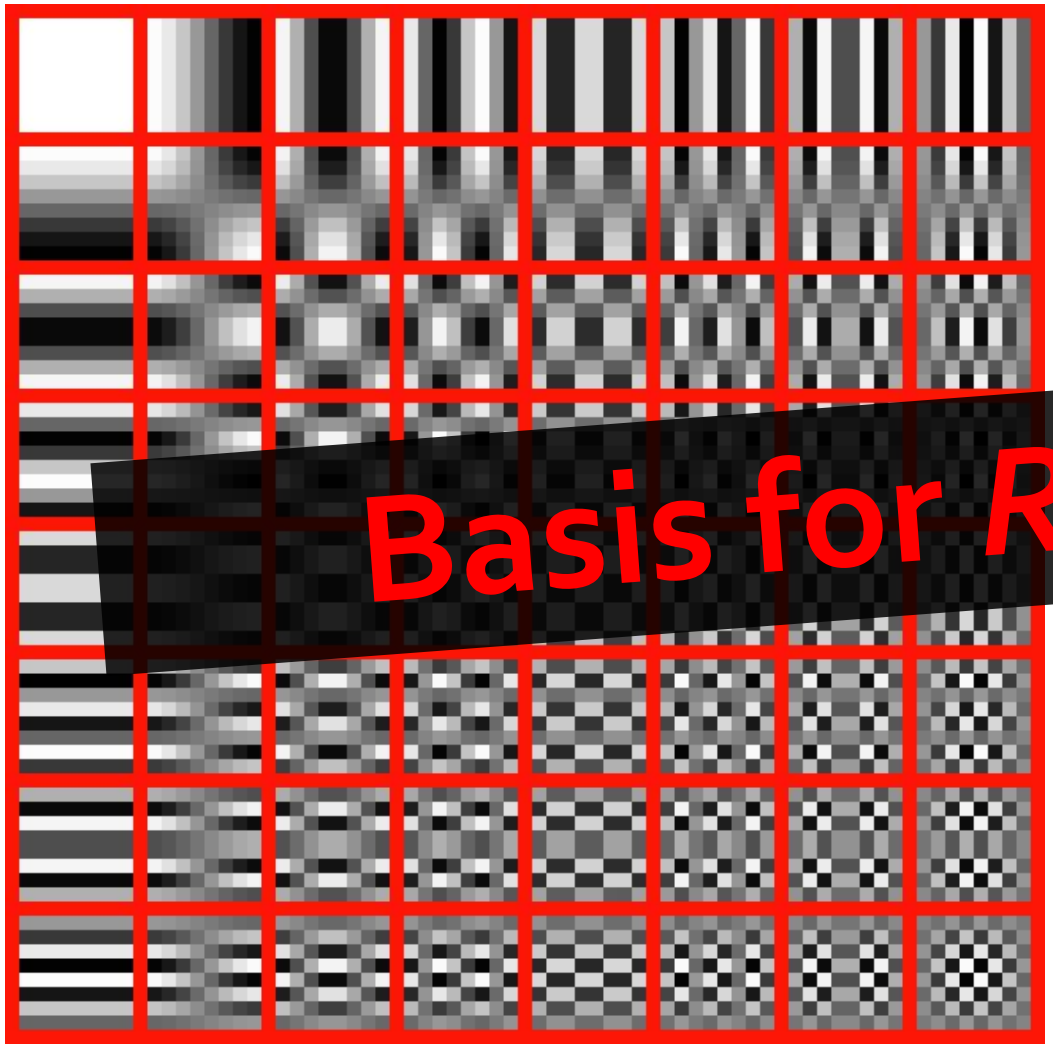
2D Discrete Cosine Basis



Basis for 8x8 blocks; product of cosine/sine basis in *x* and *y*

http://graphics.stanford.edu/courses/cs148-11-fall/lectures/compression.pdf

2D Discrete Cosine Basis



Basis for $R^{8x8} = R^{64}$

Basis for 8x8 block by product of cosine/sine basis in *x* and *y*

http://graphics.stanford.edu/courses/cs148-11-fall/lectures/compression.pdf

JFIF

1. Optionally convert color spaces
2. Reduce chroma data (usually by 2x)
3. Apply discrete cosine transform to 8x8 blocks
4. Quantize DCT values
5. Store result using lossless encoding

Joint File Interchange Format

Steps that Reduce Size

1. Optionally convert color spaces
2. Reduce chroma data (usually by 2x)
3. Apply discrete cosine transform to 8x8 blocks
4. Quantize DCT values
5. Store result using lossless encoding

Quantization

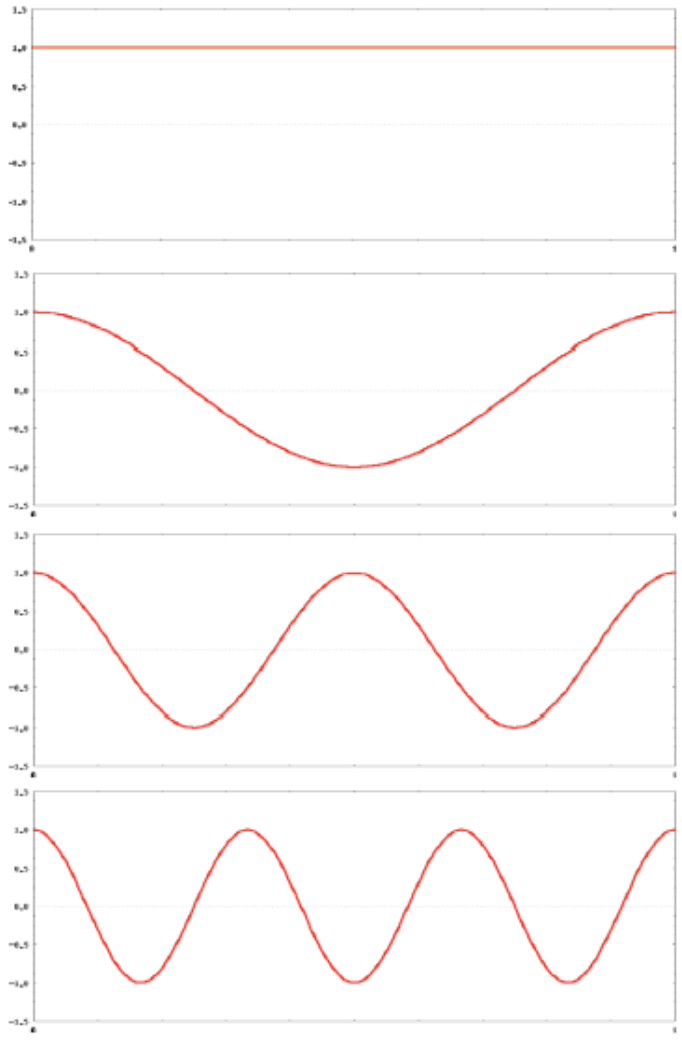
More bits for lower frequencies

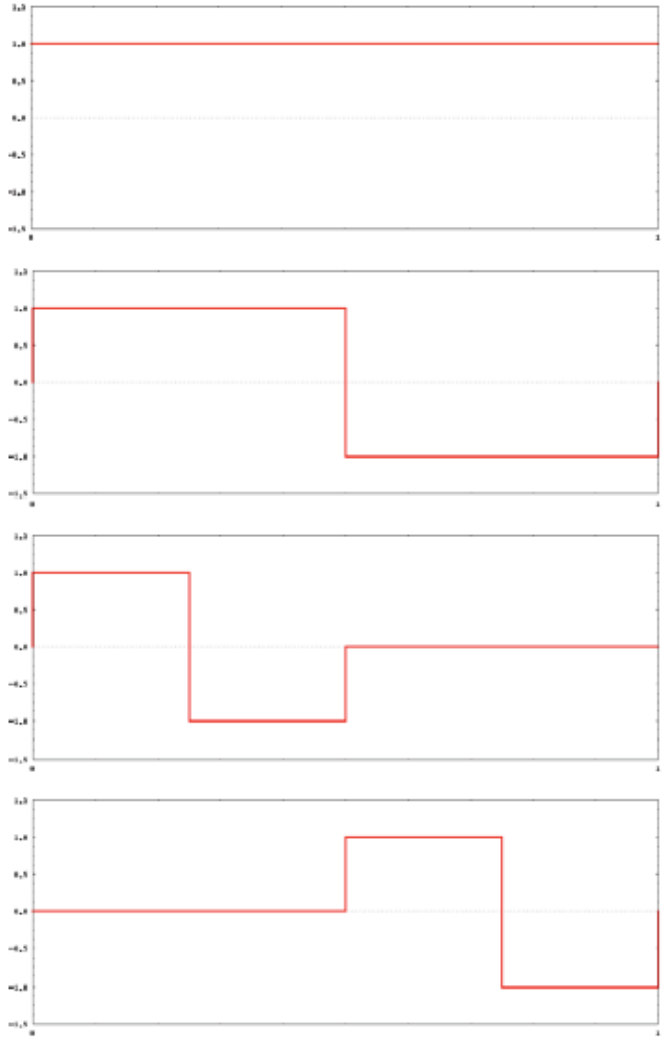
…sometimes throw out high frequencies altogether!

“Blockiness” Artifacts

http://en.wikipedia.org/wiki/JPEG

Cosine vs. Haar Basis



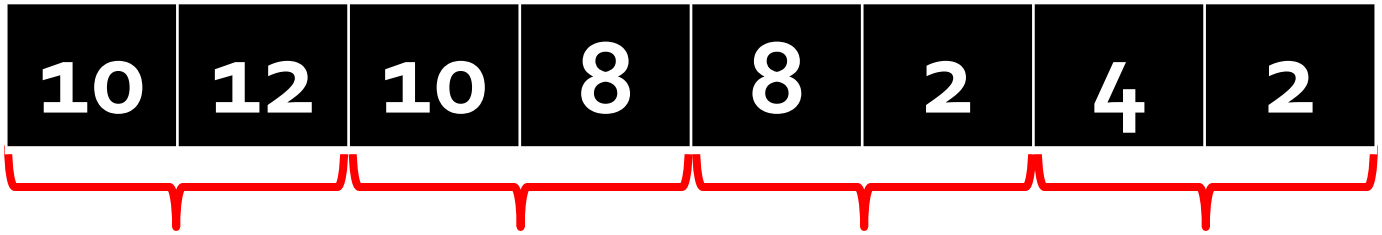


http://graphics.stanford.edu/courses/cs148-11-fall/lectures/compression.pdf

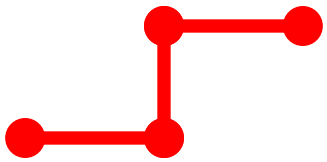
1D Haar Transform

10	12	10	8	8	2	4	2

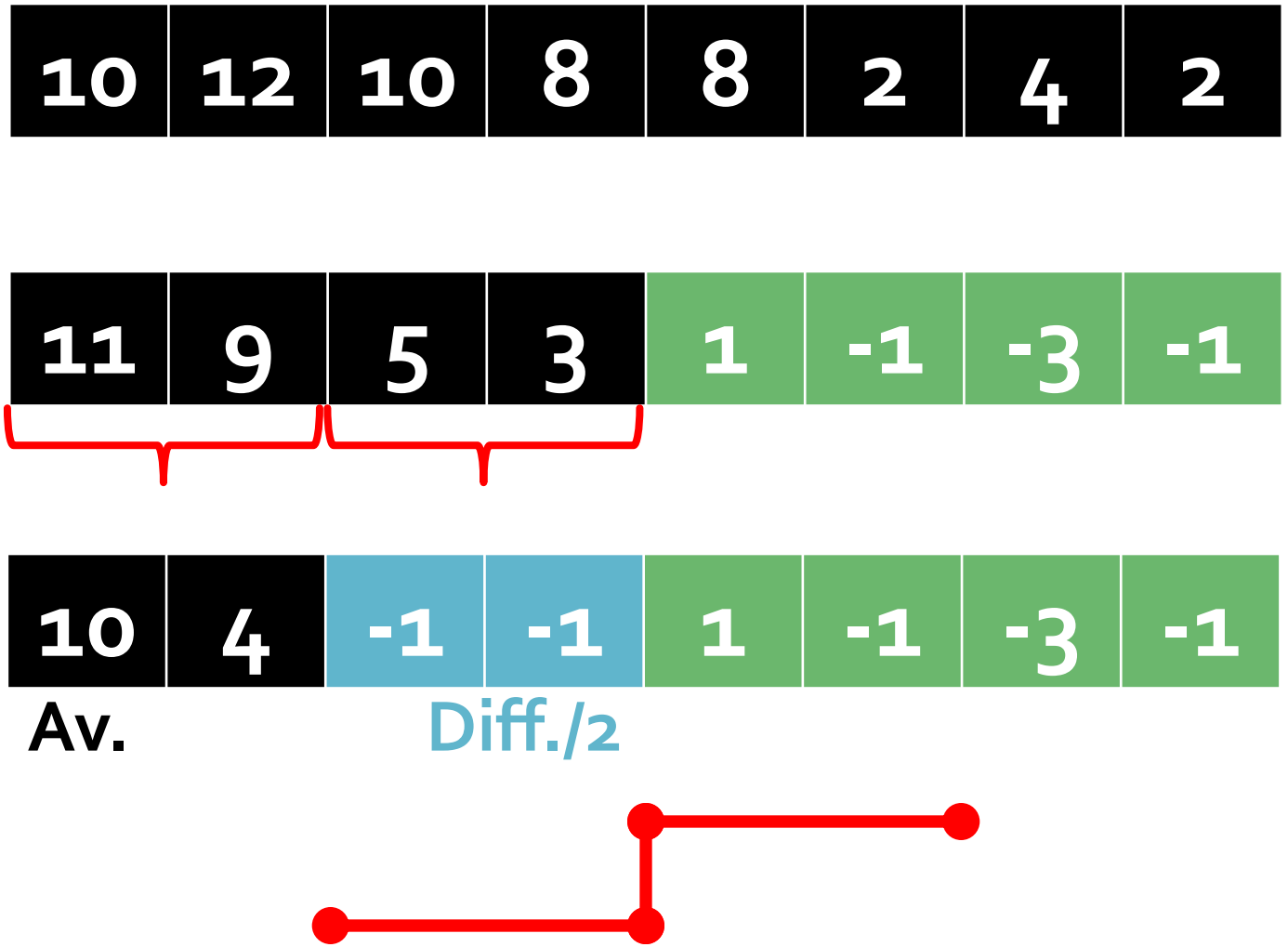
1D Haar Transform



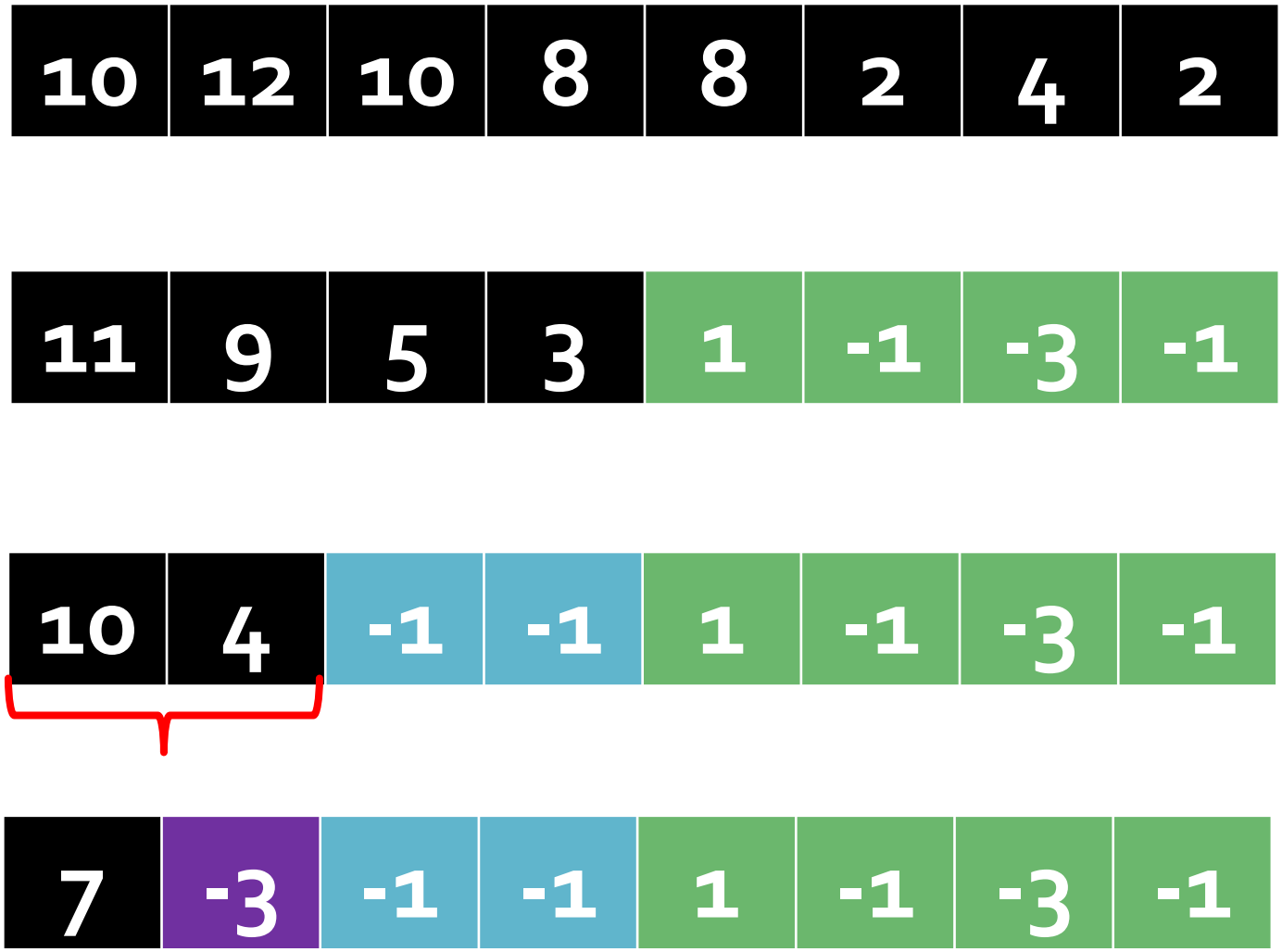
Averages Differences/2



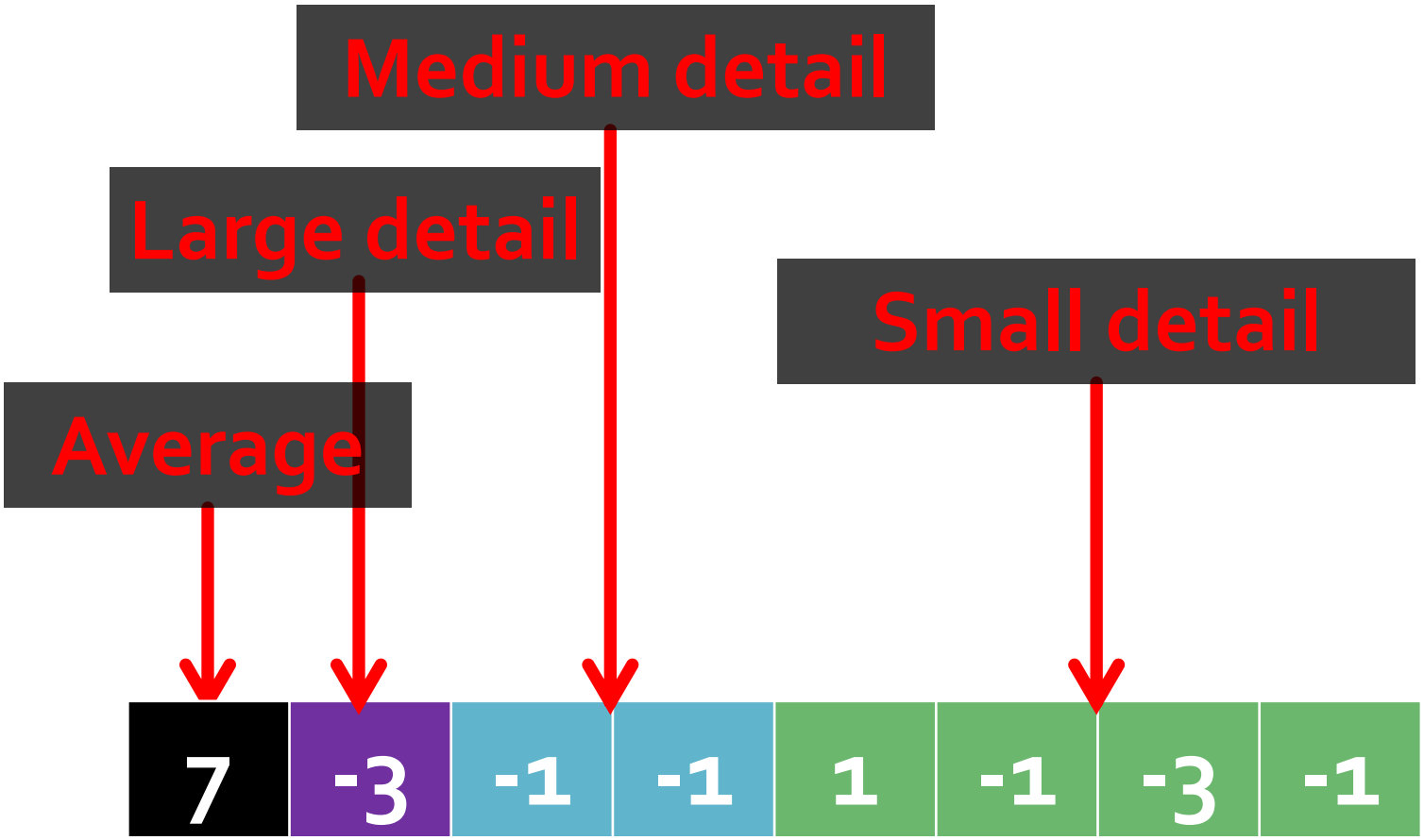
1D Haar Transform



1D Haar Transform



1D Haar Transform



Inverse Haar

$$\frac{a+b}{2} = 7$$
$$\frac{b-a}{2} = -3$$

$$a = 10$$
$$b = 4$$

Inverse Haar

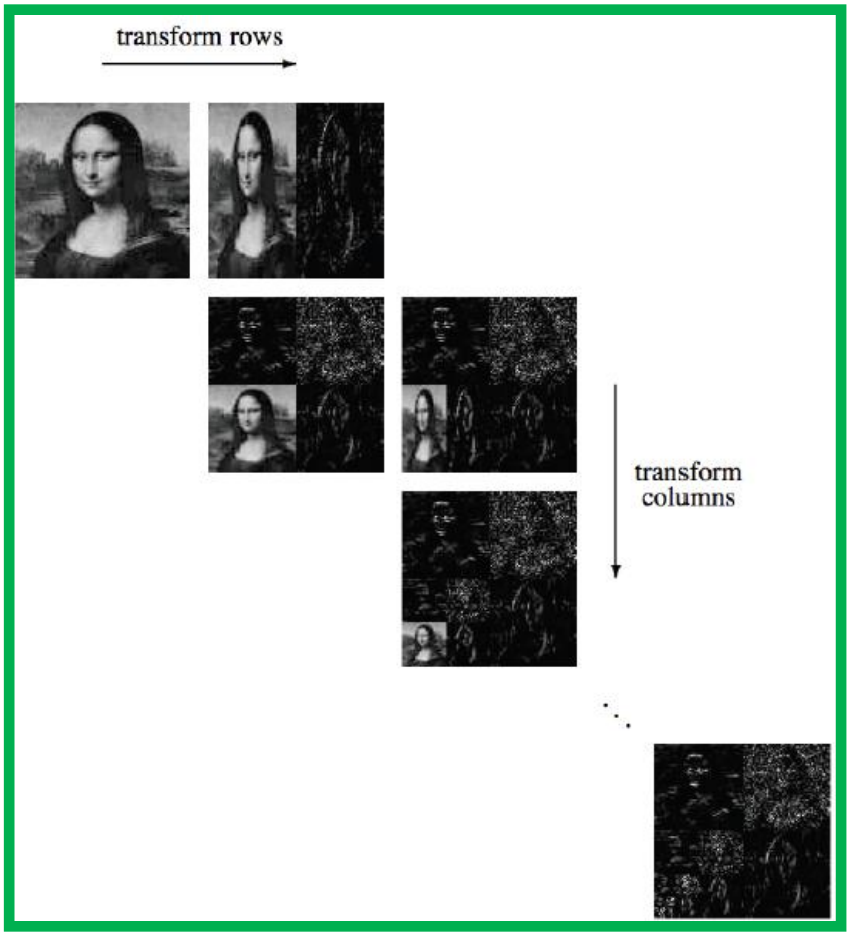
$$\frac{a+b}{2} = 7$$
$$\frac{b-a}{2} = -3$$

$$a = 10$$
$$b = 4$$

10	4	-1	-1	1	-1	-3	-1

7	-3	-1	-1	1	-1	-3	-1

Two Options



http://graphics.stanford.edu/courses/cs148-11-fall/lectures/compression.pdf

JPEG2000

Similar to JPEG, but replace DCT with Haar basis.

Fewer artifacts! No blocks! Easy to compute!
Progressive! Can recover portions of image!

Other Formats

- **PNG**
  Portable Network Graphics; GIF replacement for internet bitmaps
- **TIFF**
  Tagged Image File Format; combines images and data for printing application
- **EXR**
  Open standard for HDR (ILM)
- **PPM, PGM, …**
  Raw data
- **SVG**
  Vector using XML (easy!)

http://tutorial9.s3.amazonaws.com/uploads/2008/05/jpeg-comparisons.jpg

Next time: Video!

Image Compression

CS 148, Summer 2012
Introduction to Computer Graphics and Imaging

Justin Solomon