Quick Exercise

Write a Sound method that finds the largest change in value between any two sound samples.

For example:

42	-21	14

42	6	-57

Sound Encodings

Consider a typical pop song,
"Sweet Dreams (Are Made of This)",
by the Eurythmics.

```
> sweetDreams.getLength()
9436032
```

Each sample requires 2 bytes,
so encoding the song requires
18,872,064 bytes.

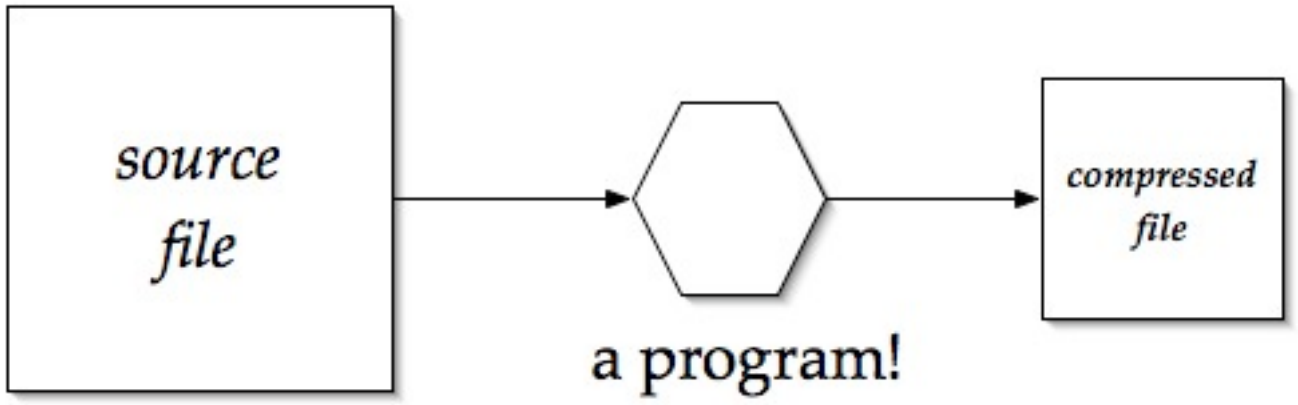
Sound Encodings — Better?

The song, stored as an `.aiff` file,
actually takes up about twice that amount
on my hard drive:
37,746,414 bytes.

Sounds encoded as a `.wav` file
requires a similar amount of space:
37,744,172 bytes.

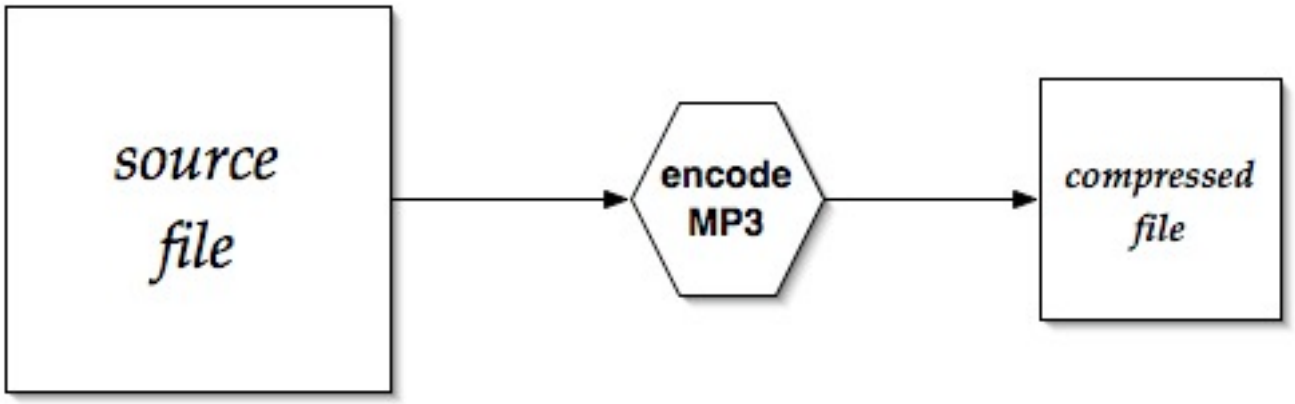
How can I fit more songs on my iPod??

Data Compression



Often we can find a way to store
the same amount of information
in less space — **with a different encoding**.

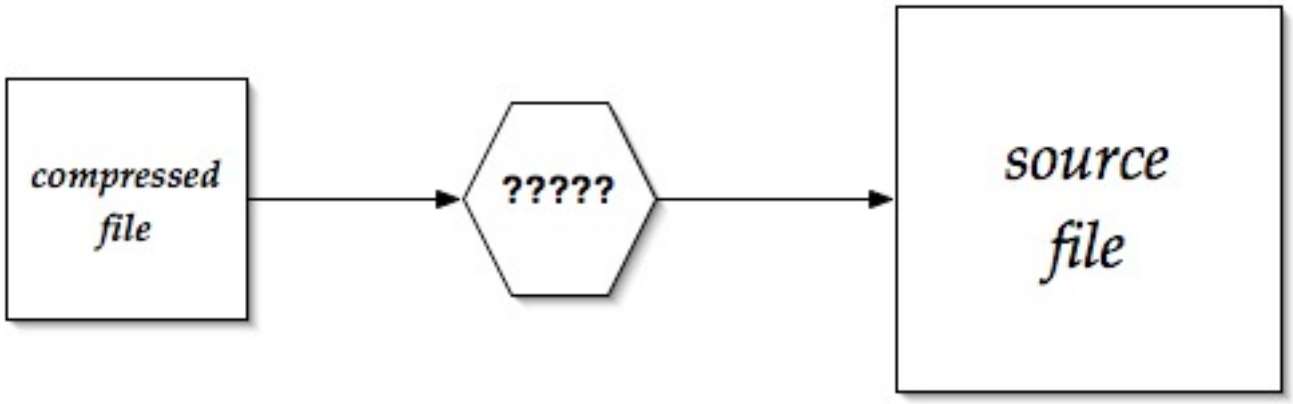
Data Compression: MP3



MP3 is an especially efficient encoding.
"Sweet Dreams" as an `.mp3` file
takes only **4,281,625 bytes**.
(about 11%)

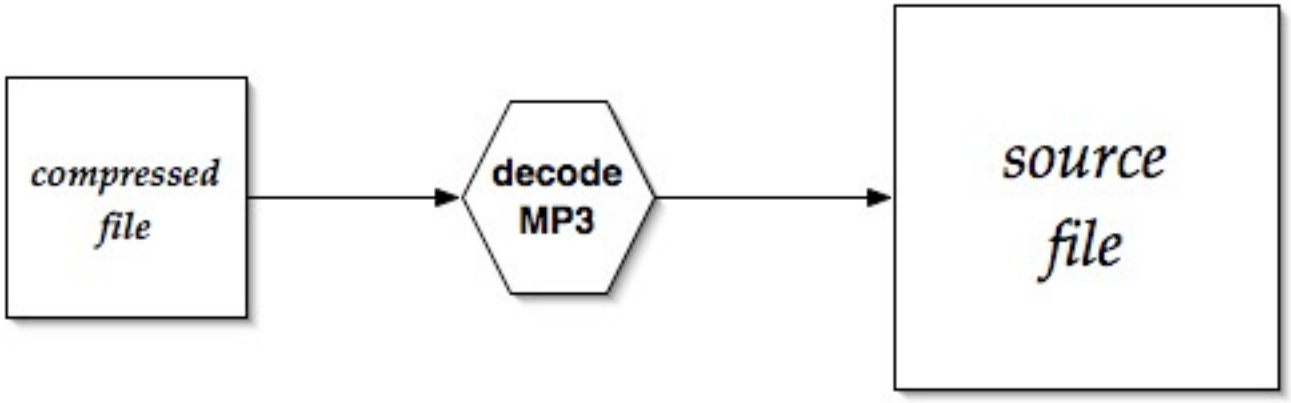
Going Back?

Most compressions are
easily manipulated only in specific way.
(MP3 encoding is great for playing, and not much else.)



Can we go backwards,
and retrieve our original source file?

Data Decompression: MP3



With MP3, no.
The source of its size efficiency is
loss of information.

Lossy versus Lossless Compression

A compression algorithm that loses information
is called **lossy**.

A compression algorithm that retains all information
is called **lossless**.

Lossy algorithms can generate smaller files,
at some cost in the quality of the file for some purposes
— including decompression.

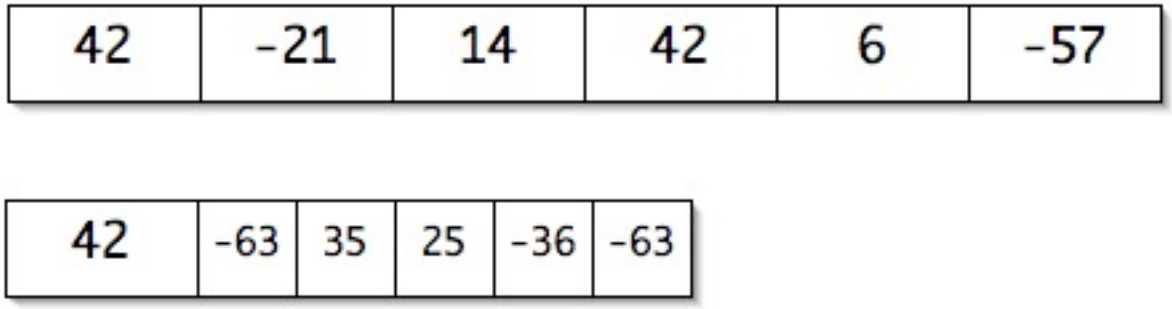
Designing a Compression Algorithm

Our Sound objects use 2 bytes for each sample value.
This gives a range of values of [-32768..32767].

But many sounds have sample-to-sample ranges
that are much smaller.

Designing a Compression Algorithm

Instead of using 2 bytes for each sample value,
we could use 1 byte to record each sample change!



This encoding uses 7 bytes instead of 12 bytes
for this example.

Our Compression Algorithm

Instead of using 2 bytes for each sample value,
we could use 1 byte to record each sample change.

For a Sound of size ***n***,
how many bytes will our encoding use?

Is the algorithm lossy or lossless?

An Exercise in Compression

Write a Sound *method that returns an array of the differences in value between each sound sample.*

Your array will be `getLength()-1` bytes long.

```
public int[] arrayOfDifferences()
{
  int[] differences = new int[ getLength()-1 ];
  // fill in the blank
  return differences;
}
```