Huffman Encoding

Learning Goals

- Describe "Binary code," "Prefix free," "Average letter length"
- Explain connection between binary codes + trees
- Describe Huffman's alg. ✓
- Analyze runtime of Huffman's alg ✓
- Describe impact of data structures alg runtime ✓
- Prove correctness of Huffman's alg

Announcements

Quiz 4 Cancelled,

Feedback: Assessment Feedback

Lots of Assignments, OH

Exit Tickets:

Average picture, manipulate sample space

Wikipedia formal languages - alphabet symbol

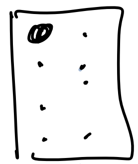
F sigma->{0,1}^*

Binary code use?

Binary Codes

ex: $\Sigma = \{a, b, c, d, \dots, z\}$

<u>def</u>: Given an alphabet Σ, a binary code is a function $f: \Sigma \to \{0,1\}^*$

<u>ex</u>: Braille  10101011, ASCII, Morse Code

• — — • —

Suppose you have a message where the letter "a" occurs 50% of the time, "b" 30%, and "c" 20%. Which is the best binary encoding of $\Sigma = \{a, b, c\}$?

A): $f(a) = 00$ B) $f(a) = 0$ C) $f(a) = 0$

$f(b) = 01$ $f(b) = 1$ $f(b) = 10$

$f(c) = 10$ $f(c) = 01$ $f(c) = 11$

A): $f(a) = 00$
$f(b) = 01$
$f(c) = 10$

$\downarrow$

Doesn't take advantage of differing occurance rates

B) $f(a) = 0$
$f(b) = 1$
$f(c) = 01$

$\downarrow$

Ambiguous for decoding

$01 \rightarrow c$?
$01 \rightarrow ab$?

$\boxed{C)}$ $f(a) = 0$
$f(b) = 10$
$f(c) = 11$

$\downarrow$

Not ambiguous, average bits per letter is small

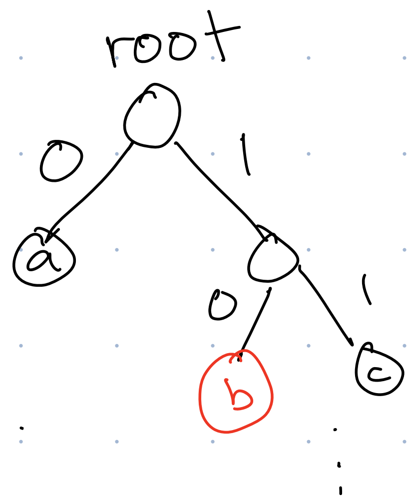
\# of bits in $f(i)$

Average letter length: $L(f) = \sum_{i \in \Sigma} |f(i)| \cdot p(i)$

ex: $\boxed{C}$ $|f(a)|p(a) + |f(b)| \cdot p(b) + |f(c)| \cdot p(c)$

$= 1 \cdot 0.5 + 2 \cdot 0.3 + 2 \cdot 0.2 = 1.5$

Binary Trees & Binary Codes



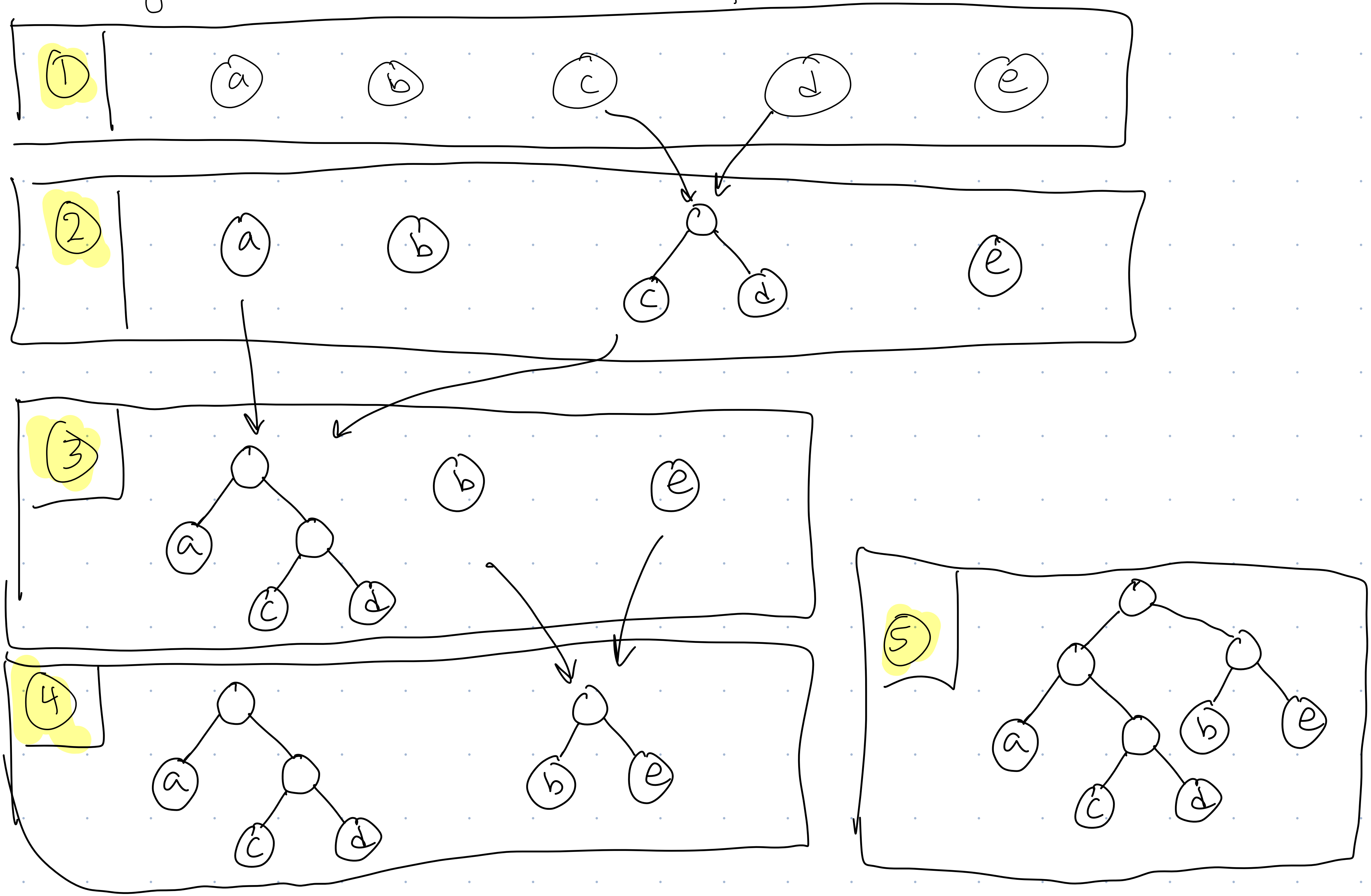
$f(a) = 0$	$f(a) = 0$
$f(b) = 01$	$f(b) = 10$
$f(c) = 11$	$f(c) = 11$

There is ambiguity for decoding if multiple letters lie on same path

$$\rightarrow \underbrace{0}_{a}\underbrace{10}_{b}\underbrace{11}_{c}\underbrace{0}_{a}1\ldots$$

<u>def</u>: A code is "prefix free" if all letters are at leaves in corresponding binary tree.

Merge Trees to Create Prefix Free Codes



Optimal Binary Encoding Problem

Input: Σ (alphabets of symbols)

$p : \Sigma \to \mathbb{R}$ (probabilities / frequency for each symbol)

Output: $f : \Sigma \to \{0,1\}^*$ s.t.

- f is prefix free
- minimize average letter length

Huffman's Algorithm

For each $i \in \Sigma$:

- Create a tree with one node, label i
- Give tree weight $p(i)$

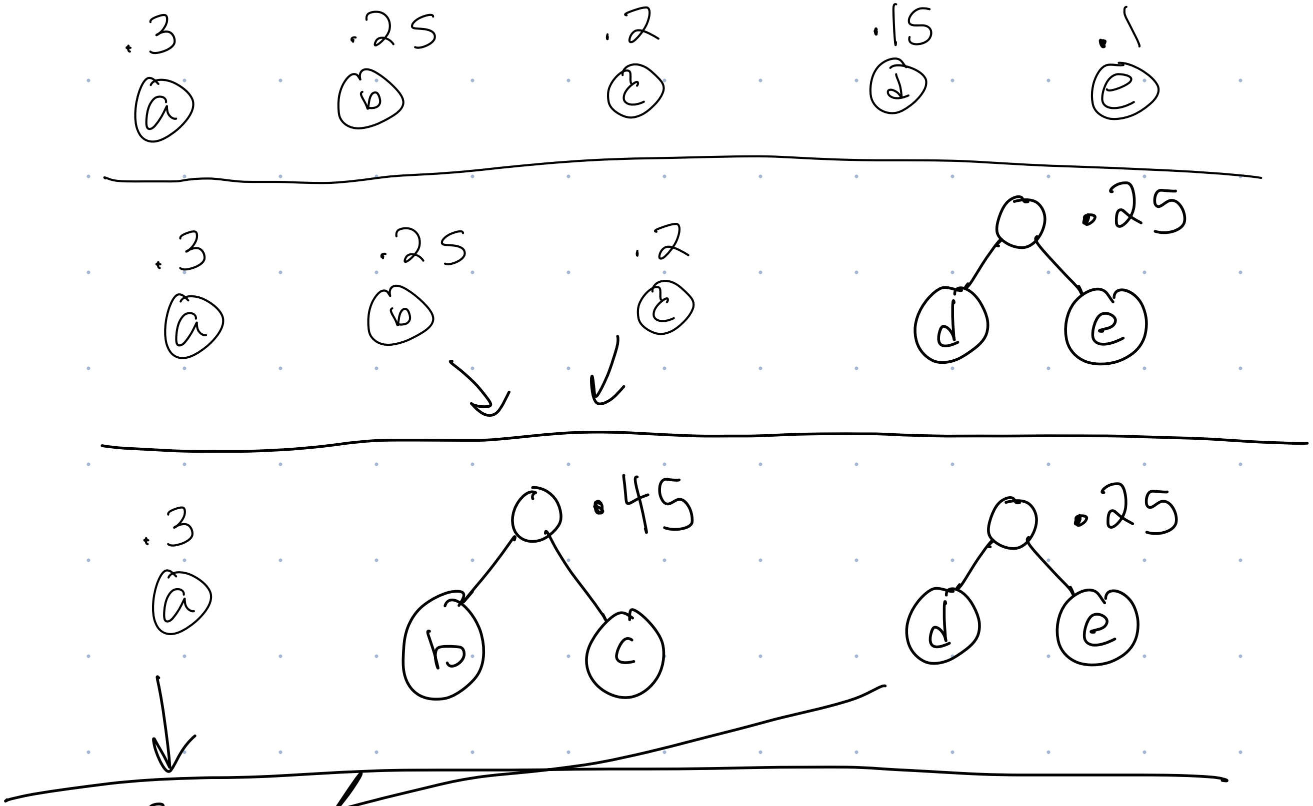
While there is more than one tree:

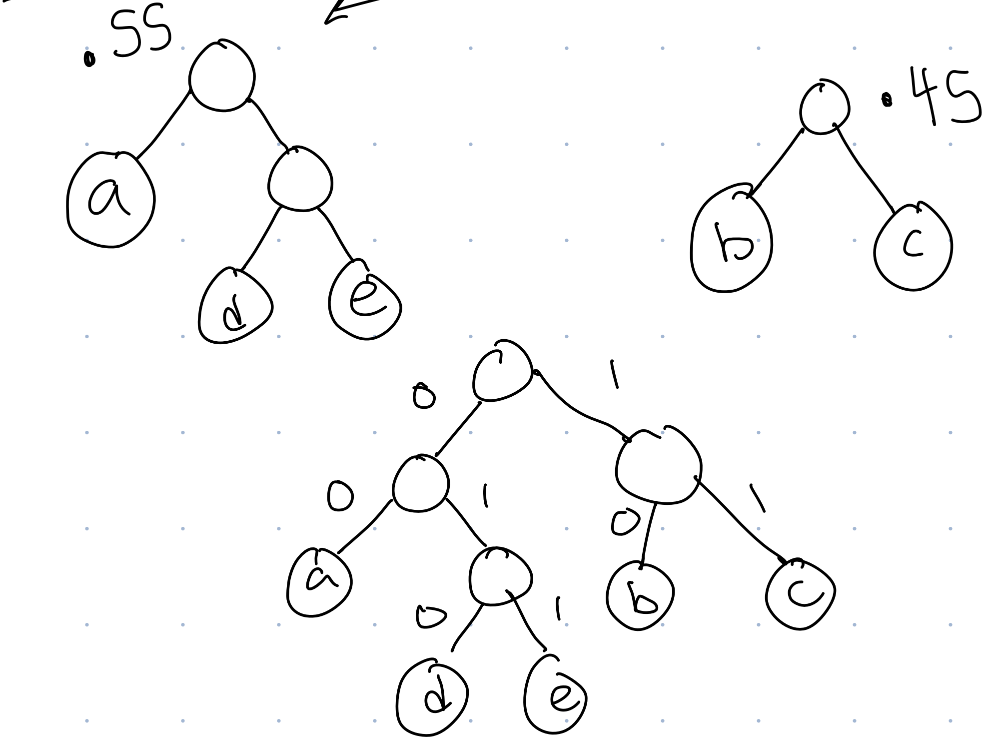
- Merge two trees with smallest weight
- Set weight of merged tree to be sum of weights.

---

i	$p(i)$
a	.3
b	.25
c	.2
d	.15
e	.1

- Use Huffman's algorithm to create a binary code
- What is the average letter length of your code
- What is the runtime of Huffman's in terms of $|\Sigma| = n$? Ideas to improve?
- Why greedy?





a b c d e

$$L = .3 \cdot 2 + .25 \cdot 2 + .2 \cdot 2 + .15 \cdot 3 + .1 \cdot 3$$

$$= 2.25$$

010

Huffman's Algorithm

$|\Sigma| = n$ $\rightarrow O(n^2)$

$O(n)$ For each $i \in \Sigma$:

- Create a tree with one node, label i $O(1)$
- Give tree weight $p(i)$ $O(1)$

$O(n)$ While there is more than one tree:

- Merge two trees with smallest weights $O(n)$
- Set weight of merged tree to be sum of weights $O(1)$

Why greedy:

Huffman's Algorithm $O(n\log n)$

For each $i \in \Sigma$: $O(n)$

- Create a tree with one node, label i
- Give tree weight $p(i)$

Initialize heap with our n-trees ← $O(n\log n)$

While there is more than one tree: $O(n)$

- Merge two trees with smallest weights $O(\log n)$
- Set weight of merged tree to be sum of weights $O(1)$
- Reinsert merged tree into heap $O(\log n)$

Improve runtime?

- At each iteration of while loop, find minimum value tree.
- Helpful data structure?? $\boxed{\text{Min Heap}}$

$O(n\log n)$ • Initialize n items in heap

$O(\log n)$ • Remove minimum wt item

$O(\log n)$ • Insert new item

Go Program! (Lots of details to figure out!)

Ethical Matrix (see programming assignment)

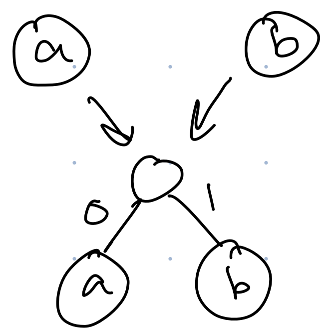
tree merges

Thm: Huffman's Algorithm produces a $\boxed{\text{prefix free code}}$ that minimizes average letter length.

Pf: We will prove correctness by induction on $n = |\Sigma|$.

Base Case: If $n=2$, there are 2 characters: a, b.

Huffman does



This has optimal average letter length. (1)

<u>Inductive Step</u>: Assume for induction that Huffmans alg produces a prefix free code with optimal average letter length for any alphabet with k characters. Consider an input Σ s.t. $|\Sigma| = k+1$.

Let a, b be the characters of Σ with the smallest p-values. Define $\Sigma^- = \Sigma - \{a, b\} \cup \{a/b\}$ where a/b is a new character with $p(a/b) = p(a) + p(b)$.

ex:

$\Sigma = \{e, f, g, h\}$ $p(e) = .1$ $p(f) = .7$ $p(g) = .15$ $p(h) = .05$

$\Sigma^- = \{e/h, f, g\}$ $p(e/h) = .15$ $p(f) = .7$ $p(g) = .15$

ex:

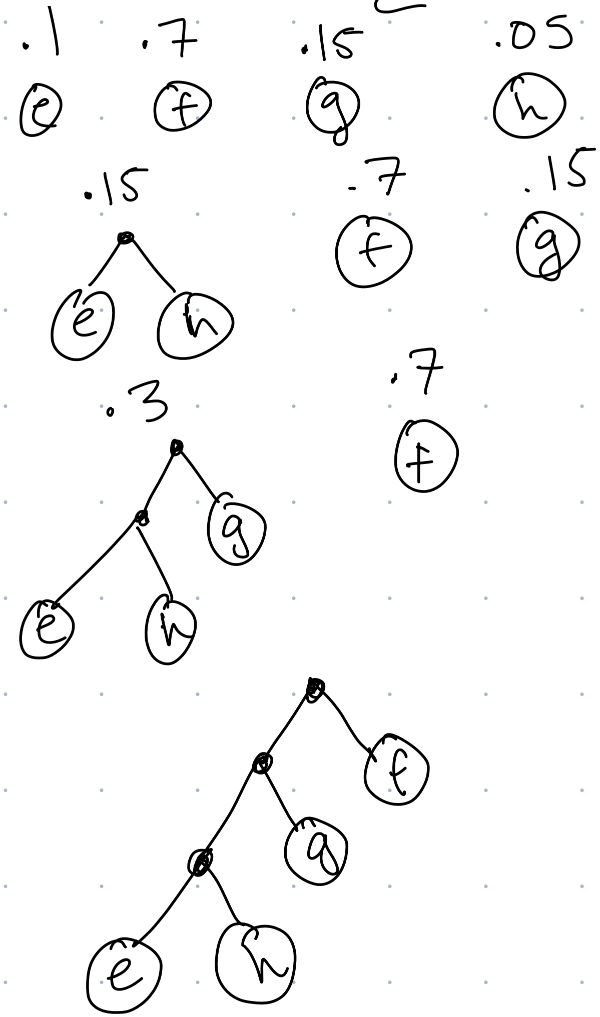
$\Sigma = \{e, f, g, h\}$ $\quad p(e) = .1$ $\quad p(f) = .7$ $\quad p(g) = .15$ $\quad p(h) = .05$

then

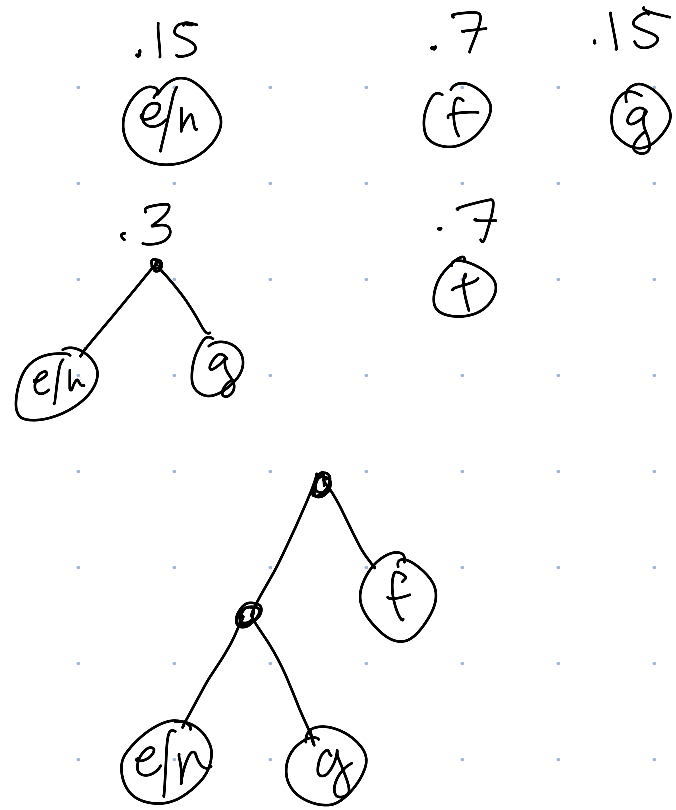
$\bar{\Sigma} =$

Huffman

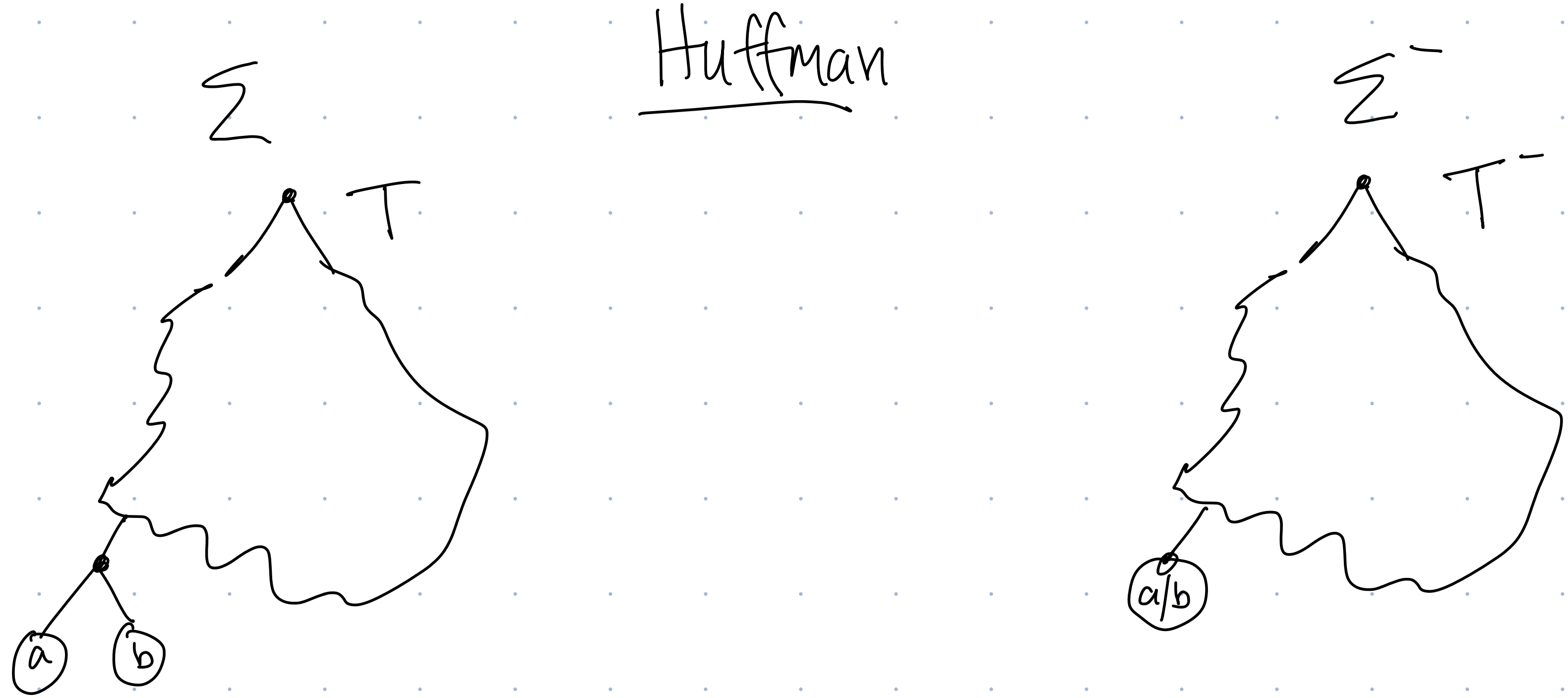
Σ



$\bar{\Sigma}$



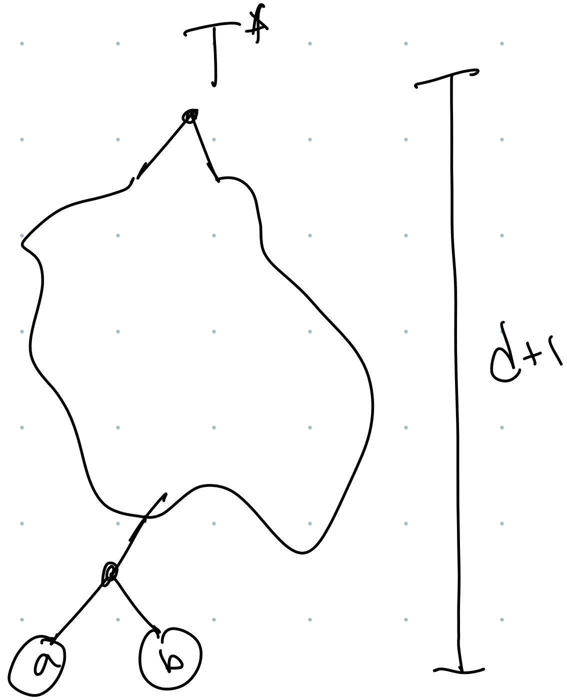
In general:



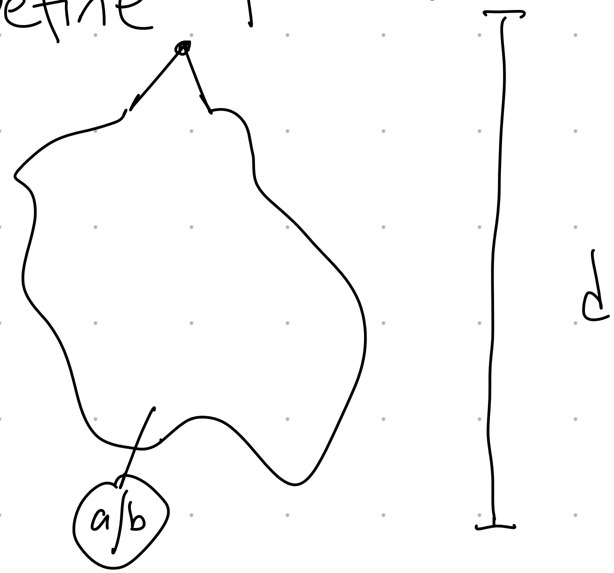
By inductive assumption, T^- gives an optimal binary code (b/c has k characters, created using Huffman)

Lemma: There is an optimal tree for Σ where a, b are siblings. (will prove later)

Suppose for contradiction that T is not optimal. Let $T^* \neq T$ be an optimal tree with a, b siblings (by Lemma)



Define T^{*-} on Σ^-



(fill in with d, $p(a)$, $p(b)$)

Then

$$L(T^*) = \sum_{i \neq a,b \in \Sigma} p(i)\, d(i) + \boxed{\quad}$$

$$L(T^{*-}) = \sum_{i \neq a/b} p(i)\, d(i) + \boxed{\quad}$$

So

$$L(T^*) - L(T^{*-}) =$$

Similarly

$$L(T) - L(T^-) =$$

Thus

$$L(T^*) - L(T^{*-}) = L(T) - L(T^-)$$

Rearranging: $L(T) - L(T^*) =$

This is a contradiction because

Lemma: There is an optimal tree for Σ with a, b (characters with smallest p-values) siblings.