Exploring Digital Information Technologies: Lecture 1- Part 2

The Landscape—Information and Computation

Encoding Information in Bits

A list of colors:

In[121]:=

```
{Purple, Blue, Green, Yellow, Orange, Red, Brown, Black, White}
```

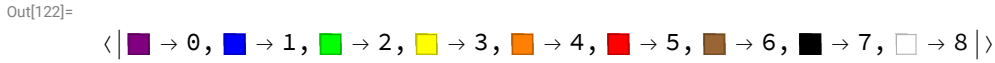
Out[121]=

Use a number to represent each color.
9 colors so numbers 0-8 can be used:

In[122]:=

```
AssociationThread[{Purple, Blue, Green, Yellow, Orange, Red, Brown, Black, White} → Range[0, 8]]
```

Out[122]=



If these 9 different numbers are all the information we had to store using Binary Digits or bits, how many bits would we need?

With 1 bit we can store 2 numbers:

```
0 -> 0
1 -> 1
```

With 2 bits we can store 4 numbers:

```
00 -> 0
01 -> 1
10 -> 2
11 -> 3
```

With 3 bits we can store 8 numbers:

```
000 -> 0
001 -> 1
010 -> 2
011 -> 3
100 -> 4
101 -> 5
```

```
110 -> 6
111 -> 7
```

We need 1 more bit to get our 9th number in:

```
0000 -> 0
0001 -> 1
0010 -> 2
0011 -> 3
0100 -> 4
0101 -> 5
0110 -> 6
0111 -> 7
1000 -> 8
```

What decimal number is represented by the bits 1000?

In[146]:=
```
FromDigits[{1, 0, 0, 0}, 2]
```
Out[146]=
```
8
```

What decimal number is represented by the bits 1111?

In[147]:=
```
FromDigits[{1, 1, 1, 1}, 2]
```
Out[147]=
```
15
```

How would I represent the number 15 in bits?

In[148]:=
```
IntegerDigits[15, 2]
```
Out[148]=
```
{1, 1, 1, 1}
```

How would I represent the number 16 in bits?

In[149]:=
```
IntegerDigits[16, 2]
```
Out[149]=
```
{1, 0, 0, 0, 0}
```

How would I represent the number 26 in bits?

In[150]:=
```
IntegerDigits[26, 2]
```
Out[150]=
```
{1, 1, 0, 1, 0}
```

How would I represent the number 31 in bits?

In[151]:=
```
IntegerDigits[31, 2]
```
Out[151]=
```
{1, 1, 1, 1, 1}
```

How many bits do I need to represent 8 billion?

In[152]:=
```
IntegerDigits[8 000 000 000, 2]
```
Out[152]=
```
{1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0,
 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
```

In[153]:=
```
IntegerDigits[8 000 000 000, 2] // Length
```
Out[153]=
```
33
```

$$\lceil \text{Log}_2 N \rceil$$

In[154]:=
```
Log[2, 8 000 000 000] // Ceiling
```
Out[154]=
```
33
```

How many different numbers can I represent with 33 bits?

In[155]:=
```
2^33
```
Out[155]=
```
8 589 934 592
```

How many different numbers can I represent with 5 bits?

In[156]:=
```
2^5
```
Out[156]=
```
32
```

In[157]:=

Encoding Images

I have a matrix of 1s and 0s:

In[158]:=

```
myImageData = {{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1},
   {1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1}, {1, 1, 1, 0, 0, 1, 1, 1,
    1, 1, 1, 0, 0, 1, 1, 1}, {1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1},
   {1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1}, {1, 0, 1, 1, 1, 0, 0, 1, 1,
    0, 0, 1, 1, 1, 0, 1}, {1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1},
   {1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1}, {1, 0, 1, 1, 1, 1, 1, 1, 1,
    1, 1, 1, 1, 1, 0, 1}, {1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1},
   {1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1}, {1, 1, 0, 1, 1, 1, 0, 0, 0,
    0, 1, 1, 1, 0, 1, 1}, {1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1},
   {1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1}, {1, 1, 1, 1, 1, 0, 0, 0, 0,
    0, 0, 1, 1, 1, 1, 1}, {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}};
```

Let me lay it out nicely:

In[159]:=

```
Grid[smiley]
```

Out[159]=

```
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 0 0 0 0 0 0 1 1 1 1 1
1 1 1 0 0 1 1 1 1 1 1 0 0 1 1 1
1 1 0 1 1 1 1 1 1 1 1 1 1 0 1 1
1 1 0 1 1 1 1 1 1 1 1 1 1 0 1 1
1 0 1 1 1 0 0 1 1 0 0 1 1 1 0 1
1 0 1 1 1 0 0 1 1 0 0 1 1 1 0 1
1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1
1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1
1 0 1 1 0 1 1 1 1 1 1 0 1 1 0 1
1 0 1 1 1 0 1 1 1 1 1 0 1 1 1 0 1
1 1 0 1 1 1 0 0 0 0 1 1 1 0 1 1
1 1 0 1 1 1 1 1 1 1 1 1 1 0 1 1
1 1 1 0 0 1 1 1 1 1 1 1 0 0 1 1 1
1 1 1 1 1 0 0 0 0 0 0 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

Let's convert the bits into an image, 1s denoting white pixel, 0s denoting black:

In[160]:=

```
Image[smiley]
```

Out[160]=

What if I had a more colorful smiley?

In[161]:=

```
ImageData[ ]
```

Out[161]=

```
{{{1., 1., 1.}, {1., 1., 1.}, {1., 1., 1.}, {1., 1., 1.}, {1., 1., 1.}, {1., 1., 1.},
  {1., 1., 1.}, {1., 1., 1.}, {1., 1., 1.}, {1., 1., 1.}, {1., 1., 1.},
  {1., 1., 1.}, {1., 1., 1.}, {1., 1., 1.}, {1., 1., 1.}, {1., 1., 1.}},
```

{{1., 1., 1.}, {1., 1., 1.}, {1., 1., 1.}, {1., 1., 1.}, {1., 1., 1.},
 {0., 0., 0.}, {0., 0., 0.}, {0., 0., 0.}, {0., 0., 0.}, {0., 0., 0.}, {0., 0., 0.},
 {1., 1., 1.}, {1., 1., 1.}, {1., 1., 1.}, {1., 1., 1.}, {1., 1., 1.}},
{{1., 1., 1.}, {1., 1., 1.}, {1., 1., 1.}, {0., 0., 0.}, {0., 0., 0.},
 {1., 1., 0.}, {1., 1., 0.}, {1., 1., 0.}, {1., 1., 0.}, {1., 1., 0.}, {1., 1., 0.},
 {0., 0., 0.}, {0., 0., 0.}, {1., 1., 1.}, {1., 1., 1.}, {1., 1., 1.}},
{{1., 1., 1.}, {1., 1., 1.}, {0., 0., 0.}, {1., 1., 0.}, {1., 1., 0.},
 {1., 1., 0.}, {1., 1., 0.}, {1., 1., 0.}, {1., 1., 0.}, {1., 1., 0.}, {1., 1., 0.},
 {1., 1., 0.}, {1., 1., 0.}, {0., 0., 0.}, {1., 1., 1.}, {1., 1., 1.}},
{{1., 1., 1.}, {1., 1., 1.}, {0., 0., 0.}, {1., 1., 0.}, {1., 1., 0.},
 {1., 1., 0.}, {1., 1., 0.}, {1., 1., 0.}, {1., 1., 0.}, {1., 1., 0.}, {1., 1., 0.},
 {1., 1., 0.}, {1., 1., 0.}, {0., 0., 0.}, {1., 1., 1.}, {1., 1., 1.}},
{{1., 1., 1.}, {0., 0., 0.}, {1., 1., 0.}, {1., 1., 0.}, {1., 1., 0.},
 {0., 0., 0.}, {0., 0., 0.}, {1., 1., 0.}, {1., 1., 0.}, {0., 0., 0.}, {0., 0., 0.},
 {1., 1., 0.}, {1., 1., 0.}, {1., 1., 0.}, {0., 0., 0.}, {1., 1., 1.}},
{{1., 1., 1.}, {0., 0., 0.}, {1., 1., 0.}, {1., 1., 0.}, {1., 1., 0.},
 {0., 0., 0.}, {0., 0., 0.}, {1., 1., 0.}, {1., 1., 0.}, {0., 0., 0.}, {0., 0., 0.},
 {1., 1., 0.}, {1., 1., 0.}, {1., 1., 0.}, {0., 0., 0.}, {1., 1., 1.}},
{{1., 1., 1.}, {0., 0., 0.}, {1., 1., 0.}, {1., 1., 0.}, {1., 1., 0.},
 {1., 1., 0.}, {1., 1., 0.}, {1., 1., 0.}, {1., 1., 0.}, {1., 1., 0.}, {1., 1., 0.},
 {1., 1., 0.}, {1., 1., 0.}, {1., 1., 0.}, {0., 0., 0.}, {1., 1., 1.}},
{{1., 1., 1.}, {0., 0., 0.}, {1., 1., 0.}, {1., 1., 0.}, {1., 1., 0.},
 {1., 1., 0.}, {1., 1., 0.}, {1., 1., 0.}, {1., 1., 0.}, {1., 1., 0.}, {1., 1., 0.},
 {1., 1., 0.}, {1., 1., 0.}, {1., 1., 0.}, {0., 0., 0.}, {1., 1., 1.}},
{{1., 1., 1.}, {0., 0., 0.}, {1., 1., 0.}, {1., 1., 0.}, {0., 0., 0.},
 {1., 1., 0.}, {1., 1., 0.}, {1., 1., 0.}, {1., 1., 0.}, {1., 1., 0.}, {0., 0., 0.},
 {1., 1., 0.}, {1., 1., 0.}, {1., 1., 0.}, {0., 0., 0.}, {1., 1., 1.}},
{{1., 1., 1.}, {0., 0., 0.}, {1., 1., 0.}, {1., 1., 0.}, {1., 1., 0.},
 {0., 0., 0.}, {1., 1., 0.}, {1., 1., 0.}, {1., 1., 0.}, {1., 1., 0.}, {0., 0., 0.},
 {1., 1., 0.}, {1., 1., 0.}, {1., 1., 0.}, {0., 0., 0.}, {1., 1., 1.}},
{{1., 1., 1.}, {1., 1., 1.}, {0., 0., 0.}, {1., 1., 0.}, {1., 1., 0.},
 {1., 1., 0.}, {0., 0., 0.}, {0., 0., 0.}, {0., 0., 0.}, {0., 0., 0.}, {1., 1., 0.},
 {1., 1., 0.}, {1., 1., 0.}, {0., 0., 0.}, {1., 1., 1.}, {1., 1., 1.}},
{{1., 1., 1.}, {1., 1., 1.}, {0., 0., 0.}, {1., 1., 0.}, {1., 1., 0.},
 {1., 1., 0.}, {1., 1., 0.}, {1., 1., 0.}, {1., 1., 0.}, {1., 1., 0.}, {1., 1., 0.},
 {1., 1., 0.}, {1., 1., 0.}, {0., 0., 0.}, {1., 1., 1.}, {1., 1., 1.}},
{{1., 1., 1.}, {1., 1., 1.}, {1., 1., 1.}, {0., 0., 0.}, {0., 0., 0.},
 {1., 1., 0.}, {1., 1., 0.}, {1., 1., 0.}, {1., 1., 0.}, {1., 1., 0.}, {1., 1., 0.},
 {0., 0., 0.}, {0., 0., 0.}, {1., 1., 1.}, {1., 1., 1.}, {1., 1., 1.}},
{{1., 1., 1.}, {1., 1., 1.}, {1., 1., 1.}, {1., 1., 1.}, {1., 1., 1.},
 {0., 0., 0.}, {0., 0., 0.}, {0., 0., 0.}, {0., 0., 0.}, {0., 0., 0.}, {0., 0., 0.},
 {1., 1., 1.}, {1., 1., 1.}, {1., 1., 1.}, {1., 1., 1.}, {1., 1., 1.}},
{{1., 1., 1.}, {1., 1., 1.}, {1., 1., 1.}, {1., 1., 1.}, {1., 1., 1.},
 {1., 1., 1.}, {1., 1., 1.}, {1., 1., 1.}, {1., 1., 1.}, {1., 1., 1.}, {1., 1., 1.},

In[162]:=
```
RGBColor[1, 1, 1]
```
Out[162]=

In[163]:=
```
RGBColor[1, 1, 0]
```
Out[163]=

In[164]:=
```
RGBColor[0, 0, 0]
```
Out[164]=

Modern systems often use 24 bits to specify one color.
How many possible colors?

In[165]:=
```
2^24
```
Out[165]=
```
16 777 216
```

How many bits to represent a color picture containing 1920×1080 pixels
(for your computer desktop background)?

In[166]:=
```
1920 * 1080 * 24
```
Out[166]=
```
49 766 400
```

In[167]:=
```
ImageData[ ]
```


Out[167]=
```
{ ··· 1 ··· }
```
Size in memory: 6.9 MB | + Show more | Show all | Iconize | Store full expression in notebook

In[168]:=

```
Image[%]
```

Out[168]=

In[169]:=

```
ImageData[ , "Bit"]
```

Out[169]=

```
{{{0, 0, 0, 1}, {0, 0, 0, 1}, {0, 0, 0, 1}, {0, 0, 0, 1}, {0, 0, 0, 1}, ··· 578 ···},
  {0, 0, 0, 1}, {0, 0, 0, 1}, {0, 0, 0, 1}, {0, 0, 0, 1}, {0, 0, 0, 1}}, ··· 365 ···}
```

Size in memory: 6.9 MB | + Show more | Show all | Iconize | Store full expression in notebook

In[170]:=
```
Image[%]
```
Out[170]=

What to do with lots of bits?

8 bits = 1 byte

49766400 bits

How many bytes?

In[171]:=
```
49 766 400 / 8
```
Out[171]=
```
6 220 800
```

How many kilobytes (kb)?

In[172]:=
```
6 220 800 / 1000
```
Out[172]=
$$\frac{31\,104}{5}$$

In[173]:=
```
6 220 800 / 1024
```
Out[173]=
```
6075
```

How many megabyte (mb)?

In[174]:=
```
6 220 800 / (1000 × 1000.)
```
Out[174]=
```
6.2208
```

In[175]:=
```
6 220 800 / (1024 × 1024.)
```
Out[175]=
```
5.93262
```

bytes	unit
10^3	kilobyte
10^6	megabyte
10^9	gigabyte
10^(12)	terabyte
10^(15)	petabyte
10^(18)	exabyte

Reference: https://mathworld.wolfram.com/Byte.html

Compression

Lots and lots of bits! Are all of them needed?

So one image requires: 6,220,800 Bytes = 6.2 MB.

What about a one-hour recorded lecture with 24 images every second?

In[176]:=
```
3600 × 24 × 6.2
```
Out[176]=
```
535 680.
```

That's why we don't usually send videos through e-mail: it's a lot of information!

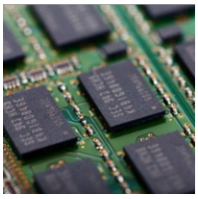
What do humans do?

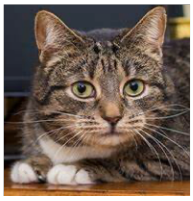
Human languages use a trick to reduce the effective size of ideas.
In particular, frequently used ideas use short words.

Uncommon ideas require long words.

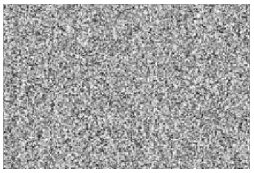
What do modern technologies do?

Modern technologies use compression a lot:
common ideas require fewer bits (like shorter words); usually a few GB for a one-hour video





uncommon videos require more bits (like longer words)



In[182]:=

```
ByteCount[ ]
```

Out[182]=

```
861 528
```

In[183]:=

```
ByteCount[ ]
```

Out[183]=

```
176 960
```

Let's Compress

In[185]:=
```
ByteCount[Compress[ ]]
```


Out[185]=
```
572 720
```

In[186]:=
```
Compress[N[Pi, 100]]
```

Out[186]=
```
1:eJwNzLkRw1AMA1G14gY0IMEDaEMVeNyAc1fvH2309vX5Ps/vui7eUdHOaba8ZlI1OUWKuW6kVOGNMW1:
    uB1oJb7nqQGIVU5jUJGSfCCxlszJiMWtllKDp8FnNO4D7DxHmG0k=
```

In[187]:=
```
Uncompress[%]
```

Out[187]=
```
3.14159265358979323846264338327950288419716939937510582097494459230781640628620899:
 86280348253421170 68
```

In[189]:=
```
ByteCount[ExampleData[{"Text", "USConstitution"}]]
```

Out[189]=
```
49 152
```

In[190]:=
```
ByteCount[Compress[ExampleData[{"Text", "USConstitution"}]]]
```

Out[190]=
```
19 456
```

What is Computation?

The second key concept to be discussed in this class: computation

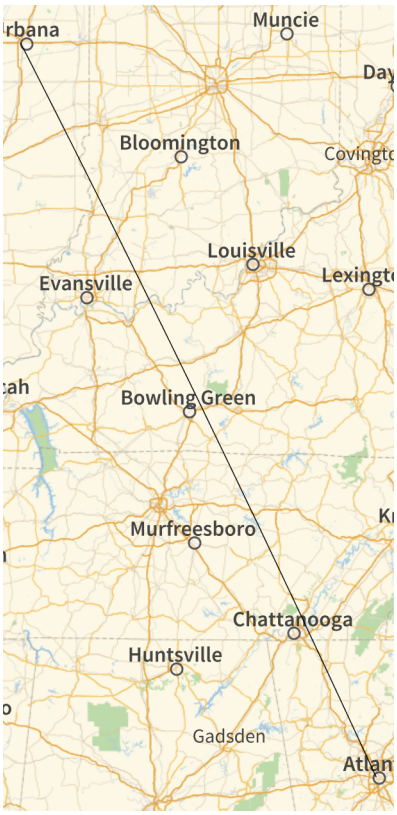
Given some information, answer a question, or make a decision.
Humans are good at such things. So are computers.
Let's consider some examples.

Travel Directions

In[191]:=

```
GeoGraphics[GeoPath[{Champaign CITY, Atlanta CITY}]]
```

Out[191]=



In[192]:=

```
GeoDistance[{Champaign CITY, Atlanta CITY}]
```

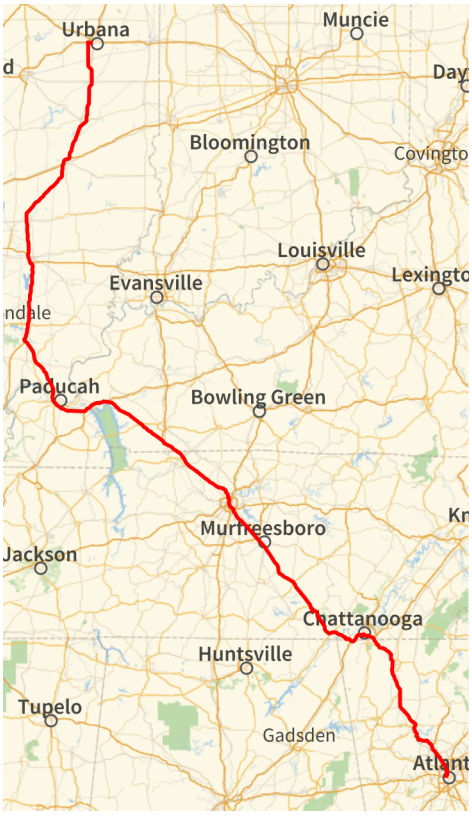
Out[192]=

475.953 mi

In[193]:=

```
GeoGraphics[
 Style[Line[TravelDirections[{Champaign CITY, Atlanta CITY}]], Thick, Red]]
```

Out[193]=



In[194]:=

```
TravelDirections[{Champaign CITY, Atlanta CITY}, "TravelDistance"]
```

Out[194]=

```
618.279 mi
```

In[195]:=

```
TravelDirections[{Champaign CITY, Atlanta CITY}, "TravelTime"]
```

Out[195]=

```
10 h 14 min
```

Identifying Images

In[196]:=

```
ImageIdentify[{ , , , , }]
```

Out[196]=

{jungle cat, Pomeranian, European wildcat, caracal, shire horse}

Social Network Analysis

In[197]:=

```
ExampleData[{"NetworkGraph", "DolphinSocialNetwork"}, "LongDescription"]
```

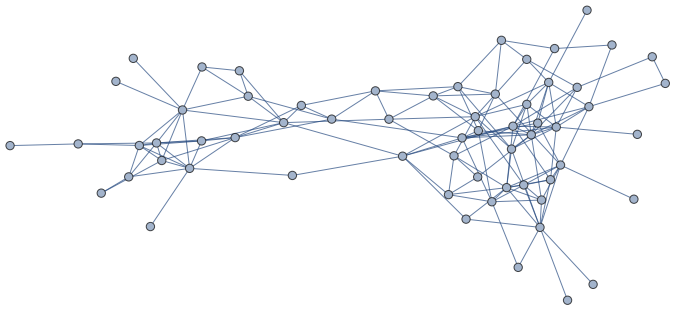
Out[197]=

```
An undirected social network of frequent associations between
  62 dolphins in a community living off Doubtful Sound, New Zealand.
```

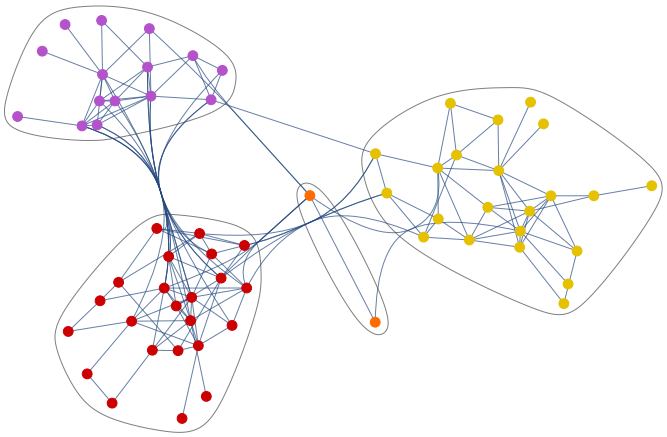
In[198]:=

```
ExampleData[{"NetworkGraph", "DolphinSocialNetwork"}]
```

Out[198]=

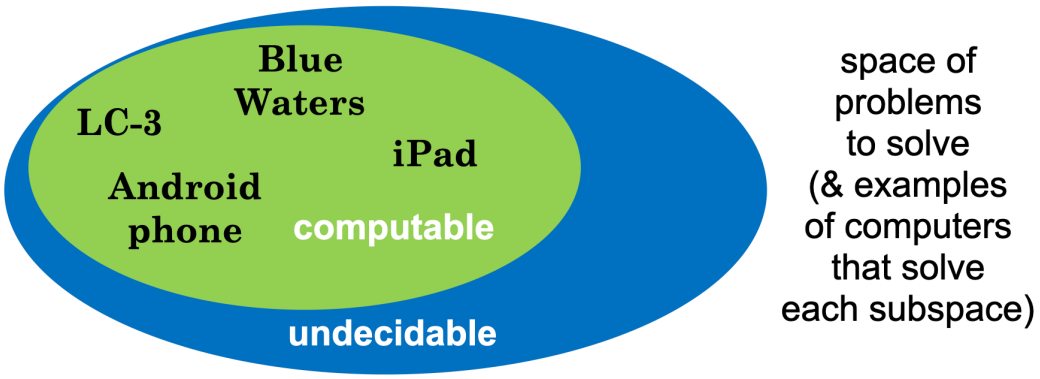


In[199]:=
```
CommunityGraphPlot[ExampleData[{"NetworkGraph", "DolphinSocialNetwork"}]]
```
Out[199]=



Church-Turing Hypothesis

Described by Alan Turing in 1936



Church-Turing Hypothesis: Computers and humans can compute the same things.

Terminology You Should Know from These Slides

- information
- computation
- technology
- bit (BInary digiT) and Byte (8 bits)
- kilo, Mega, Giga, Tera, (Peta, Exa)
- compression

Concepts You Should Know from These Slides

- information is stored as bits (0s and 1s)
- using physical quantities
- how to find the number of bits needed
- to represent one thing from a group
- compression is used to reduce the number of bits needed for more common things in a group
- humans and computers are (in theory) equally capable of computation, given sufficient memory and time (the Church-Turing hypothesis)