Entropy & coding

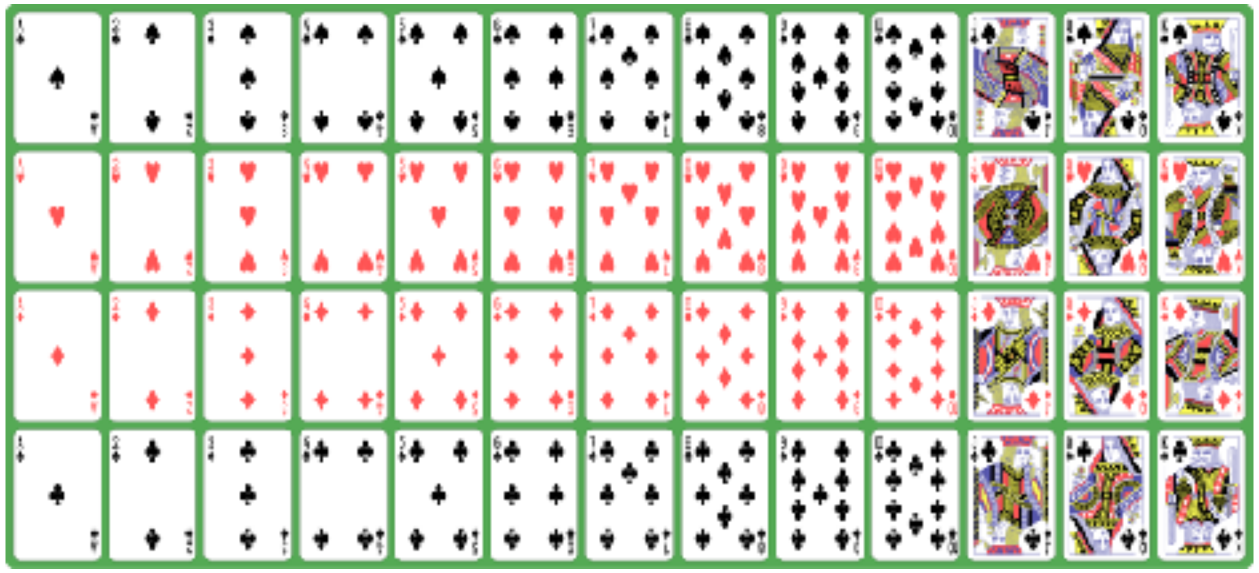
Ben Langmead

JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

Department of Computer Science

Entropy & coding

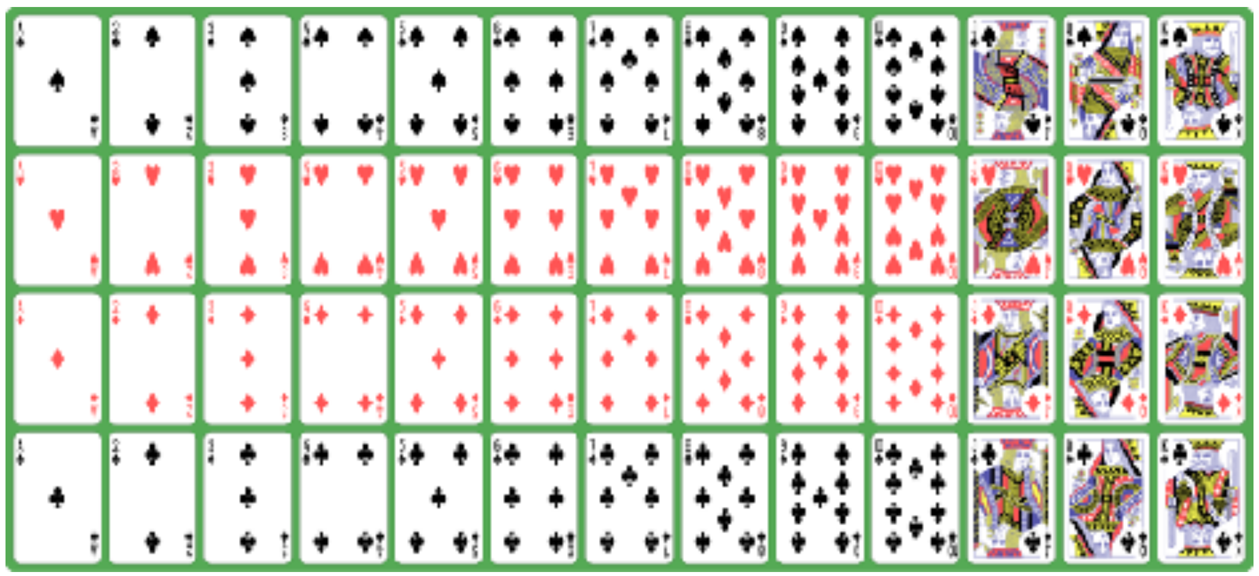
Let's identify items with ***codes***, made of bits



https://commons.wikimedia.org/wiki/File:English_pattern_playing_cards_deck_PLUS.svg

Entropy & coding

Let's identify items with ***codes***, made of bits



Say, code = rank + (13 * suit)

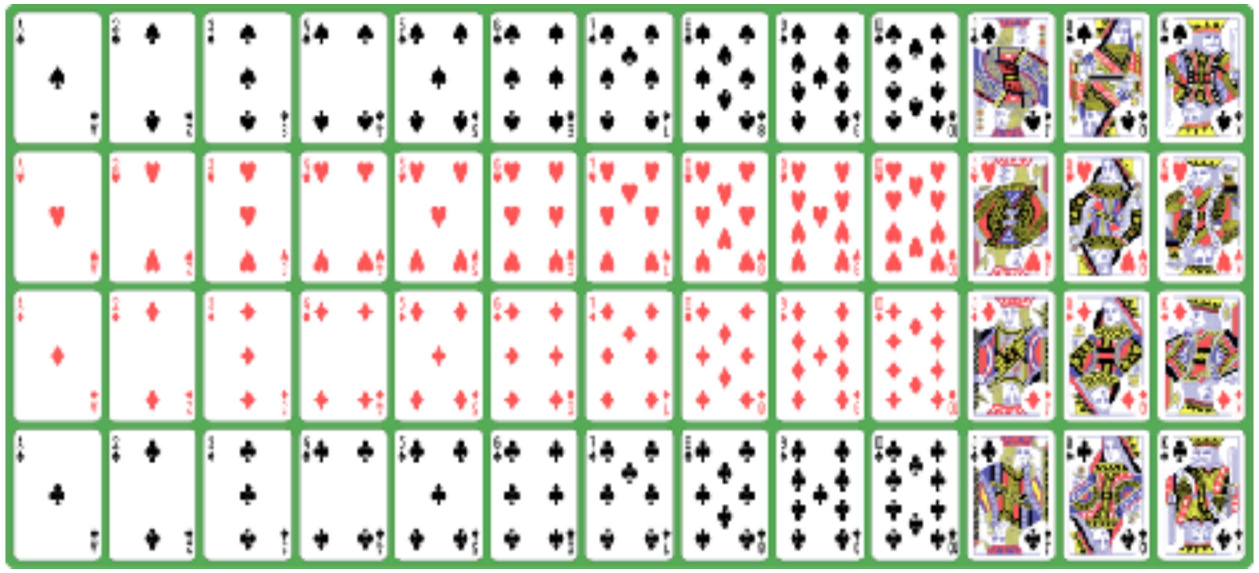
Where Ace = 0, Jack = 10, ...

♠ = 0, ♥= 1, ...

https://commons.wikimedia.org/wiki/File:English_pattern_playing_cards_deck_PLUS.svg

Entropy & coding

Let's identify items with ***codes***, made of bits



A ♠	
2 ♠	
3 ♠	
4 ♠	
⋮	
10 ♣	
J ♣	
Q ♣	
K ♣	

Say, code = rank + (13 * suit)

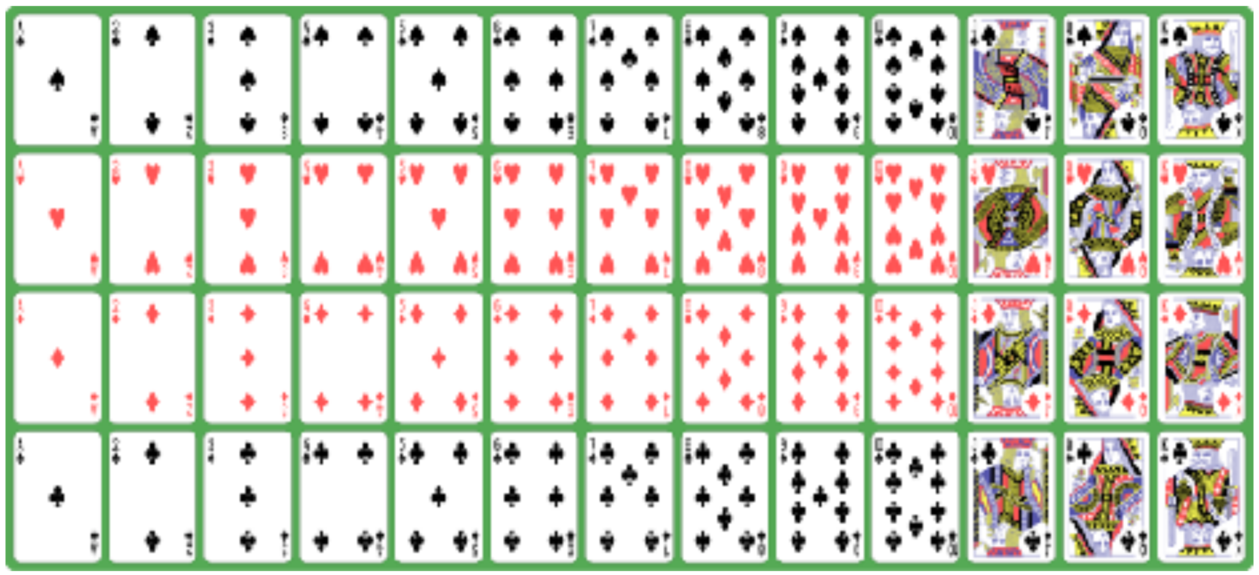
Where Ace = 0, Jack = 10, ...

♠ = 0, ♥= 1, ...

https://commons.wikimedia.org/wiki/File:English_pattern_playing_cards_deck_PLUS.svg

Entropy & coding

Let's identify items with ***codes***, made of bits



Say, code = rank + (13 * suit)

Where Ace = 0, Jack = 10, ...

♠ = 0, ♥= 1, ...

A ♠	0
2 ♠	1
3 ♠	10
4 ♠	11

⋮

10 ♣	110000
J ♣	110001
Q ♣	110010
K ♣	110011

https://commons.wikimedia.org/wiki/File:English_pattern_playing_cards_deck_PLUS.svg

Entropy & coding

How many bits are required to encode items from universe U?

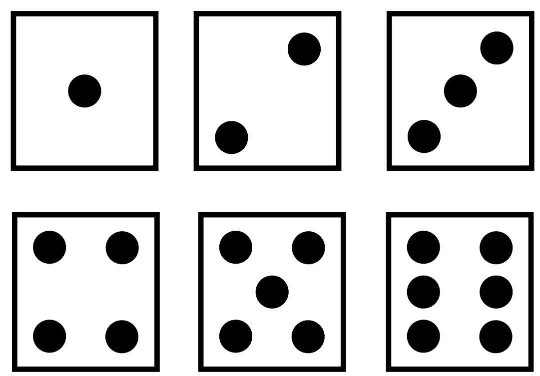
$$H_{wc}(U) = \log_2 |U|$$

If codes must have ***same*** length, length must be $\geq \log_2(|U|)$, best choice is $\lceil \log_2(|U|) \rceil$

If codes can have ***various*** lengths, *longest* code must be $\geq \log_2(|U|)$

Entropy

How many bits required to identify an item from this set?



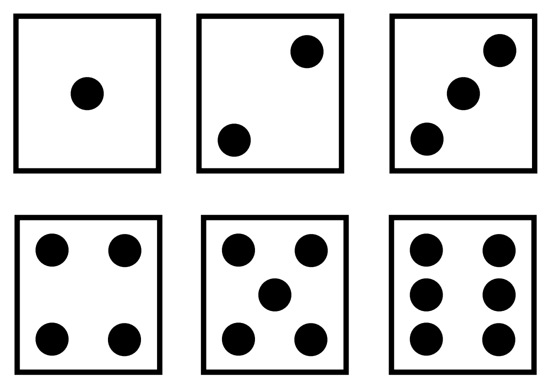
(37 or 38 slots)

https://commons.wikimedia.org/wiki/File:13-02-27-spielbank-wiesbaden-by-RalfR-094.jpg

Entropy

How many bits required to identify an item from this set?

1 bit



3 bits

(37 or 38 slots)

6 bits

https://commons.wikimedia.org/wiki/File:13-02-27-spielbank-wiesbaden-by-RalfR-094.jpg

Entropy

$$H_{wc}(U) = \log_2 |U|$$

This is ***worst-case entropy***

If $|U| = 2^n$, then

Entropy

$$H_{wc}(U) = \log_2 |U|$$

This is ***worst-case entropy***

If $|U| = 2^n$, then $H_{wc}(U) = \ \ n$

Entropy

$$H_{wc}(U) = \log_2 |U|$$

This is ***worst-case entropy***

If $|U| = 2^n$, then $H_{wc}(U) = \;\; n$

If $U = \{$length-n strings from $\Sigma = \{1, \ldots, \sigma\}\}$,

then

Entropy

$$H_{wc}(U) = \log_2 |U|$$

This is ***worst-case entropy***

If $|U| = 2^n$, then $H_{wc}(U) = \ n$

If $U = \{$length-n strings from $\Sigma = \{1, \ldots, \sigma\}\}$,

then $H_{wc}(U) = \ \log_2 \sigma^n = n \log_2 \sigma$

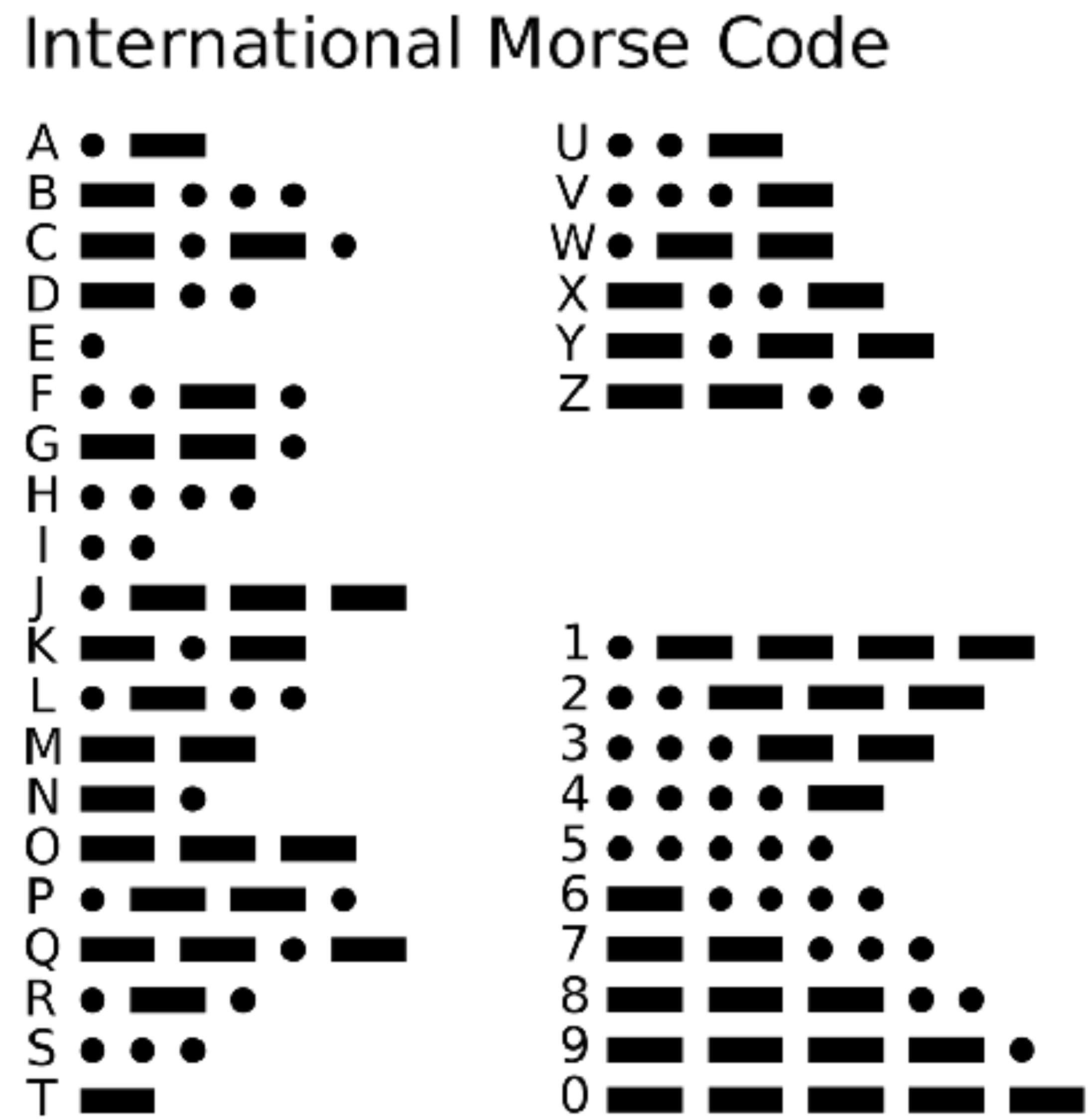
Entropy

If codes can vary in length, we can use shorter codes for more frequent events

Seeking to minimize average (or *expected*) code length $\bar{\ell}$

$$\bar{\ell} = \sum_{u \in U} \Pr(u) \cdot \ell(u)$$

$\ell(u)$ = length of code for u



Entropy

Instead of items $u \in U$, let's think of a discrete r.v. X and its sample space Ω & probability function $\Pr$

$$H(X) = \sum_{s \in \Omega} \Pr(s) \cdot \log_2 \frac{1}{\Pr(s)}$$

Entropy

Instead of items $u \in U$, let's think of a discrete r.v. X and its sample space Ω & probability function $\Pr$

$$H(X) = \sum_{s \in \Omega} \Pr(s) \cdot \log_2 \frac{1}{\Pr(s)}$$

$$= - \sum_{s \in \Omega} \Pr(s) \cdot \log_2 \Pr(s)$$

This is *Shannon entropy*

Entropy

$X = \Bigg\{$  $: 0.5,$  $: 0.5 \Bigg\}$

$H(X) =$

Entropy

$X = \Big\{$ 

$: 0.5,$ 

$: 0.5 \Big\}$

$$
\begin{aligned}
H(X) &= 0.5 \cdot \log_2 \frac{1}{0.5} + 0.5 \cdot \log_2 \frac{1}{0.5} \\
&= 0.5 \cdot 1 + 0.5 \cdot 1 \\
&= 1
\end{aligned}
$$

Entropy

$X = \{$ 

$: 0.9,$ 

$: 0.1 \}$

$$H(X) =$$

Entropy

$X = \Big\{$ 

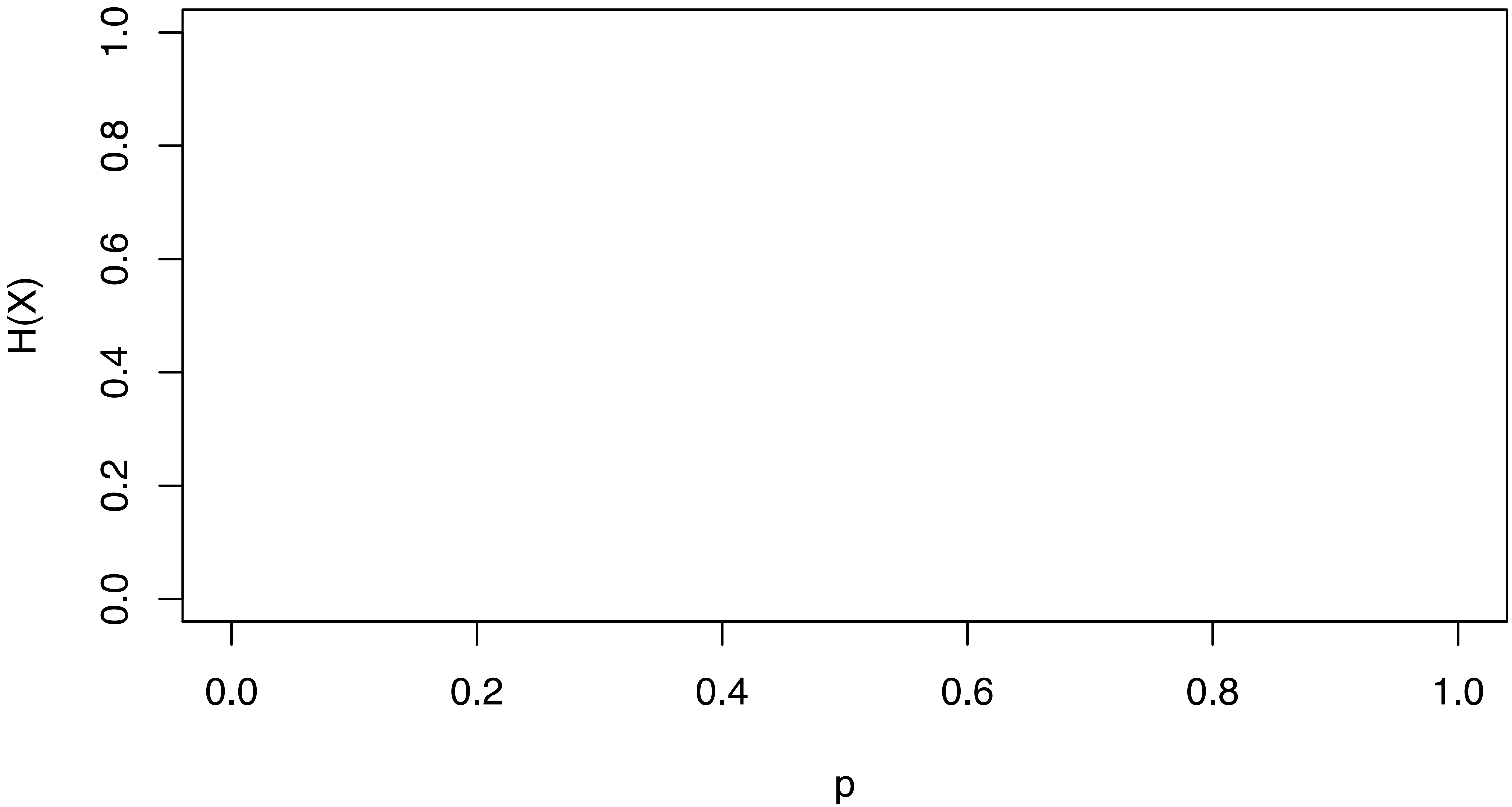
$: 0.9,$ 

$: 0.1 \Big\}$

$$
\begin{aligned}
H(X) &= 0.9 \cdot \log_2 \frac{1}{0.9} + 0.1 \cdot \log_2 \frac{1}{0.1} \\
&= 0.9 \cdot 0.15 + 0.1 \cdot 3.32 \\
&= 0.47
\end{aligned}
$$

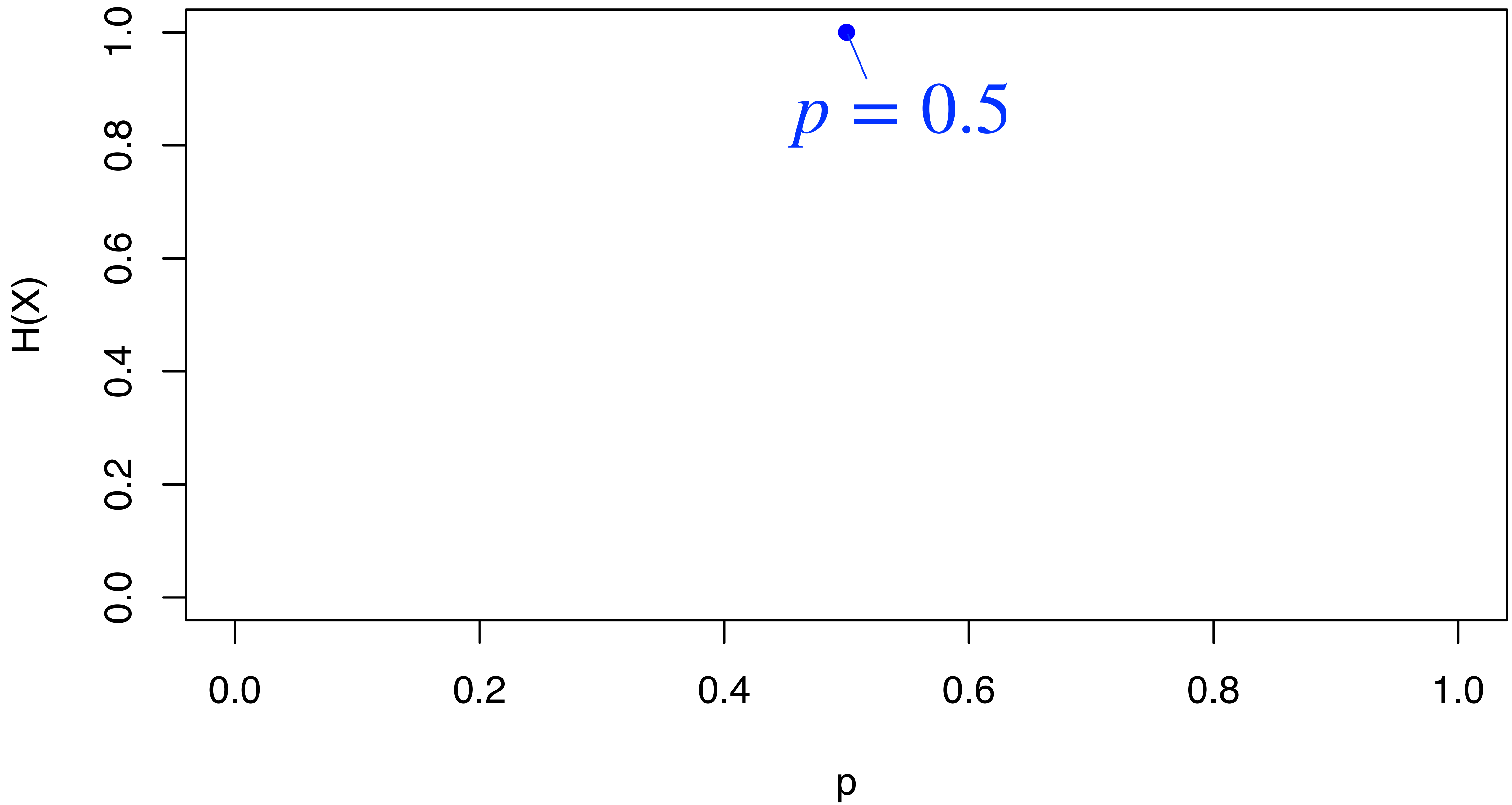
Entropy

$X = \{$  $: p,$  $: 1 - p \}$



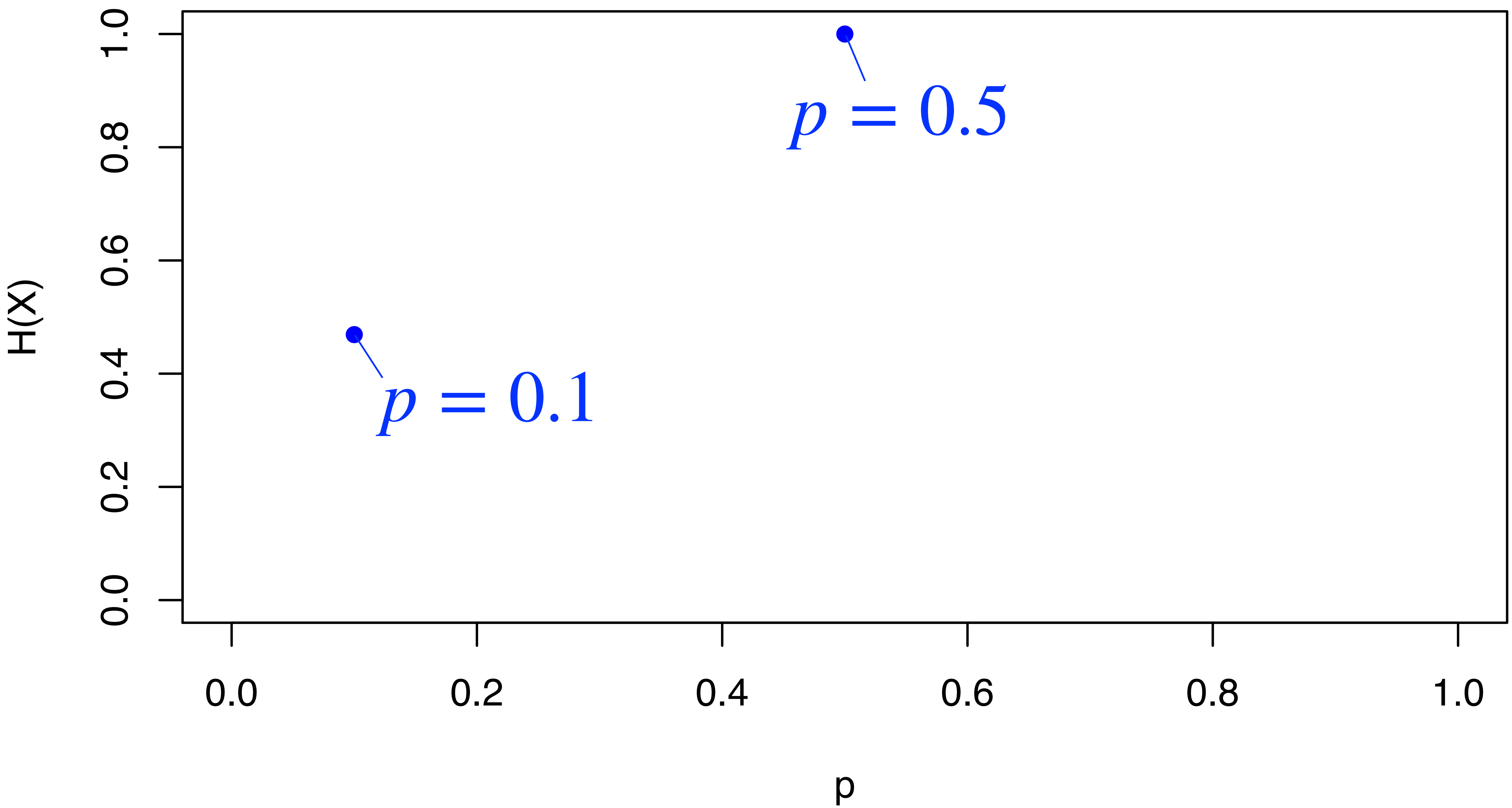
Entropy

$X = \{$  $: p,$  $: 1 - p \}$



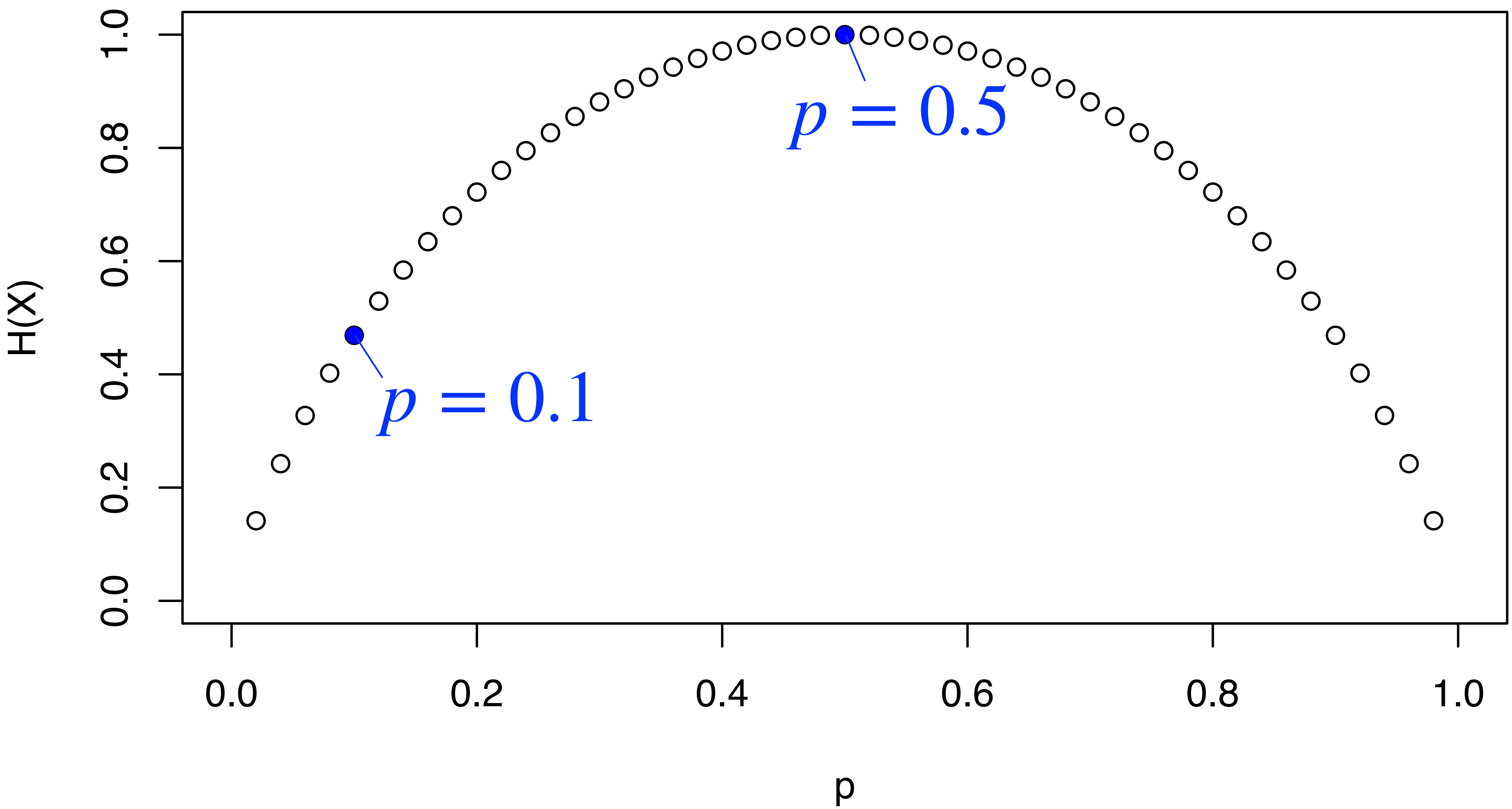
Entropy

$X = \{$  $: p,$  $: 1 - p \}$



Entropy

$X = \{$  $: p,$  $: 1 - p \}$



Entropy

$$X = \left\{ \text{⚀ ⚁ ⚂ ⚃ ⚄ ⚅} : \frac{1}{6} \text{ each} \right\}$$

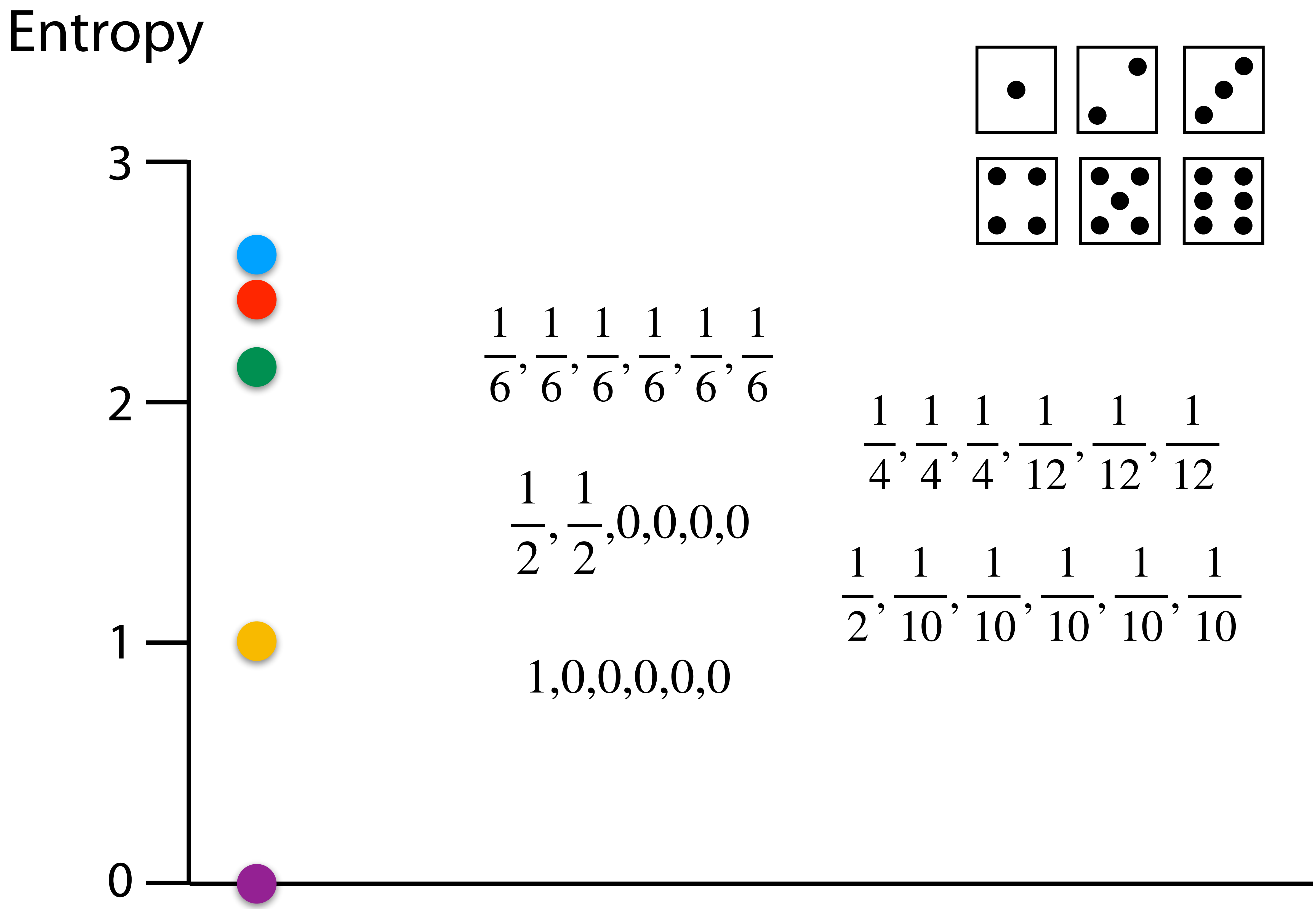
$$H(X) =$$

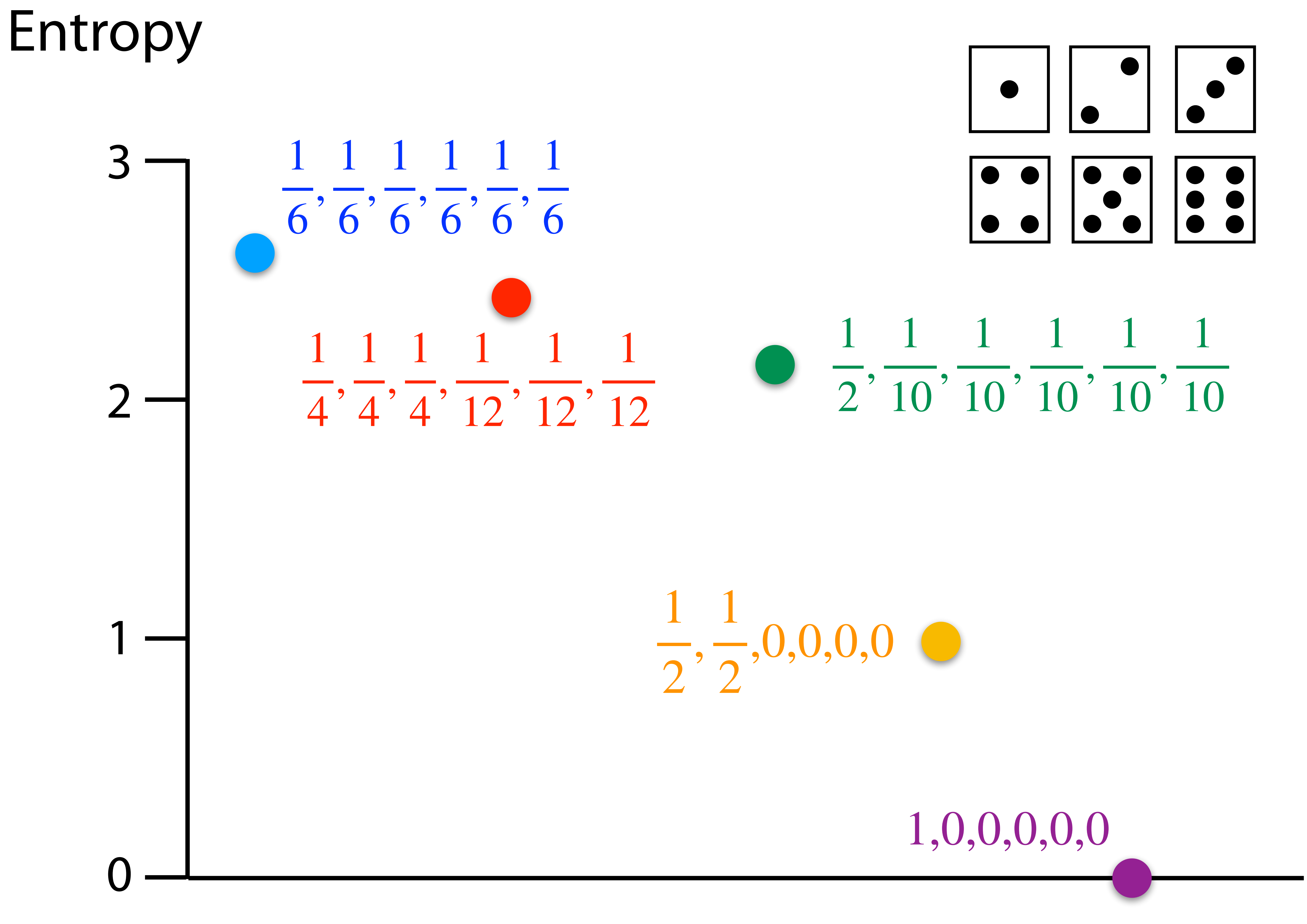
Entropy

$$X = \left\{ \text{⚀ ⚁ ⚂ ⚃ ⚄ ⚅} : \frac{1}{6} \text{ each} \right\}$$

$$H(X) = \sum_{i=1}^{6} \frac{1}{6} \log_2 6$$

$$= \log_2 6 = 2.58$$


Entropy
3
2
1
0
$\frac{1}{6},\frac{1}{6},\frac{1}{6},\frac{1}{6},\frac{1}{6},\frac{1}{6}$
$\frac{1}{4},\frac{1}{4},\frac{1}{4},\frac{1}{12},\frac{1}{12},\frac{1}{12}$
$\frac{1}{2},\frac{1}{2},0,0,0,0$
$\frac{1}{2},\frac{1}{10},\frac{1}{10},\frac{1}{10},\frac{1}{10},\frac{1}{10}$
$1,0,0,0,0,0$



Entropy
3
2
1
0
$\frac{1}{6},\frac{1}{6},\frac{1}{6},\frac{1}{6},\frac{1}{6},\frac{1}{6}$
$\frac{1}{4},\frac{1}{4},\frac{1}{4},\frac{1}{12},\frac{1}{12},\frac{1}{12}$
$\frac{1}{2},\frac{1}{10},\frac{1}{10},\frac{1}{10},\frac{1}{10},\frac{1}{10}$
$\frac{1}{2},\frac{1}{2},0,0,0,0$
1,0,0,0,0,0

Entropy

When outcomes are equally probable:

$$H(X) =$$

Entropy

When outcomes are equally probable:

$$
\begin{aligned}
H(X) &= \sum_{s \in \Omega_X} \Pr(s) \cdot \log_2 \frac{1}{\Pr(s)} \\
&= \sum_{s \in \Omega_X} \frac{1}{|\Omega_X|} \cdot \log_2 |\Omega_X| \\
&= \log_2 |\Omega_X|
\end{aligned}
$$

Matching the definition of worst-case entropy

Entropy

Shannon entropy $H(X)$ is a function of a random variable

The r.v. models a data ***source***; e.g. a person speaking, or letters of a DNA string

Assumes a ***memoryless*** source; each item is an i.i.d. draw

Entropy

So far we've seen

Worst-case entropy $H_{wc}(U)$ is a function of a ***set***

Shannon entropy $H(X)$, a function of a ***random variable***

When outcomes are equiprobable, $H(X) = H_{wc}(\Omega_X)$

Entropy

Say we have a memoryless binary source and an ***example string*** B it emitted

We can count B's 0s & 1s to "train" a model

$$H_0(B) =$$

$$m = \text{\# 1s in } B$$

$$n = |B|$$

Entropy

Say we have a memoryless binary source and an ***example string*** B it emitted

We can count B's 0s & 1s to "train" a model

$$H_0(B) = H\left(X \sim \text{Bern}\left(\frac{m}{n}\right)\right)$$

$m = \#\ \text{1s in } B$

$n = |B|$

Entropy

Say we have a memoryless binary source and an ***example string*** B it emitted

We can count B's 0s & 1s to "train" a model

$$H_0(B) = H\left(X \sim \text{Bern}\left(\frac{m}{n}\right)\right) \qquad \begin{aligned} m &= \#\text{ 1s in } B \\ n &= |B| \end{aligned}$$

$$= \frac{m}{n}\log_2\frac{n}{m} + \frac{n-m}{n}\log_2\frac{n}{n-m}$$

H_0 is the ***empirical zero order entropy***

Entropy

So:

Worst-case entropy $H_{wc}(U)$ is a function of a ***set***

Shannon entropy $H(X)$, a function of a ***random variable***

Empirical zero order entropy $H_0(B)$ of a ***sequence*** B is the Shannon entropy of a memoryless source "trained" to B

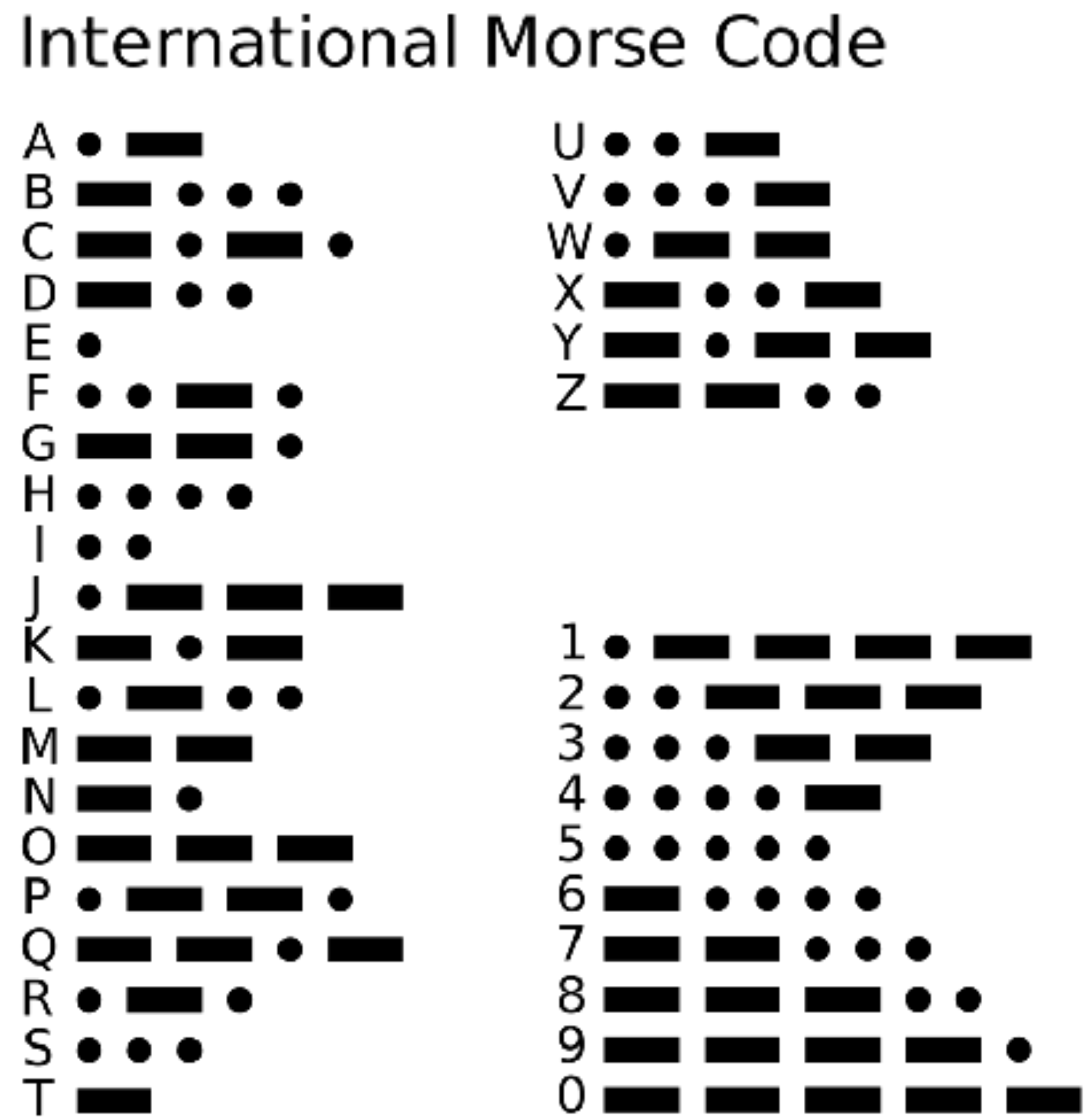
Codes

A good code will:

Minimize average code length (approach H_0)

Give ***unambiguous*** mappings for encoding & decoding

Allow efficient encoding & decoding



Codes

$$H(X) = \sum_{s \in \Omega} \Pr(s) \cdot \log_2 \frac{1}{\Pr(s)}$$

Shannon entropy equation hints at codes of length $\log_2 \dfrac{1}{\Pr(s)}$

Codes

Say we have a source emitting ***symbols*** from ***alphabet*** $\Sigma = \{\texttt{a}, \texttt{c}, \texttt{g}, \texttt{t}\}$

Source is ***memoryless,*** modeled by r.v.:

$$X = \{\ \texttt{a} : \frac{1}{2},\ \texttt{c} : \frac{1}{4},\ \texttt{g} : \frac{1}{8},\ \texttt{t} : \frac{1}{8}\ \}$$

C is a function mapping symbols to binary code sequences. $C : \Sigma \rightarrow \{0,1\}*$

What kind of C do we want?

Codes

$$X = \{\ \mathtt{a} : \frac{1}{2},\ \mathtt{c} : \frac{1}{4},\ \mathtt{g} : \frac{1}{8},\ \mathtt{t} : \frac{1}{8}\ \}$$

<u>Proposal 1</u>

$$C(\mathtt{a}) = \mathtt{0}$$
$$C(\mathtt{c}) = \mathtt{10}$$
$$C(\mathtt{g}) = \mathtt{110}$$
$$C(\mathtt{t}) = \mathtt{111}$$

a a g c

$\downarrow$

Each codeword is unique; i.e. C is injective

Example from Mathematicalmonk videos on info. theory https://youtu.be/9MCxXJn7TPU

Codes

$$X = \{\, \mathtt{a} : \frac{1}{2},\ \mathtt{c} : \frac{1}{4},\ \mathtt{g} : \frac{1}{8},\ \mathtt{t} : \frac{1}{8} \,\}$$

<u>Proposal 1</u>

$C(\mathtt{a}) = 0$

$C(\mathtt{c}) = 10$

$C(\mathtt{g}) = 110$

$C(\mathtt{t}) = 111$

a a g c

$\downarrow$

0 0 1 1 0 1 0

Each codeword is unique; i.e. C is injective

Example from Mathematicalmonk videos on info. theory https://youtu.be/9MCxXJn7TPU

Codes

Can we go recover original string from code?

Proposal 1

$$C(\texttt{a}) = \texttt{0}$$
$$C(\texttt{c}) = \texttt{10}$$
$$C(\texttt{g}) = \texttt{110}$$
$$C(\texttt{t}) = \texttt{111}$$

$\uparrow$?

`1 1 1 0 0 1 0`

Example from Mathematicalmonk videos on info. theory https://youtu.be/9MCxXJn7TPU

Codes

Can we go recover original string from code?

Proposal 1

$$C(\mathtt{a}) = \mathtt{0}$$
$$C(\mathtt{c}) = \mathtt{10}$$
$$C(\mathtt{g}) = \mathtt{110}$$
$$C(\mathtt{t}) = \mathtt{111}$$

t a a c

$\uparrow$ yes

1 1 1 0 0 1 0

Example from Mathematicalmonk videos on info. theory https://youtu.be/9MCxXJn7TPU

Codes

$$X = \{\, \mathtt{a} : \frac{1}{2},\ \mathtt{c} : \frac{1}{4},\ \mathtt{g} : \frac{1}{8},\ \mathtt{t} : \frac{1}{8} \,\}$$

Proposal 2

$$C(\mathtt{a}) = \mathtt{0}$$
$$C(\mathtt{c}) = \mathtt{1}$$
$$C(\mathtt{g}) = \mathtt{01}$$
$$C(\mathtt{t}) = \mathtt{10}$$

a a g c

$$\downarrow$$

Again, C is injective

Example from Mathematicalmonk videos on info. theory https://youtu.be/9MCxXJn7TPU

Codes

$$X = \{\, \mathtt{a} : \frac{1}{2},\ \mathtt{c} : \frac{1}{4},\ \mathtt{g} : \frac{1}{8},\ \mathtt{t} : \frac{1}{8} \,\}$$

<u>Proposal 2</u>

$C(\mathtt{a}) = \mathtt{0}$

$C(\mathtt{c}) = \mathtt{1}$

$C(\mathtt{g}) = \mathtt{01}$

$C(\mathtt{t}) = \mathtt{10}$

a a g c

$\downarrow$

0 0 0 1 1

Again, C is injective

Example from Mathematicalmonk videos on info. theory https://youtu.be/9MCxXJn7TPU

Codes

Can we go recover original string from code?

<u>Proposal 2</u>

$$C(\mathtt{a}) = \mathtt{0}$$
$$C(\mathtt{c}) = \mathtt{1}$$
$$C(\mathtt{g}) = \mathtt{01}$$
$$C(\mathtt{t}) = \mathtt{10}$$

?

`0 0 0 1 1`

Example from Mathematicalmonk videos on info. theory https://youtu.be/9MCxXJn7TPU

Codes

Can we go recover original string from code?

Proposal 2

$$C(\mathtt{a}) = \mathtt{0}$$
$$C(\mathtt{c}) = \mathtt{1}$$
$$C(\mathtt{g}) = \mathtt{01}$$
$$C(\mathtt{t}) = \mathtt{10}$$

no

?

0 0 0 1 1

a a a c ?

a a g ?

Example from Mathematicalmonk videos on info. theory https://youtu.be/9MCxXJn7TPU

Codes

Let C' be the code extended to *sequences*

$$C' : \Sigma^* \to \{0,1\}^*$$

$$C(\mathtt{a}) = \mathtt{0} \qquad C'(\mathtt{a}) = \mathtt{0}$$

$$C(\mathtt{c}) = \mathtt{10} \qquad C'(\mathtt{ag}) = \mathtt{0110}$$

$$C(\mathtt{g}) = \mathtt{110} \qquad C'(\mathtt{tt}) = \mathtt{111111}$$

$$C(\mathtt{t}) = \mathtt{111} \qquad C'(\mathtt{aaaac}) = \mathtt{000010}$$

C' should be injective, giving *unambiguous* code

(C being injective is not enough)

Codes

Consider two codes, both unambiguous

A	**B**
$C(\mathrm{a}) = 1$	$C(\mathrm{a}) = 1$
$C(\mathrm{c}) = 10$	$C(\mathrm{c}) = 01$
$C(\mathrm{g}) = 00$	$C(\mathrm{g}) = 00$

`a a g c` $\downarrow$ `1 1 0 0 1 0`

`a a g c` $\downarrow$ `1 1 0 0 0 1`

Codes

Now we decode:

A

$C(\mathrm{a}) = 1$

$C(\mathrm{c}) = 10$

$C(\mathrm{g}) = 00$

`1 1 0 0 1 0`

Considering first 1, can't yet tell if it's an a or part of a c

Codes

Now we decode:

A

$C(\text{a}) = 1$

$C(\text{c}) = 10$

$C(\text{g}) = 00$

`1 1 0 0 1 0`

Now sure that first 1 is a.

Not sure about second 1.

Codes

Now we decode:

A

$C(\mathrm{a}) = 1$

$C(\mathrm{c}) = 10$

$C(\mathrm{g}) = 00$

$\underbrace{110}\,010$

Either we have

`ac...`

`aag...`

Codes

Now we decode:

A

$C(\mathrm{a}) = 1$

$C(\mathrm{c}) = 10$

$C(\mathrm{g}) = 00$

$\underbrace{\texttt{1100}}\texttt{10}$

Either we have

`acg...`

`aag...`

Codes

Now we decode:

A

$C(\mathrm{a}) = 1$

$C(\mathrm{c}) = 10$

$C(\mathrm{g}) = 00$

$\underbrace{\texttt{11001}}\texttt{0}$

Now we're sure we have:

`aag...`

But could still be `aaga...` or `aagc...`

Codes

Now we decode:

A

$C(\text{a}) = 1$

$C(\text{c}) = 10$

$C(\text{g}) = 00$

`110010`

Now we're sure we have:

aagc

Codes

Consider an example with a longer run of 0s:

A

$C(\mathrm{a}) = 1$

$C(\mathrm{c}) = 10$

$C(\mathrm{g}) = 00$

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

a?

c?

Can't distinguish a from c until we see whether run of `0`s is odd or even

Since it's odd, must be a c: a c g g g c

Codes

Now we decode:

B

$C(\mathrm{a}) = 1$

$C(\mathrm{c}) = 01$

$C(\mathrm{g}) = 00$

1 1 0 0 0 1

Considering first 1, we're immediately sure it's an a

Codes

Now we decode:

B

$C(\text{a}) = 1$

$C(\text{c}) = 01$

$C(\text{g}) = 00$

$$\underbrace{1\,1}\,0\,0\,0\,1$$

Definitely aa

Codes

Now we decode:

B

$C(\mathrm{a}) = \mathtt{1}$

$C(\mathrm{c}) = \mathtt{01}$

$C(\mathrm{g}) = \mathtt{00}$

`110001`

Could be aac or aag

Codes

Now we decode:

B

$C(\mathrm{a}) = \mathtt{1}$

$C(\mathrm{c}) = \mathtt{01}$

$C(\mathrm{g}) = \mathtt{00}$

`1 1 0 0 0 1`

Definitely aag

Codes

Now we decode:

B

$C(\mathrm{a}) = \mathtt{1}$

$C(\mathrm{c}) = \mathtt{01}$

$C(\mathrm{g}) = \mathtt{00}$

$\underbrace{\mathtt{1\,1\,0\,0\,0}}\mathtt{1}$

Could be aagc or aagg

Codes

Now we decode:

B

$C(\mathrm{a}) = 1$

$C(\mathrm{c}) = 01$

$C(\mathrm{g}) = 00$

`1 1 0 0 0 1`

Definitely aagc

Codes

No problems with decoding efficiency here.

B

$$C(\mathrm{a}) = 1$$
$$C(\mathrm{c}) = 01$$
$$C(\mathrm{g}) = 00$$

$$1\,\underbrace{0\,1}_{\mathrm{c}}\,\overbrace{0\,0\,0\,0\,0\,0}^{\mathrm{ggg}}\,1\,0$$

Code is ***prefix-free***; no code is a prefix of another.
Also called a ***prefix code*** for short.

AKA ***instantaneous***

Huffman

Say we start with a string: **`abracadabra`**

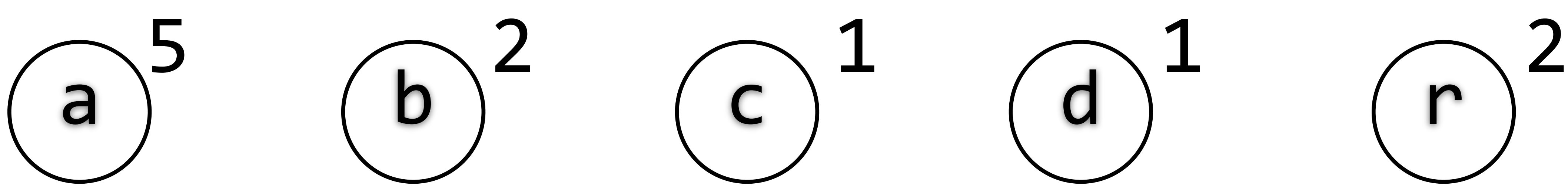
Can compile symbols and their frequencies:

$$\{\, \texttt{a} : 5,\ \texttt{b} : 2,\ \texttt{c} : 1,\ \texttt{d} : 1\ \texttt{r} : 2 \}$$

Huffman

{ `a` : 5, `b` : 2, `c` : 1, `d` : 1 `r` : 2}

In each round, ***join*** the 2 subtrees with lowest total weight

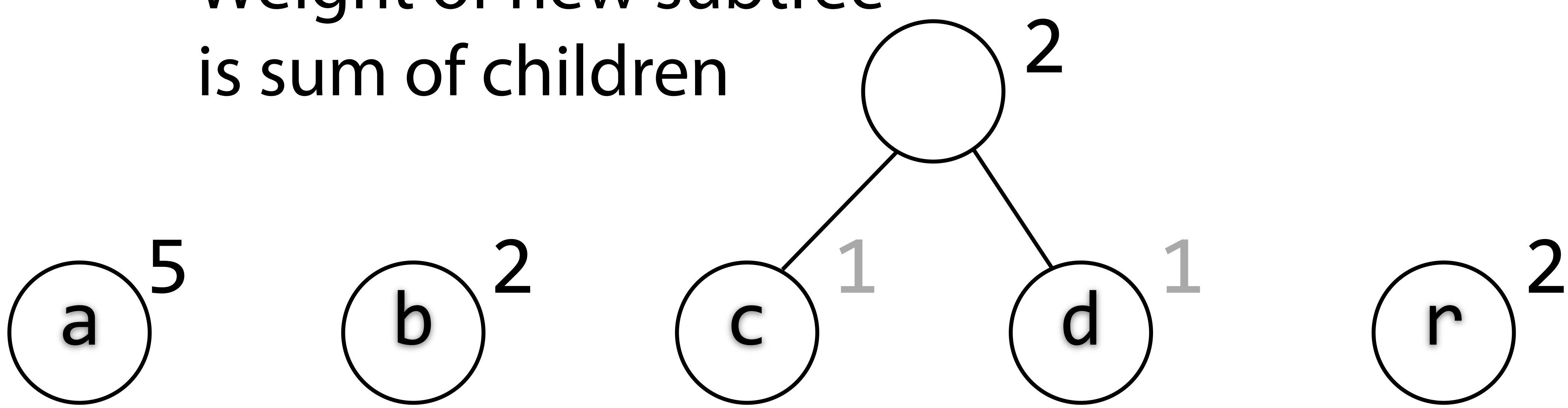


Huffman

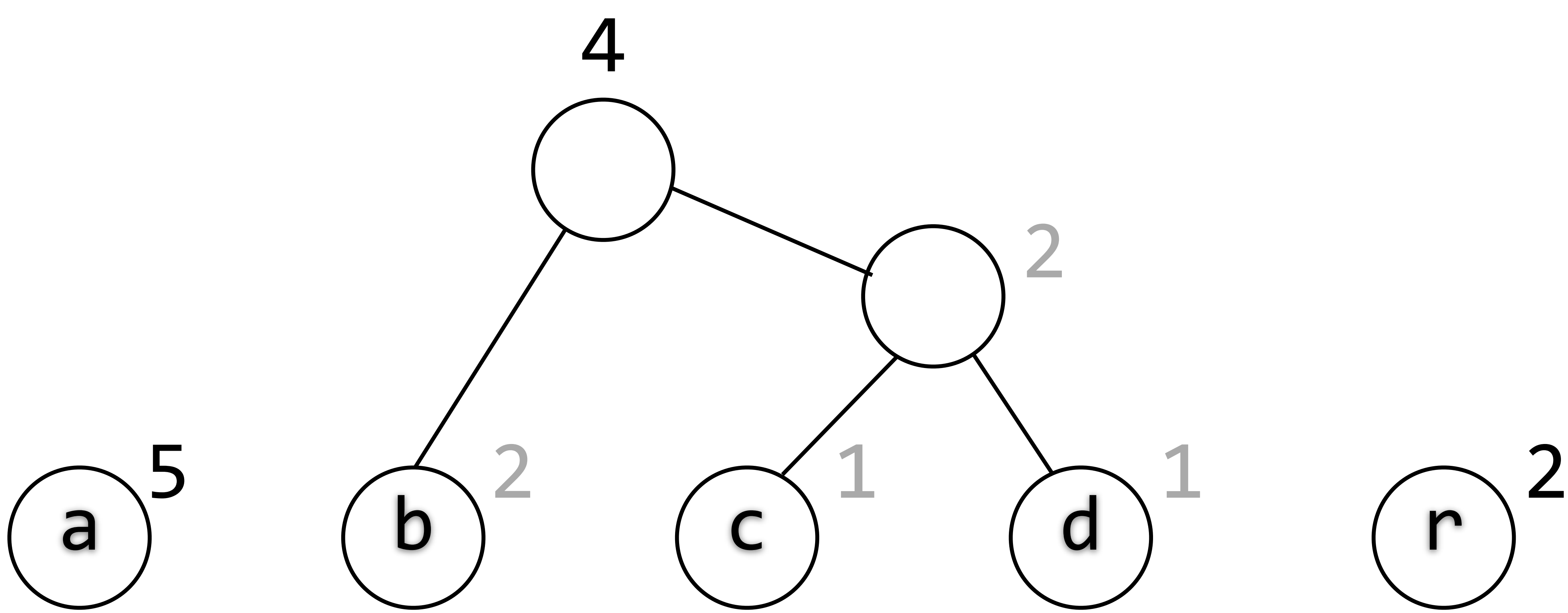
$$\{ \texttt{a} : 5,\ \texttt{b} : 2,\ \texttt{c} : 1,\ \texttt{d} : 1\ \texttt{r} : 2 \}$$

In each round, ***join*** the 2 subtrees with lowest total weight

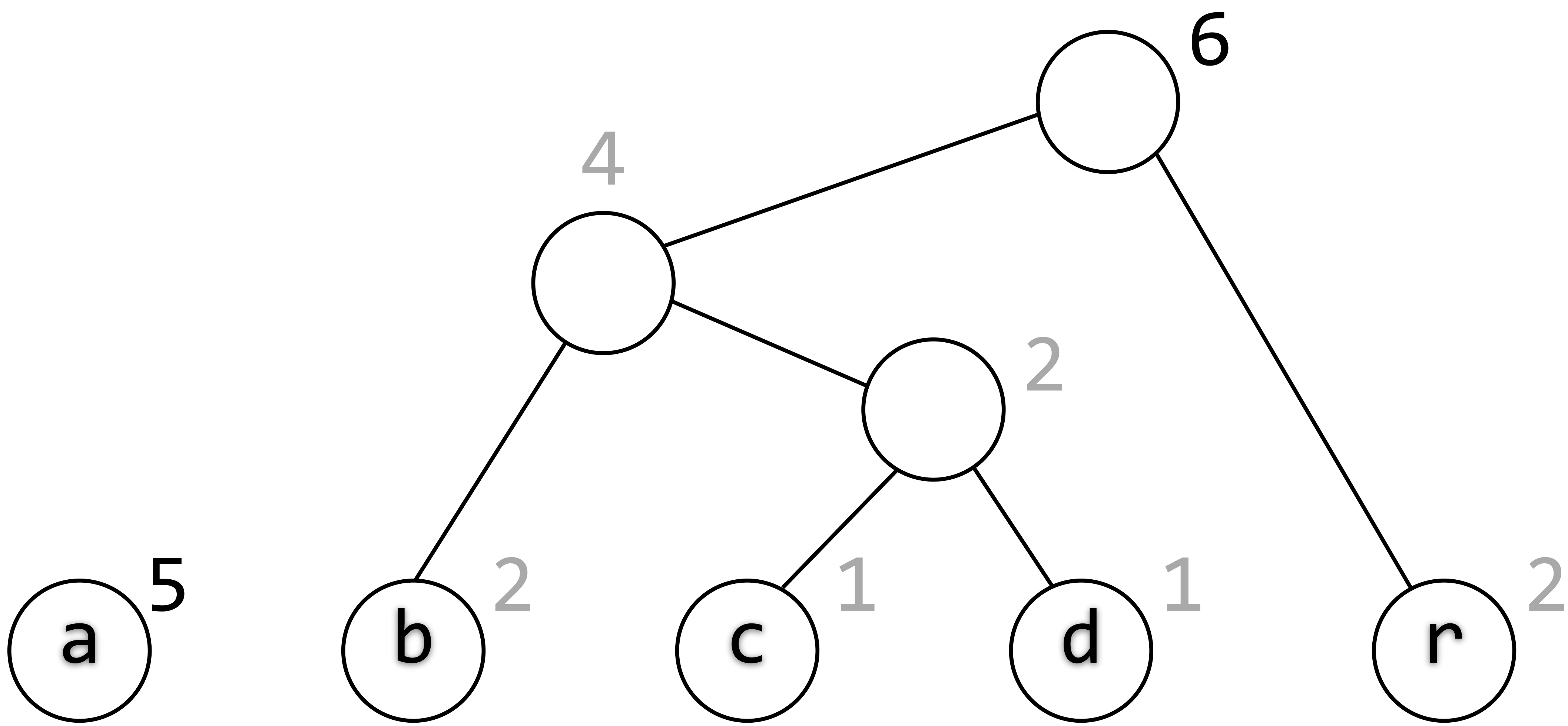
Weight of new subtree
is sum of children



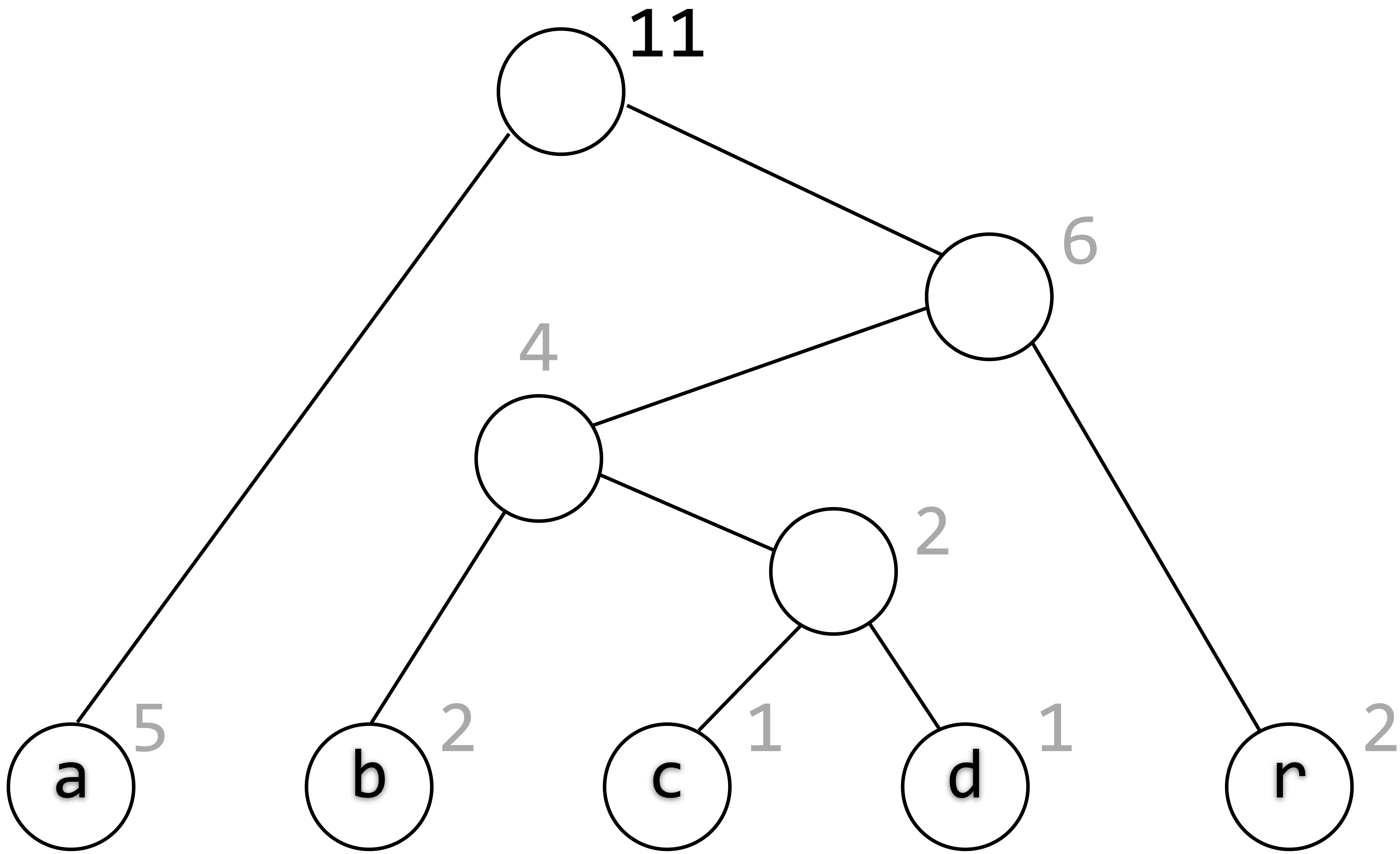
Huffman



Huffman



Huffman



This is the tree but what is the code?

Huffman

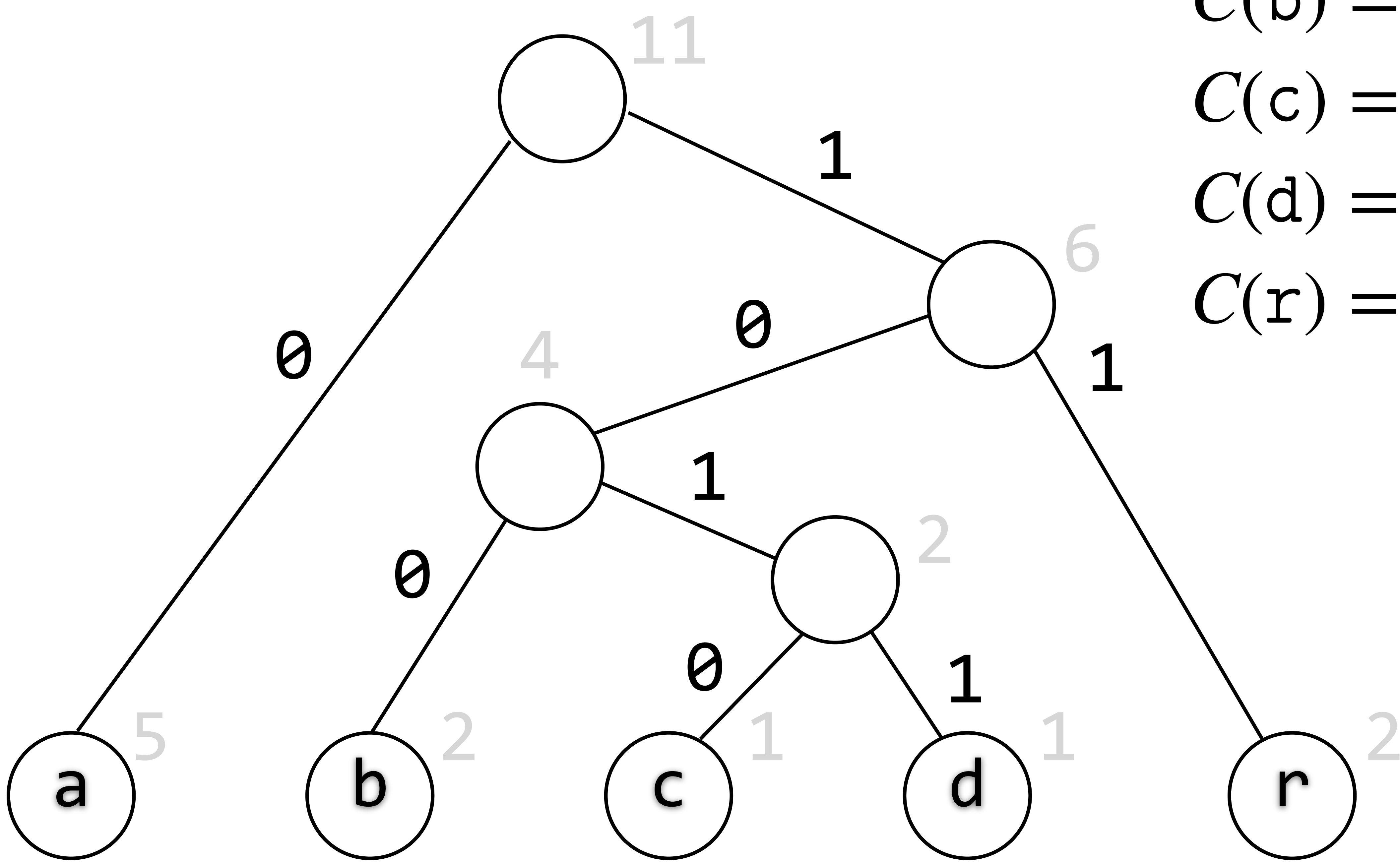
$C(\mathtt{a}) = \mathtt{0}$

$C(\mathtt{b}) = \mathtt{100}$

$C(\mathtt{c}) = \mathtt{1010}$

$C(\mathtt{d}) = \mathtt{1011}$

$C(\mathtt{r}) = \mathtt{11}$



Label edges with 0/1 according to left/right child of parent

Codes equal root-to-leaf concatenations of 0/1's

Huffman

Huffman codes are “optimal" in that each code is at most 1 bit longer then optimal

In other words, for an input string S:

$$|C'(S)| \leq$$

Huffman

Huffman codes are “optimal" in that each code is at most 1 bit longer then optimal

In other words, for an input string S:

$$|C'(S)| \leq |S|(H_0(S) + 1)$$