Unit “Naught”: Codes

Dave Abel

March 25th, 2016

Outline

- Why do we need codes?
- Solutions
  - Correcting Errors
  - Compression

Why Do We Need Codes?

Why Do We Need Codes?

Why Do We Need Codes?

Why Do We Need Codes?

ERRORS

Cello

Why Do We Need Codes?

ERRORS

Cello

Reason One: Come up with a *code* that is robust to errors!

Why Do We Need Codes?

Reason One: Come up with a *code* that is robust to errors!

Why Do We Need Codes?

Reason One: Come up with a *code* that is robust to errors!

Why *else* Might We Need Codes?

Why *Else* Might We Need Codes?

Why *Else* Might We Need Codes?

Why *Else* Might We Need Codes?

Why *Else* Might We Need Codes?

Why *Else* Might We Need Codes?

Why Do We Need Codes?

error

Reason One: Communication that is robust to errors!

Reason Two: Send information in *compressed form!*

compression

Error Correcting Codes

- Idea: what if our messages fixed themselves?
- Idea: or what if we could at least *tell* our message broke?

Error Correcting Codes

- Idea: what if our messages fixed themselves?
- Idea: or what if we could at least *tell* our message broke?

Suppose I send you our zip code

Error Correcting Codes

- Idea: what if our messages fixed themselves?
- Idea: or what if we could at least *tell* our message broke?

Suppose I send you our zip code

02906

Error Correcting Codes

- Idea: what if our messages fixed themselves?
- Idea: or what if we could at least *tell* our message broke?

Suppose I send you our zip code

Error Correcting Codes

- Idea: what if our messages fixed themselves?
- Idea: or what if we could at least *tell* our message broke?

Suppose I send you our zip code

02906

First Idea:

- Send it twice! (or more)
- If they’re the same, great!
- If they’re different, ask to be sent the message again

Suppose I send you our zip code

02906

02906

First Idea:

- Send it twice! (or more)
- If they’re the same, great!
- If they’re different, ask to be sent the message again

Suppose I send you our zip code

First Idea:

- Send it twice! (or more)
- If they’re the same, great!
- If they’re different, ask to be sent the message again

Suppose I send you our zip code

02905
02906

Clicker Question!

- Send it twice! (or more)
- If they’re the same, great!
- If they’re different, ask to be sent the message again

Q: Is this scheme perfect? Will you *always* catch errors?

Clicker Question!

- Send it twice! (or more)
- If they’re the same, great!
- If they’re different, ask to be sent the message again

Q: Is this scheme perfect? Will you *always* catch errors?

[A] Yes [B] No [C] I’m Confused

Clicker Answer!

- Send it twice! (or more)
- If they’re the same, great!
- If they’re different, ask to be sent the message again

Q: Is this scheme perfect? Will you *always* catch errors?

[A] Yes **[B] No** [C] I’m Confused

Clicker Answer!

- Send it twice! (or more)
- If they’re the same, great!
- If they’re different, ask to be sent the message again

Suppose I send you our zip code

02906

02906

Clicker Answer!

- Send it twice! (or more)
- If they’re the same, great!
- If they’re different, ask to be sent the message again

Suppose I send you our zip code

Clicker Answer!

- Send it twice! (or more)
- If they’re the same, great!
- If they’re different, ask to be sent the message again

Suppose I send you our zip code

02905
02905

Clicker Answer!

- Send it twice! (or more)
- If they’re the same, great!
- If they’re different, ask to be sent the message again

Suppose I send you our zip code

First Idea:

- Send it twice! (or more)
- If they're the same, great!
- If they're different, ask to be sent the message again
- Note: not perfect!
- The more copies we send, the better we do.
- But then we have to send way more stuff!

Second Idea: Checksum

- Idea: add some information to the message that can help detect errors!

Second Idea: Checksum

- Idea: add some information to the message that can help detect errors!
- Checksum: add the sum of the message *mod 10* to the end!
- Example: 02906, the sum is 0 + 2 + 9 + 0 + 6 = 17
- So we add 17 *mod 10* to the end, which is **029067**.
- Then check to make sure the checksum is still correct!

Clicker Question!

What is the checksum for 42304?

[A] 423041

[B] 423049

[C] 423047

[D] 423043

Clicker Answer!

What is the checksum for 42304?

[A] 423041

[B] 423049

[C] 423047

[D] 423043

Clicker Answer!

What is the checksum for 42304?

[A] 423041 [B] 423049

[C] 423047 **[D] 423043**

Since: 4 + 2 + 3 + 4 = 13, and 13 mod 10 = 3

Clicker Question!

- Idea: add some information to the message that can help detect errors!
- Checksum: add the sum of the message *mod 10* to the end!

Q: Is this scheme perfect? Will you *always* catch errors?

[A] Yes [B] No [C] I'm Confused

Clicker Answer!

- Idea: add some information to the message that can help detect errors!
- Checksum: add the sum of the message *mod 10* to the end!

Q: Is this scheme perfect? Will you *always* catch errors?

[A] Yes **[B] No** [C] I'm Confused

Clicker Answer!

5040

Q: Is this scheme perfect? Will you *always* catch errors?

[A] Yes **[B] No** [C] I'm Confused

Clicker Answer!

5040 ⟶ 50409

Q: Is this scheme perfect? Will you *always* catch errors?

[A] Yes **[B] No** [C] I'm Confused

Clicker Answer!

5040 ⟶ 50409 ⟶

Q: Is this scheme perfect? Will you *always* catch errors?

[A] Yes **[B] No** [C] I'm Confused

Clicker Answer!

5040 ⟶ 50409 ⟶  ⟶ 50301

Q: Is this scheme perfect? Will you *always* catch errors?

[A] Yes **[B] No** [C] I'm Confused

Error Correcting Codes

- In a computer, communication errors happen regularly!
  - Errors in transmitting to and from the internet
  - Errors reading/writing from our computers memory
    - Old news: Errors reading/writing to CDs, floppies.
- We want to know when errors happen so we can fix them.
- Solutions:
  - Send copies! (But too much space)
  - Send a checksum!

Compression

- Idea: We can turn *big* things into *small* things that effectively *preserve the main information*.

Compression

- Idea: We can turn *big* things into *small* things that effectively *preserve the main information*.

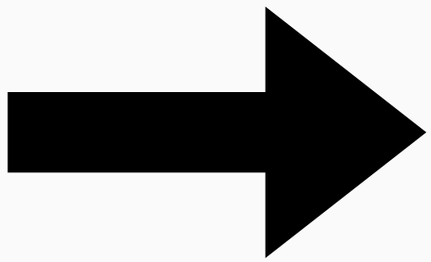
Compression

- Idea: We can turn *big* things into *small* things that effectively *preserve the main information*.

, but way bigger

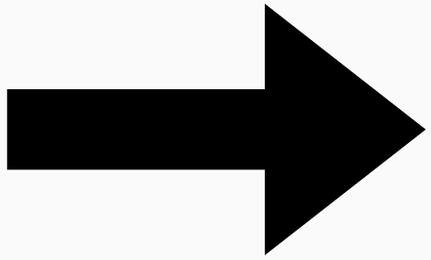
Simple Explanations

Challenge 1:



Simple Explanations

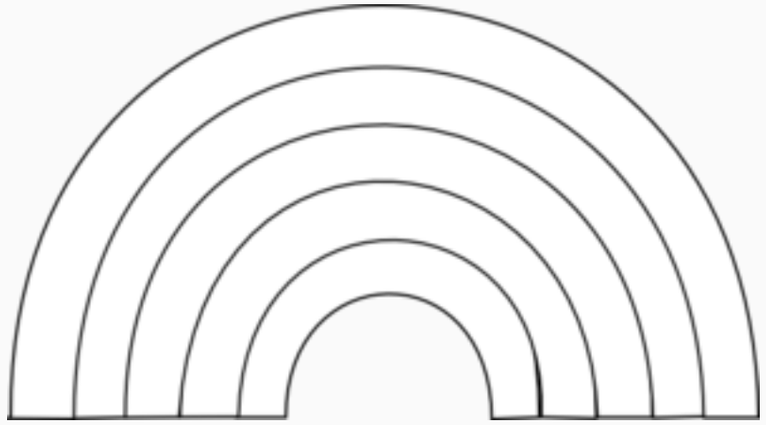
Challenge 1:



Shortest description: east

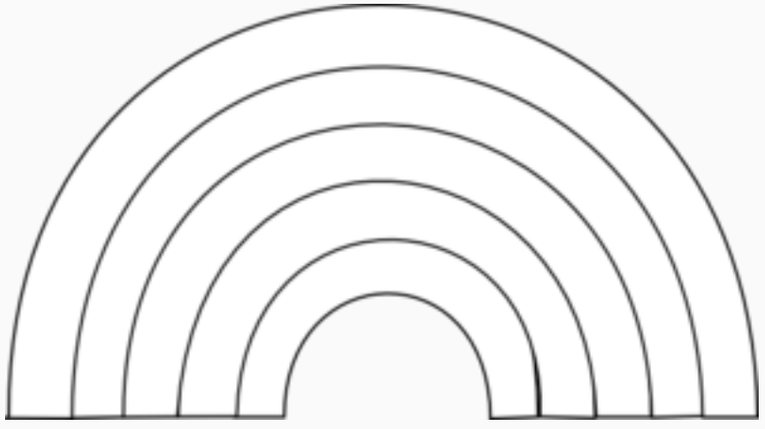
Simple Explanations

Challenge 2:



Simple Explanations

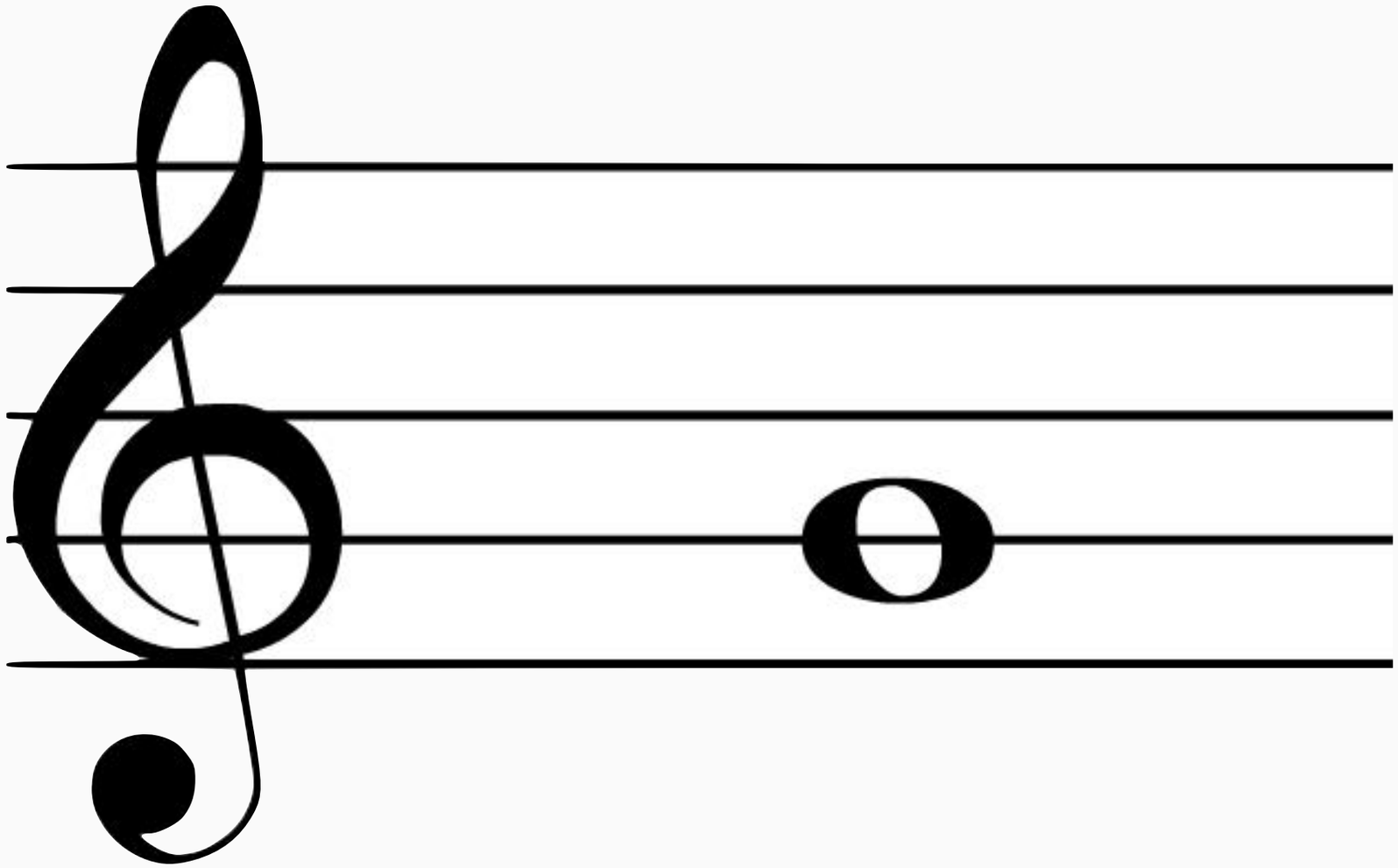
Challenge 2:



Shortest description: colorless rainbow

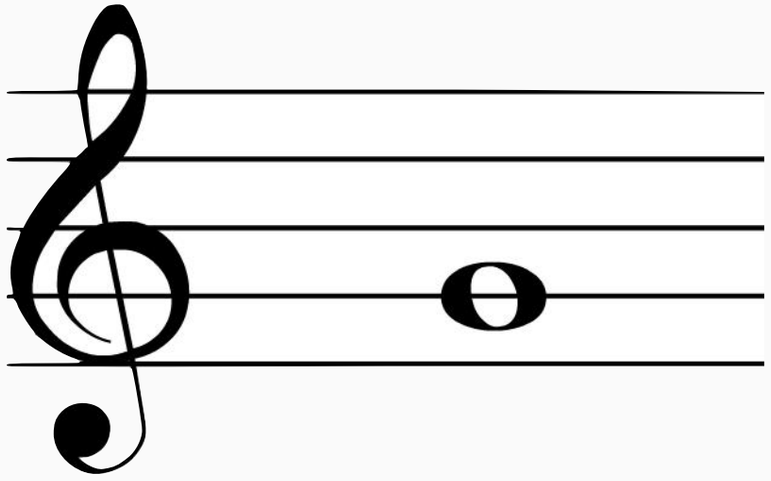
Simple Explanations

Challenge 4:



Simple Explanations

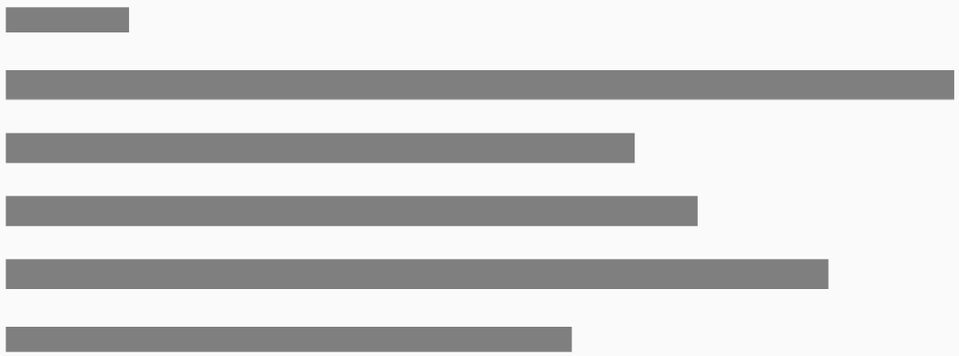
Challenge 4:



Shortest description: whole note g

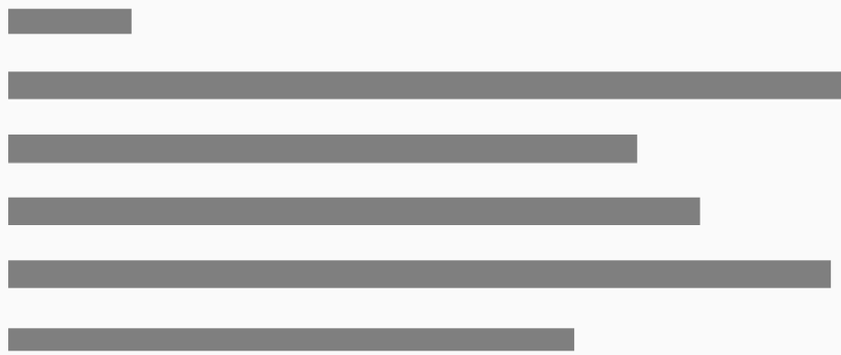
Simple Explanations

Challenge 5:



Simple Explanations

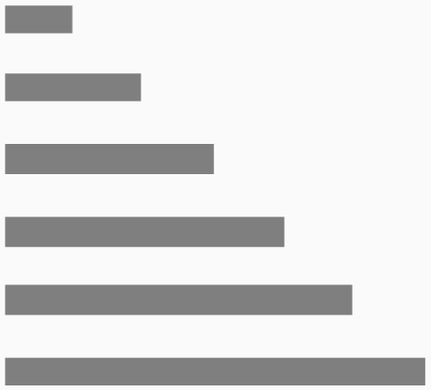
Challenge 5:



Shortest description: six bar histogram switch 2 and 6

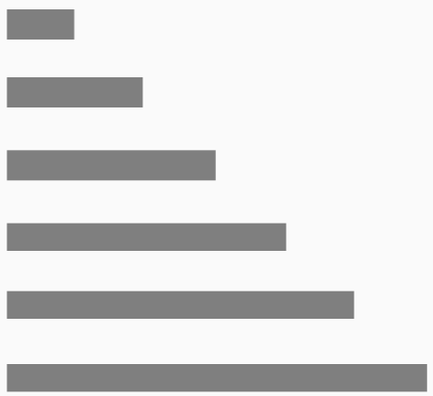
Simple Explanations

Challenge 6:



Simple Explanations

Challenge 6:



Shortest description: bars ordered

Compression

- Q: How can we make an object *as small as possible*, but still preserve what the object is?

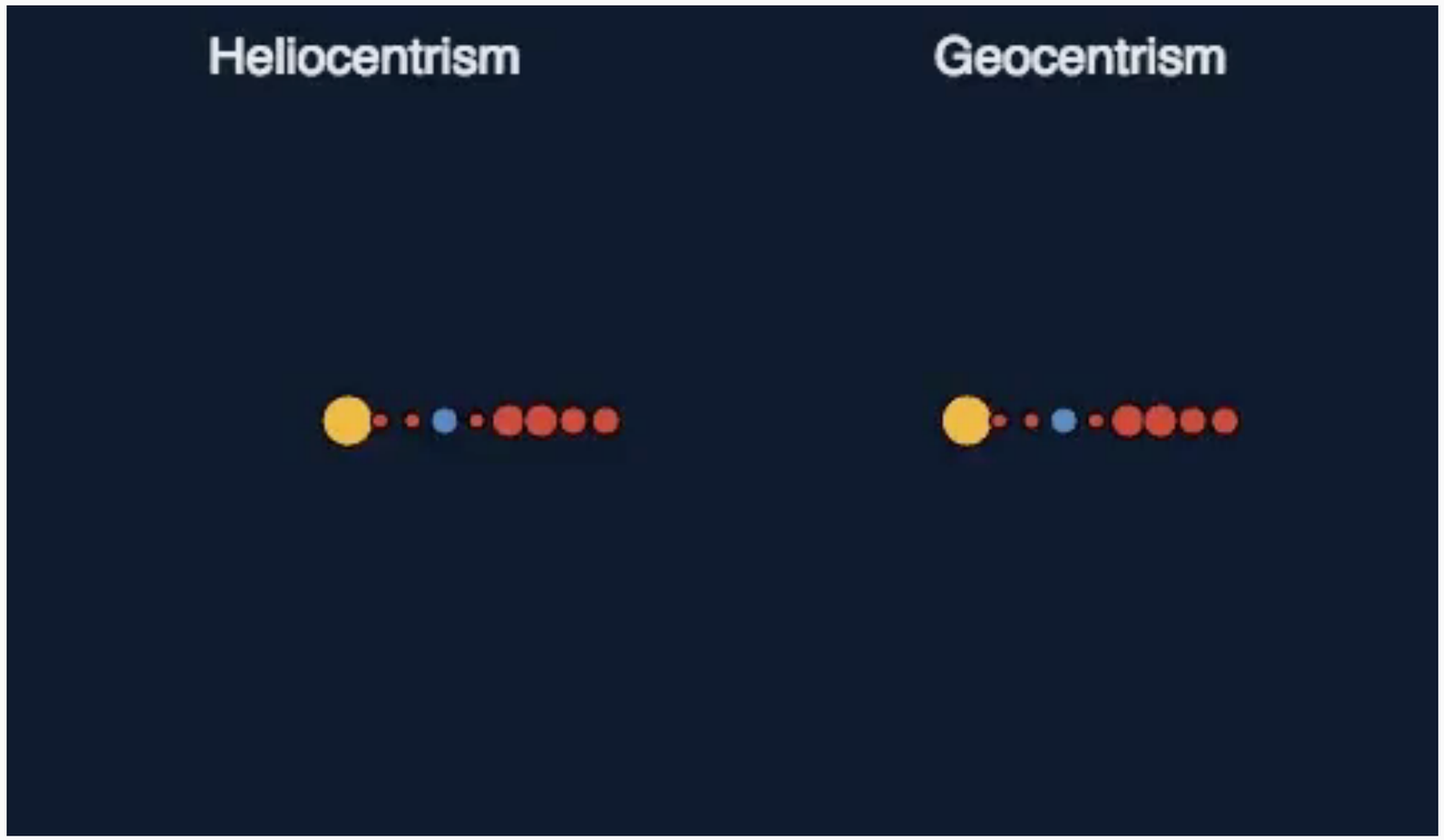
, but way bigger

Compression

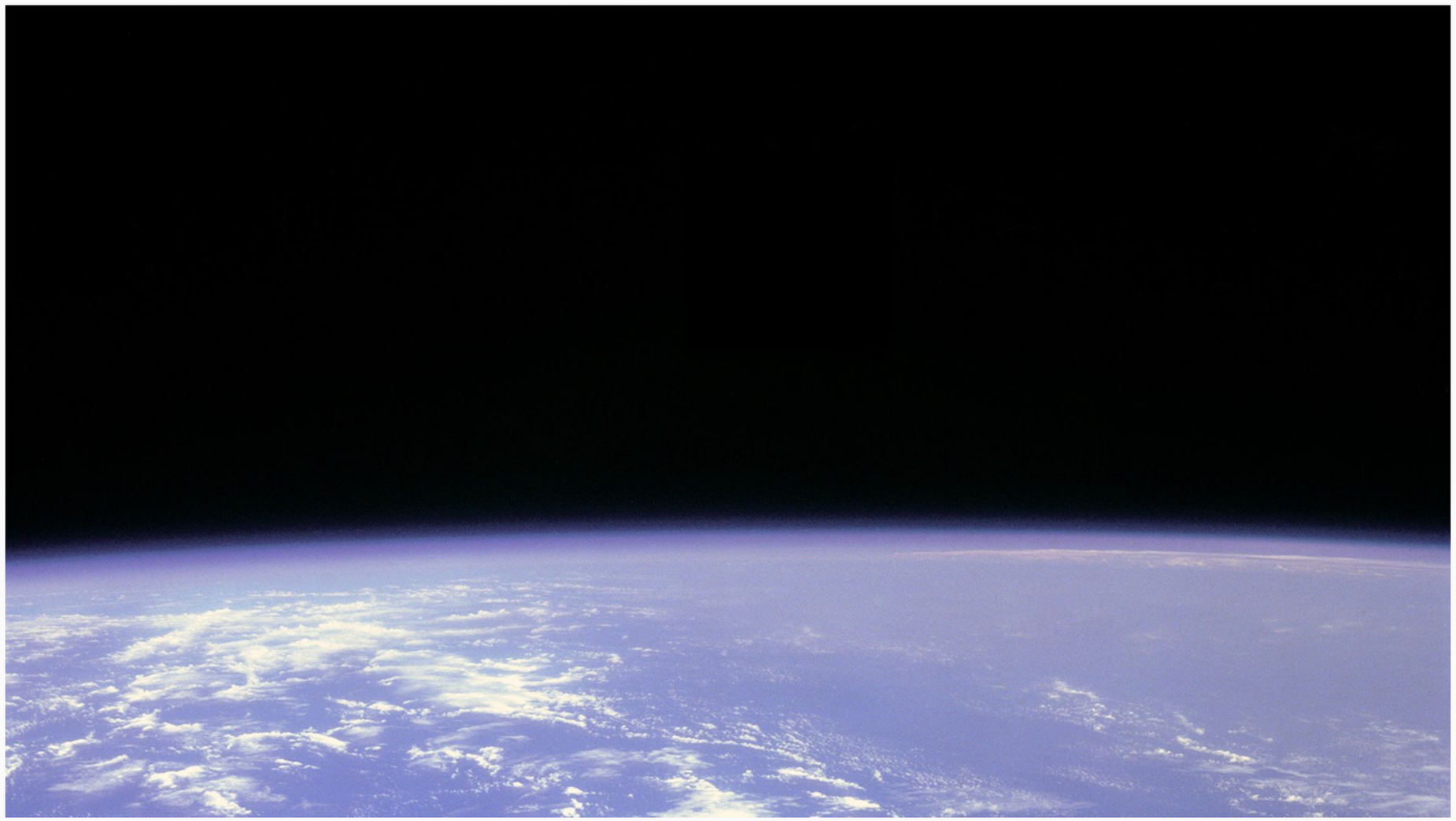
- Q: How can we make an object *as small as possible*, but still preserve what the object is?

Compression

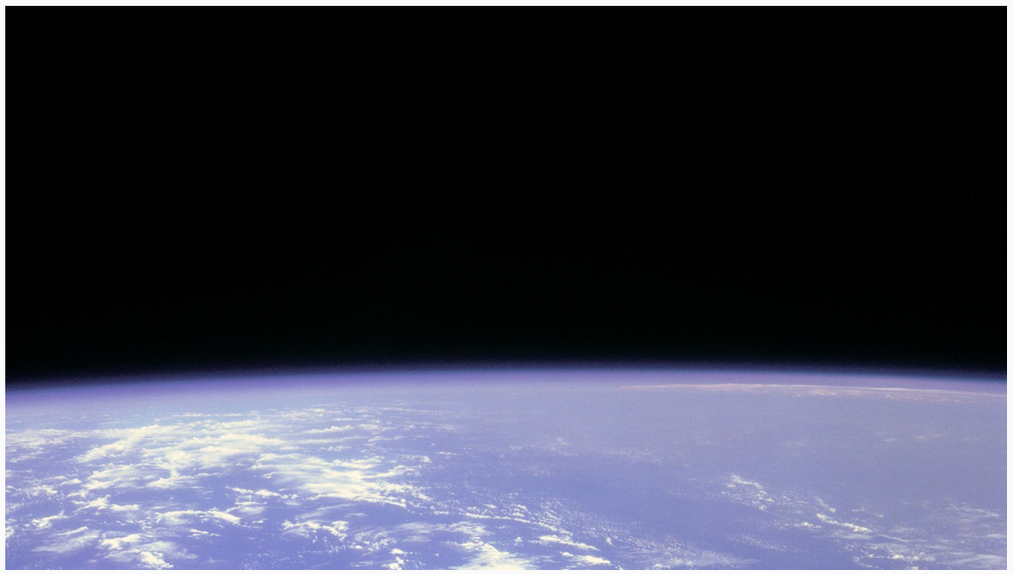
- Q: How can we make an object *as small as possible*, but still preserve what the object is?



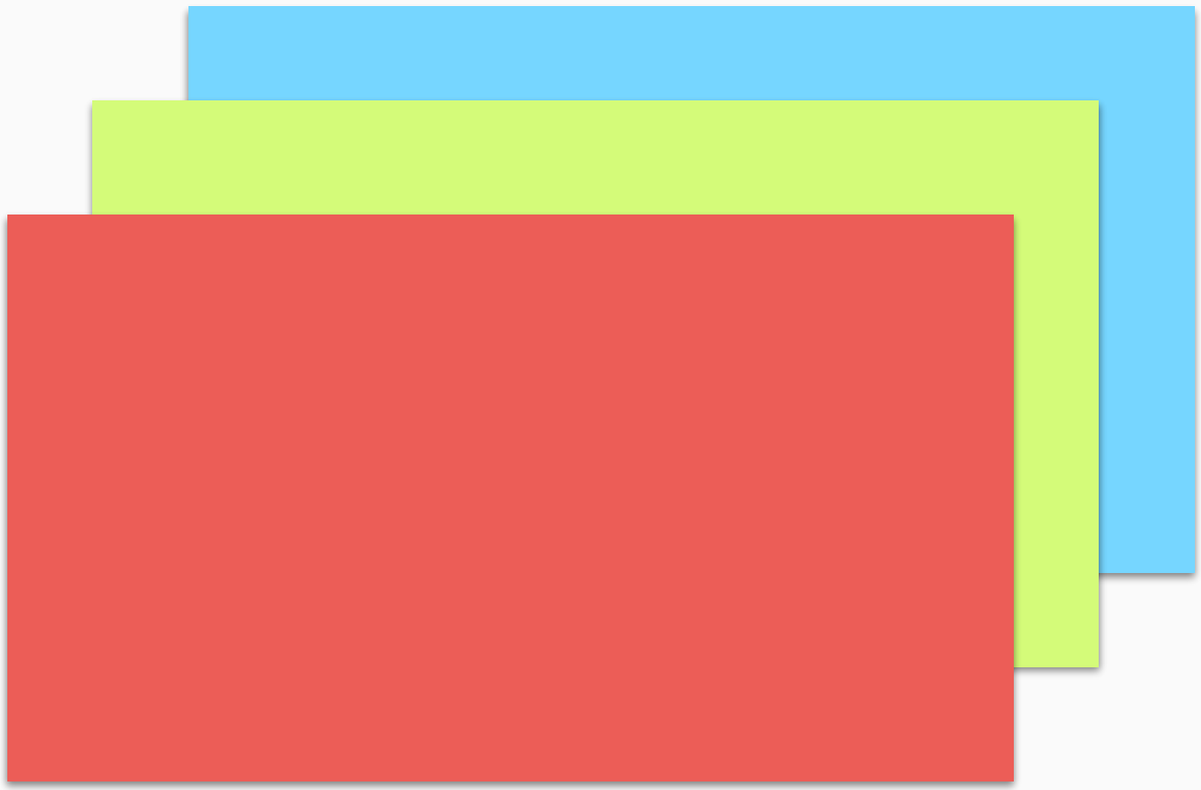
Compression: Intuition



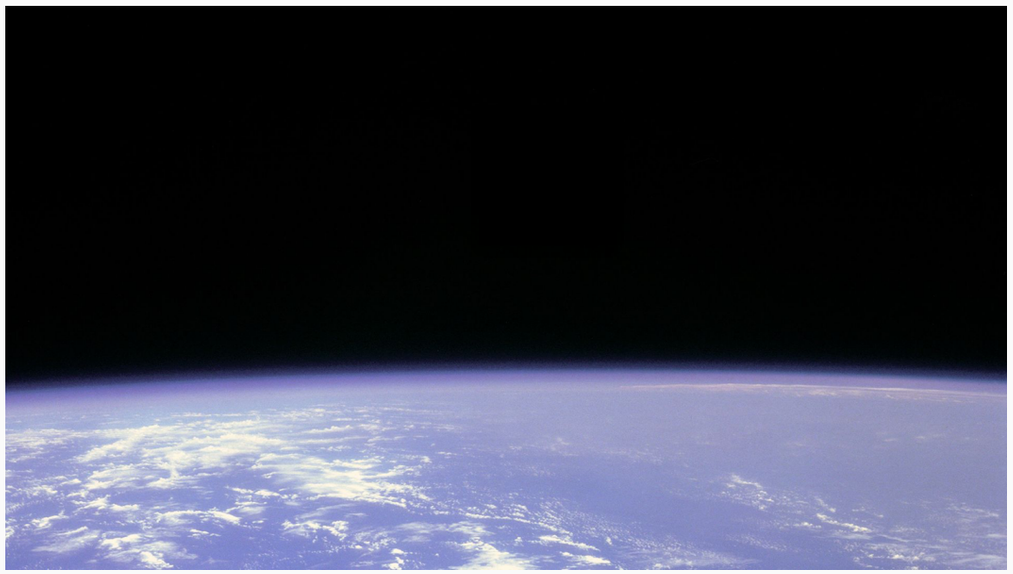
Compression: Intuition



=

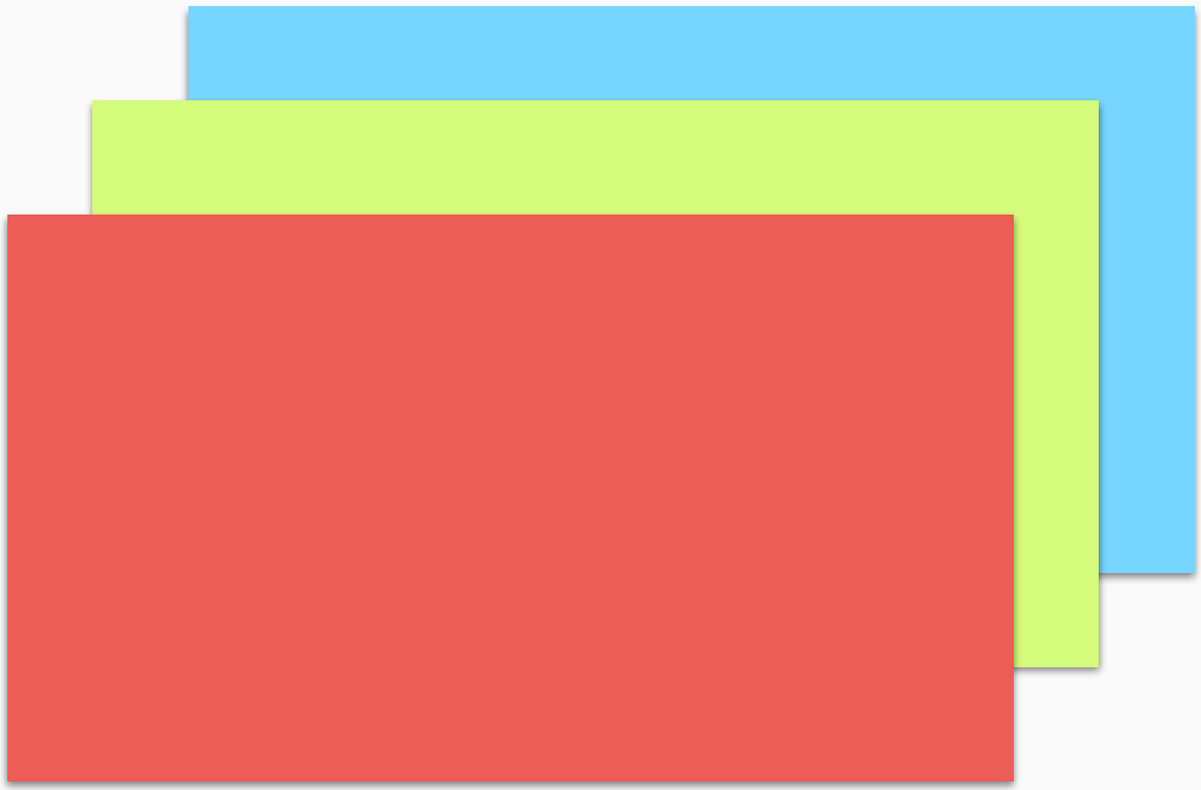


Compression: Intuition

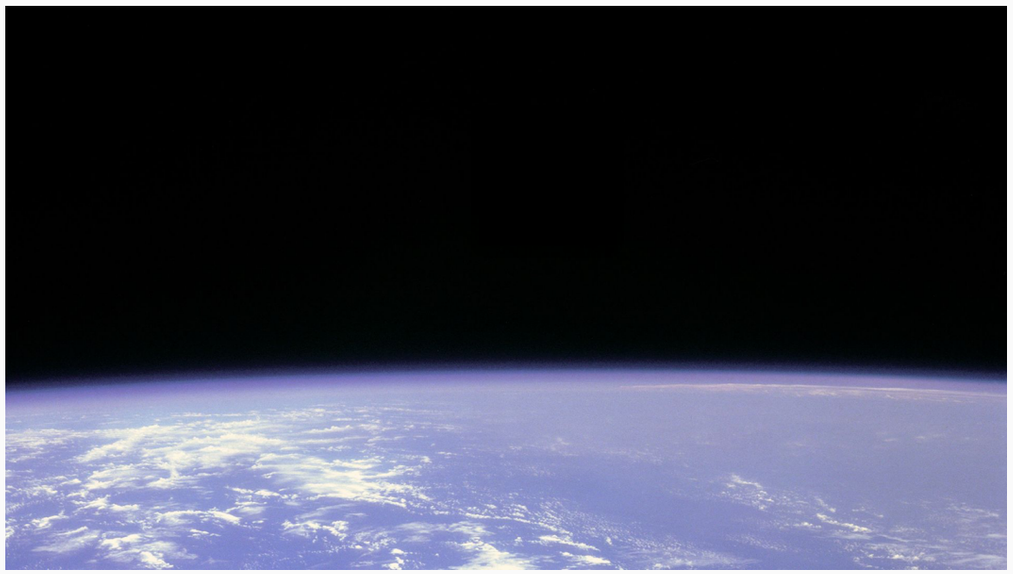


1Kb

=

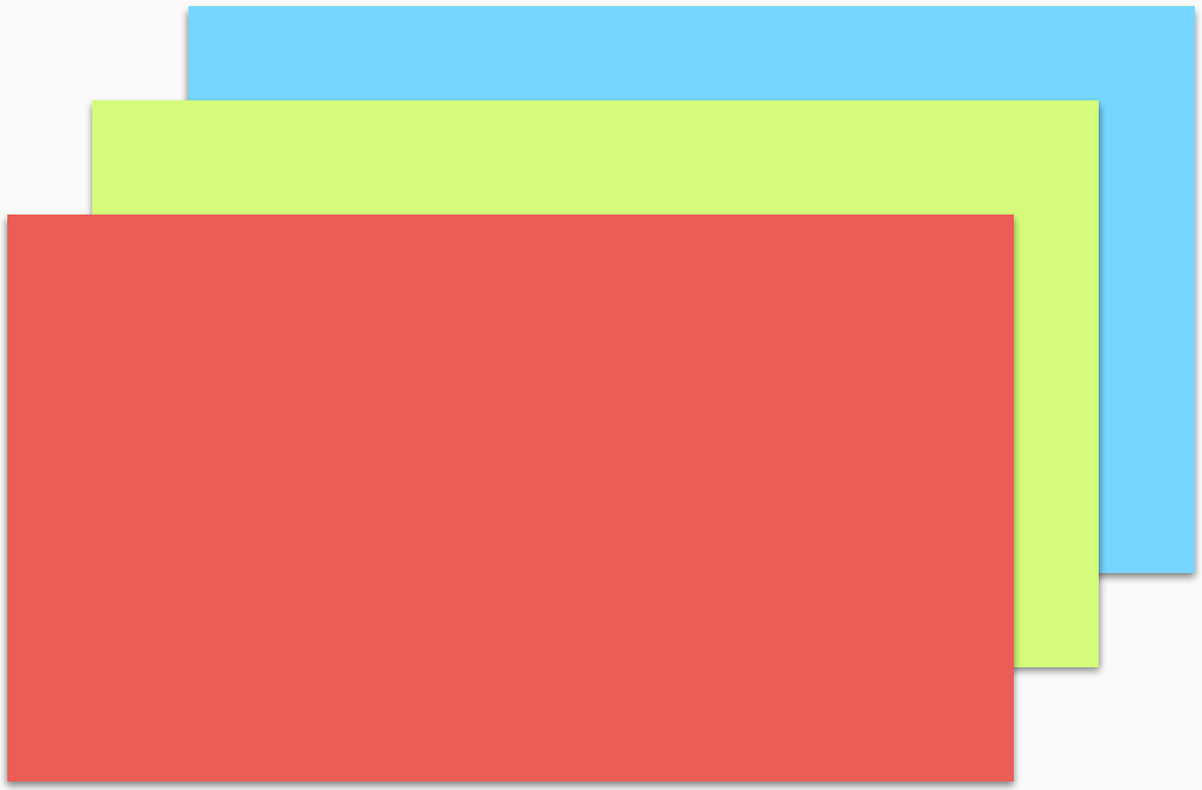


Compression: Intuition



1Kb

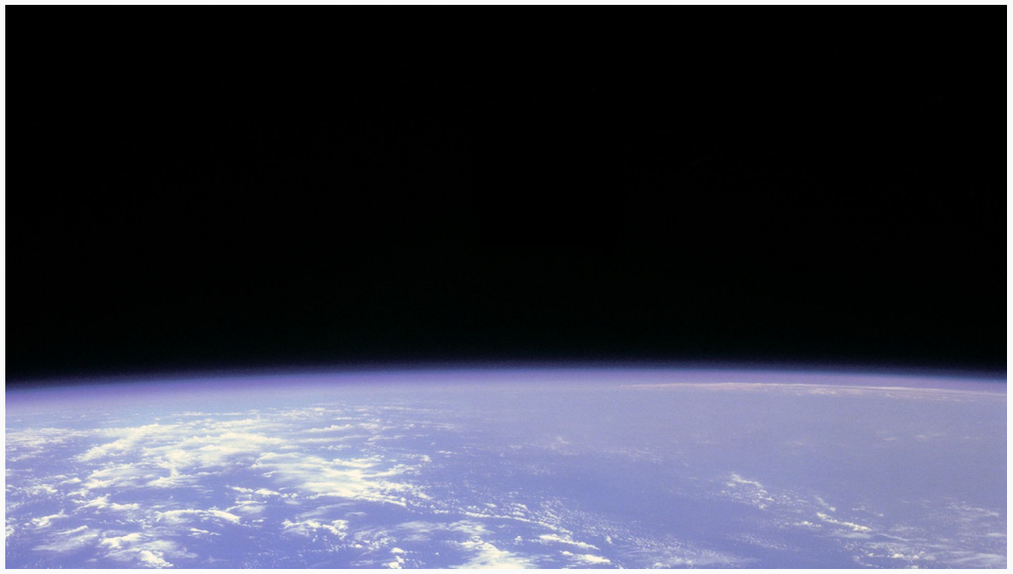
=



1Kb

Compression: Intuition

Everything else, black

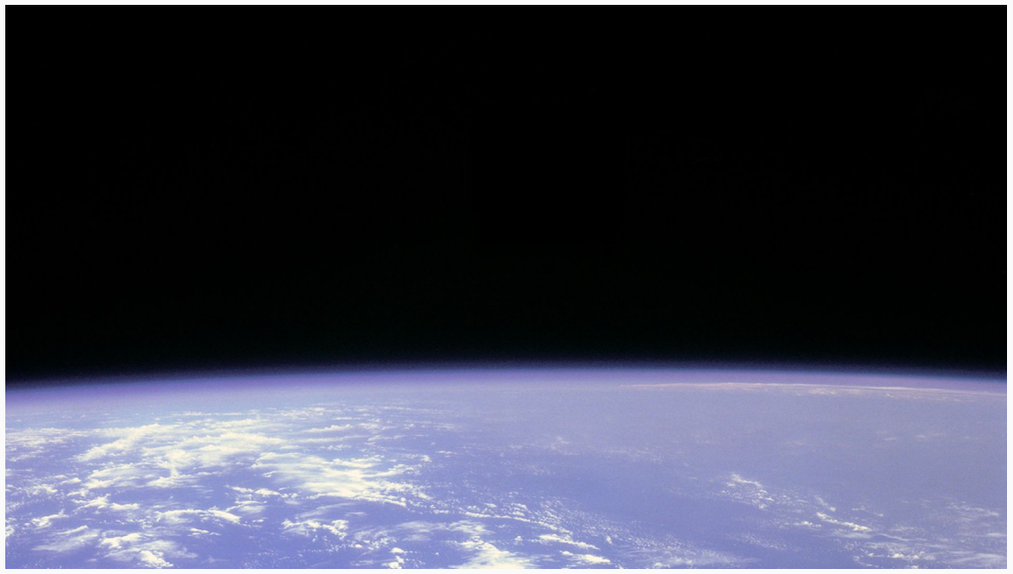


1Kb

=

0.3Kb

Compression: Intuition



1Kb

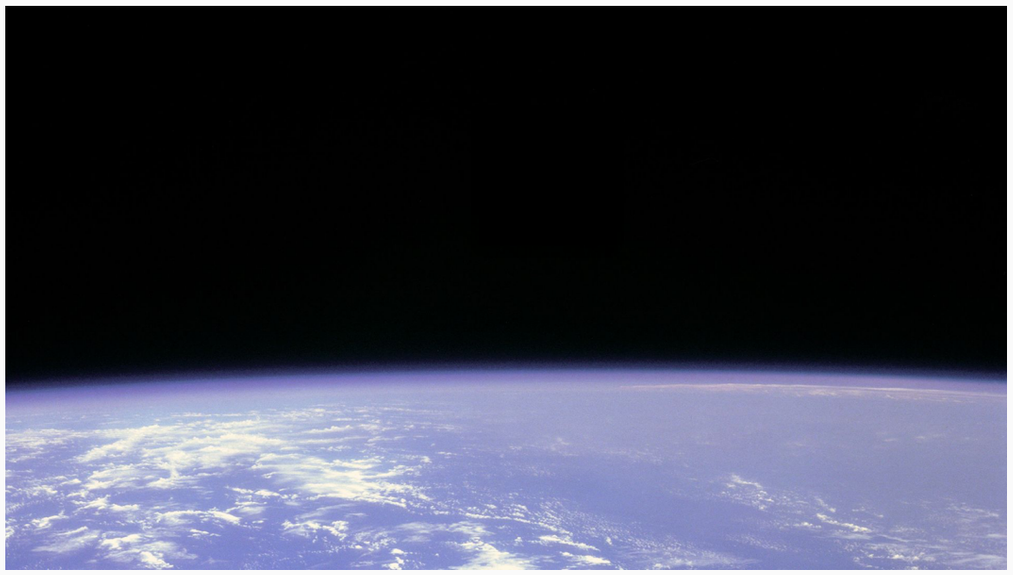
=

Everything else, black

0.3Kb

Shaved off .7 Kilobytes!

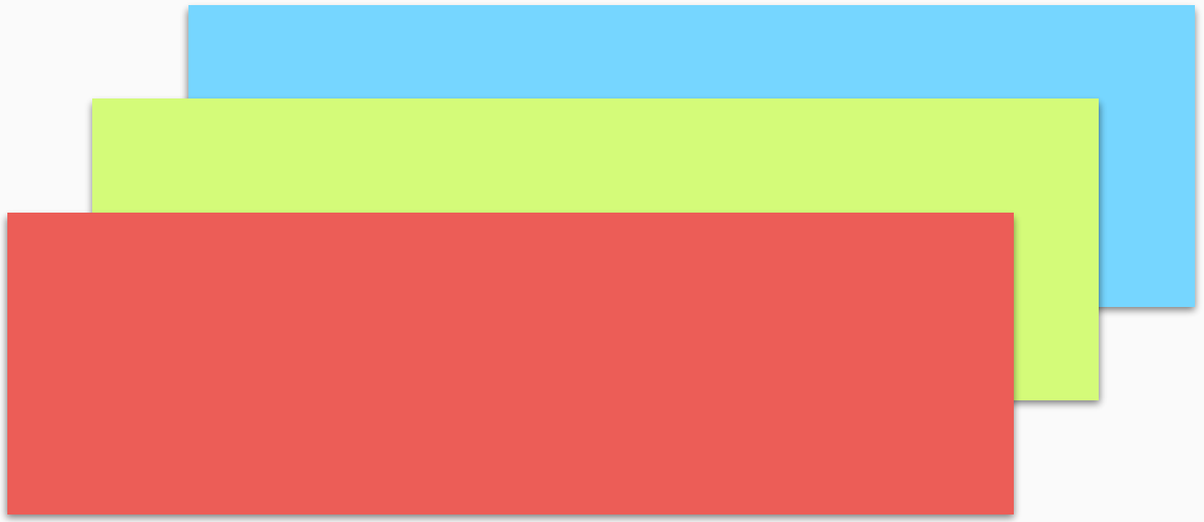
Compression



1Kb

=

Everything else, black

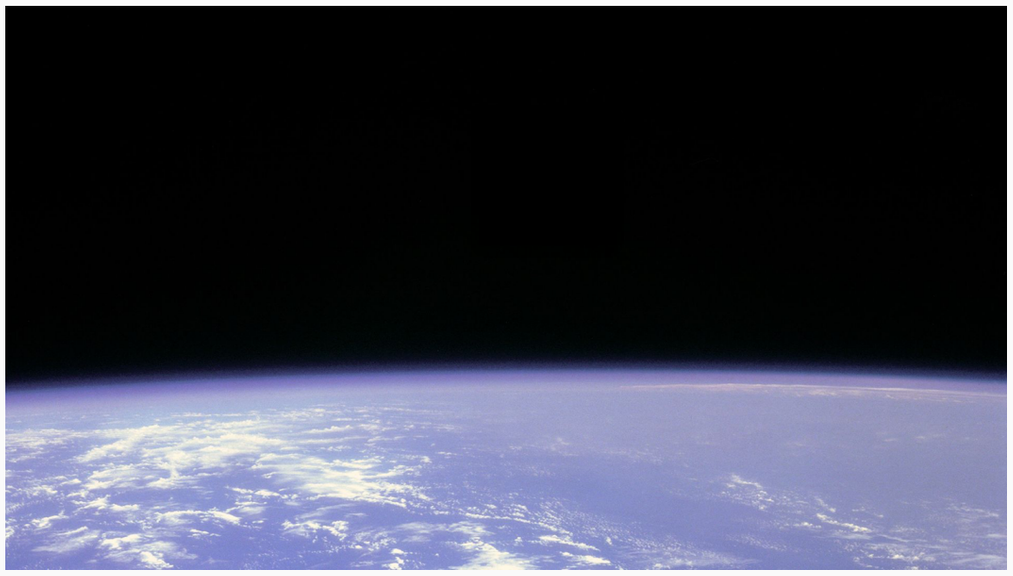


0.3Kb

Critical: we can do this with everything

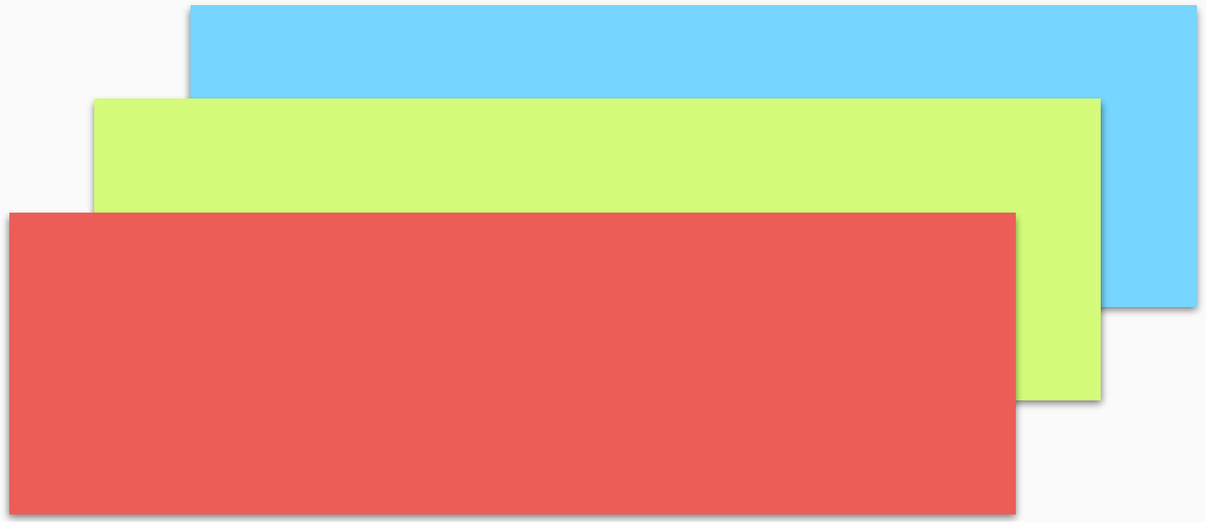
Clicker Question:

Everything else, black



1Kb

=



0.3Kb

Critical: we can do this with everything

Algorithmic Information

Critical: we can do this with everything

For a given object…

Q: What is the shortest algorithm that outputs the object we want to describe?

Algorithmic Information

Q: What is the shortest algorithm that outputs the object we want to describe?

Let’s consider sequences of english characters…

Algorithmic Information

Q: What is the shortest algorithm that outputs the object we want to describe?

Let's consider sequences of english characters…

Example: Scooooooooooooooooooooooby doo!

Algorithmic Information

Q: What is the shortest algorithm that outputs the object we want to describe?

Let’s consider sequences of english characters…

Example: Scooooooooooooooooooooooby doo!

Algorithmic Information

Q: What is the shortest algorithm that outputs the object we want to describe?

Let's consider sequences of english characters…

Example: Scooooooooooooooooooooby doo!

Shaggy's algorithm:
Output "Sc", then 20 "o"'s, then "by doo!"

Algorithmic Information

Q: What is the shortest algorithm that outputs the object we want to describe?

Let's consider sequences of english characters…

Example: aaaaaaaaaaaaaaaaaaaaaaaaa

Shaggy's algorithm:
Output 23 "a"

Problem: Compress Words

- INPUT: A phrase in english
- OUTPUT: A compressed version of that phrase.

Algorithm: Run Length Encoding

- Compress repeated sequences into #repeats, then the letter:
  - E.g. “fooooooooooxesssss!”
    - Becomes: 1f11o1x1e5s
  - E.g. “abbbbbaa zabbbaa”
    - Becomes: 1a5b2a 1z1a3b2a

Clicker Question!

Q: What is the run length encoding for “Wobbbless and bbbbbles”?

[A] 1Wobbless 1and 1bbbbbles

[B] 1W1o3b1l1e2s 1a1n1d 5b1l1e1s

[C] Wo3b1l1e2s and 5bles

[D] I’m confused

Clicker Answer!

Q: What is the run length encoding for “Wobbbless and bbbbbles”?

[A] 1Wobbless 1and 1bbbbbles

[B] 1W1o3b1l1e2s 1a1n1d 5b1l1e1s

[C] Wo3b1l1e2s and 5bles

[D] I’m confused

Clicker Question!

Q: What is, “1W5o1d1e1n 1B3i1r1d1s”, when *uncompressed*?

[A] Wooooooden Biiirds

[B] Wooden Birds

[C] Wooden Biiirds

[D] Wooooooden BBBirds

Clicker Answer!

Q: What is, “1W5o1d1e1n 1B3i1r1d1s”, when *uncompressed*?

[A] Woooooooden Biiirds

[B] Wooden Birds

[C] Wooden Biiirds

[D] Wooooooden BBBirds

What Do We Think of This?

- Compress repeated sequences into #repeats, then the letter:
  - E.g. “fooooooooooxesssss!”
    - Becomes: 1f11o1x1e5s
  - E.g. “abbbbbaa zabbbaa”
    - Becomes: 1a5b2a 1z1a3b2a

What Do We Think of This?

- Two drawbacks of note:
  - Q: Are repeated sequences that regular in English?
  - Q: What about numbers?

Algorithmic Information

Q: What is the shortest algorithm that outputs the object we want to describe?

Algorithmic Information

Q: What is the shortest algorithm that outputs the object we want to describe?

This question ends up being *super fascinating*.

Algorithmic Information

Q: What is the shortest algorithm that outputs the object we want to describe?

This question ends up being *super fascinating*.

Q: What objects generally have shorter algorithms?

Q: What objects generally have complex algorithms?

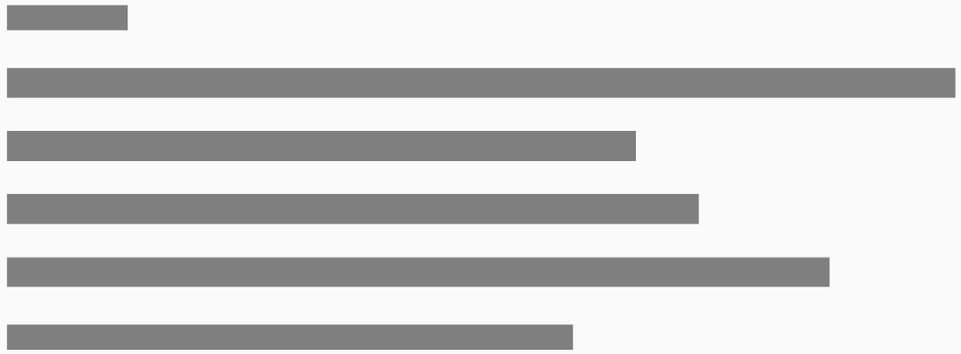
Q: Given an object, how can *find* the shortest algorithm?

Algorithmic Information

Q: What is the shortest algorithm that outputs the object we want to describe?

This question ends up being *super fascinating*.

Q: What objects generally have more complex algorithms?

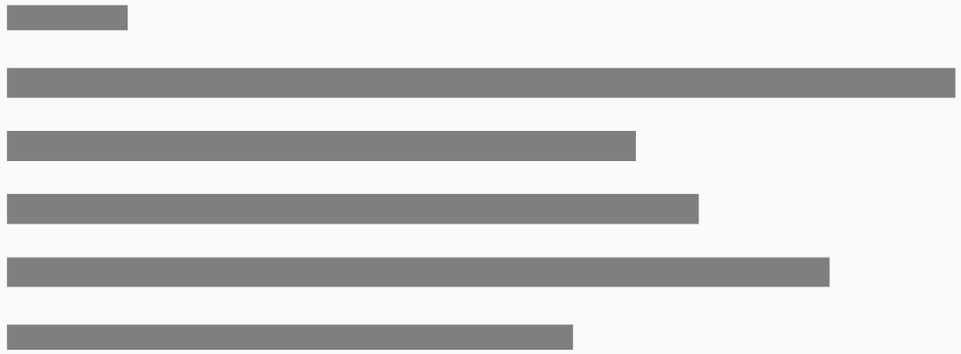


Algorithmic Information

Q: What is the shortest algorithm that outputs the object we want to describe?

This question ends up being *super fascinating*.

Q: What objects generally have more complex algorithms?



Have a Great Break!