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 ✓

Do we use induction to prove correctness of greedy algorithms?

A) Never

B) Sometimes

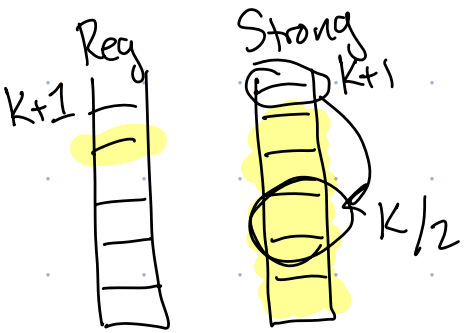
C) Always

What is true about strong + regular induction

A) They are equivalent, the choice is stylistic

B) You can always use strong induction

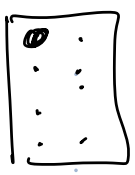
C) Regular induction can be used to prove strong induction.



Binary Codes

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

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

ex: Braille  01010111, 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): $f(a) = 00$, $f(b) = 01$, $f(c) = 10$

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

C) $f(a) = 0$, $f(b) = 10$, $f(c) = 11$

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

↓

Doesn't take advantage of differing occurance rates

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

↓

Ambiguous for decoding

$01 \rightarrow c$?

$01 \rightarrow ab$?

C) $f(a)=0$, $f(b)=10$, $f(c)=11$

↓

Not ambiguous, average bits per letter is small

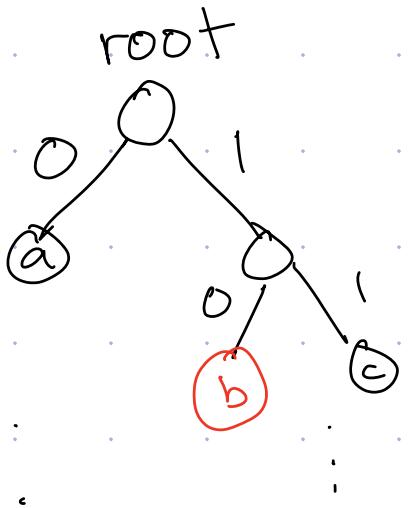
$|f(i)|$ ← # of bits in $f(i)$

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

ex: 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



$f(a) = 0$ $\quad f(a) = 0$

$f(b) = 01$ $\quad f(b) = 10$ ⟷

$f(c) = 11$ $\quad f(c) = 11$

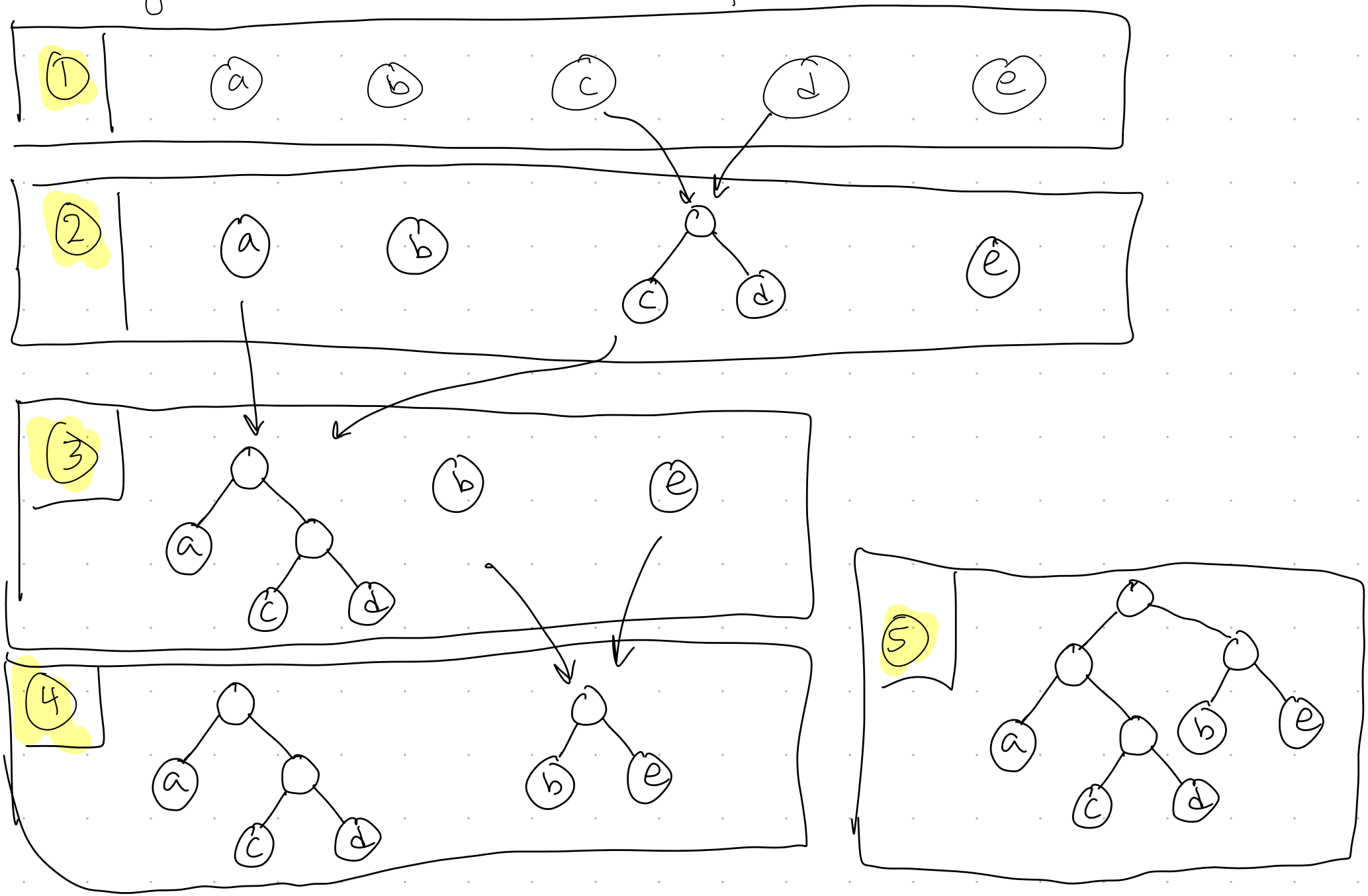
There is ambiguity for decoding if multiple letters lie on same path

→ 0 1 0 1 1 0 1 ...

a b c a

<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

$$L = \sum_{i \in \Sigma} p(i) \, |f(i)|$$

Objective function

Huffman's Algorithm

For each $i \in \Sigma$:

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

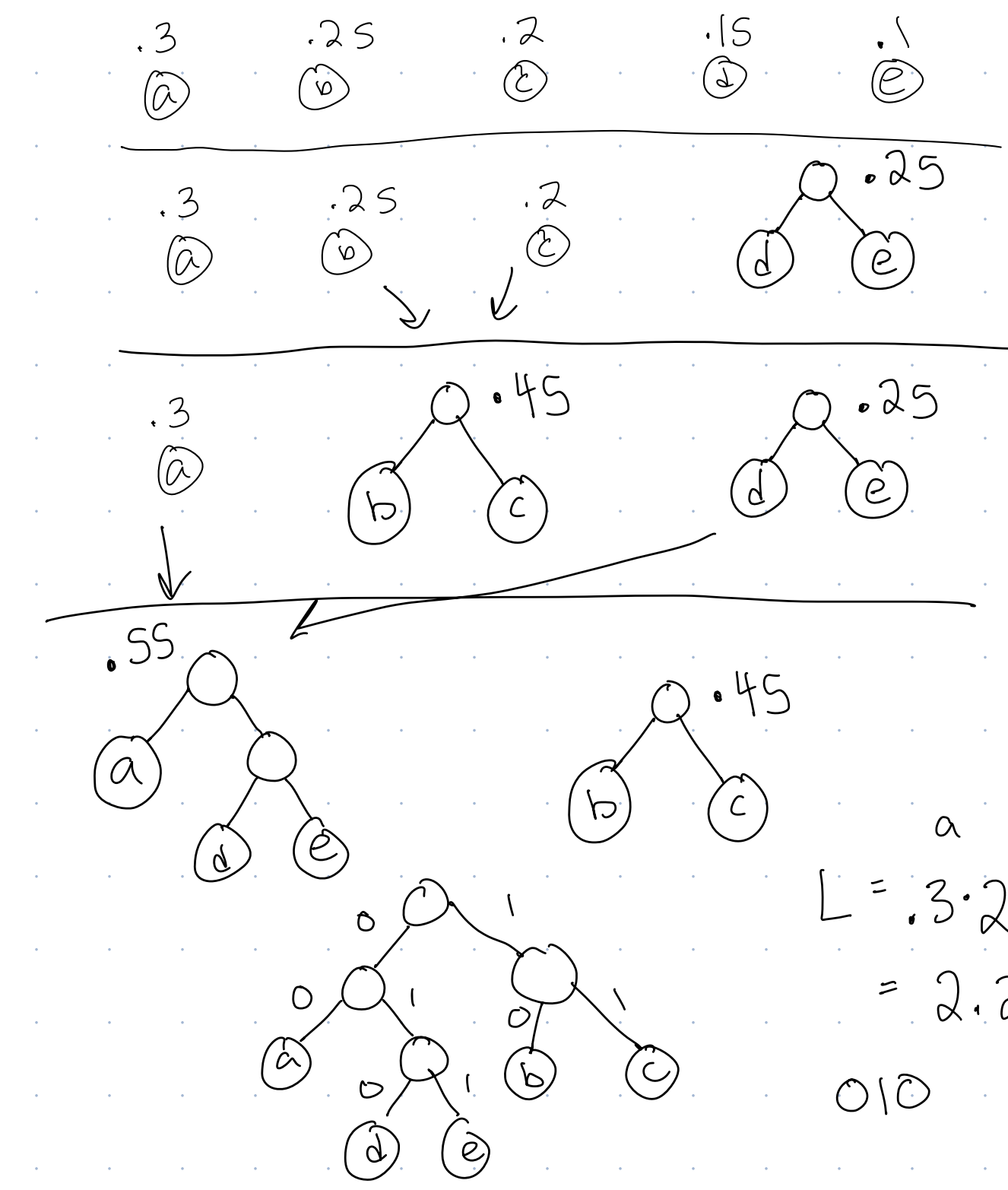
While there is more than one tree:

- 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

$|\Sigma| = n$ → $O(n^2)$

$O(n)$ 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(nlogn)

For each $i \in \Sigma$:
- Create a tree with one node, label i
- Give tree weight $p(i)$

} O(n)

Initialize heap with our n-trees ← O(nlogn)

While there is more than one tree: O(n)
- Merge two trees with smallest weights O(logn)
- Set weight of merged tree to be sum of weights O(1)
- Reinsert merged tree into heap O(logn)

Improve runtime?

- At each iteration of while loop, find minimum value tree.
- Helpful data structure?? **Min Heap**

O(nlogn) • Initialize n items in heap
O(logn) • Remove minimum wt item
O(logn) • Insert new item

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

Ethical Matrix (see programming assignment)

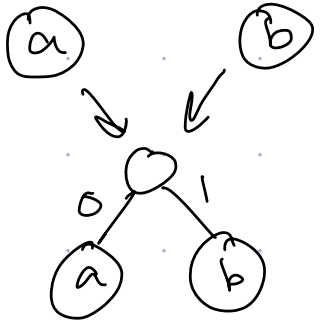
tree merges

Thm: Huffman's Algorithm produces a 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)

Inductive Step: 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:

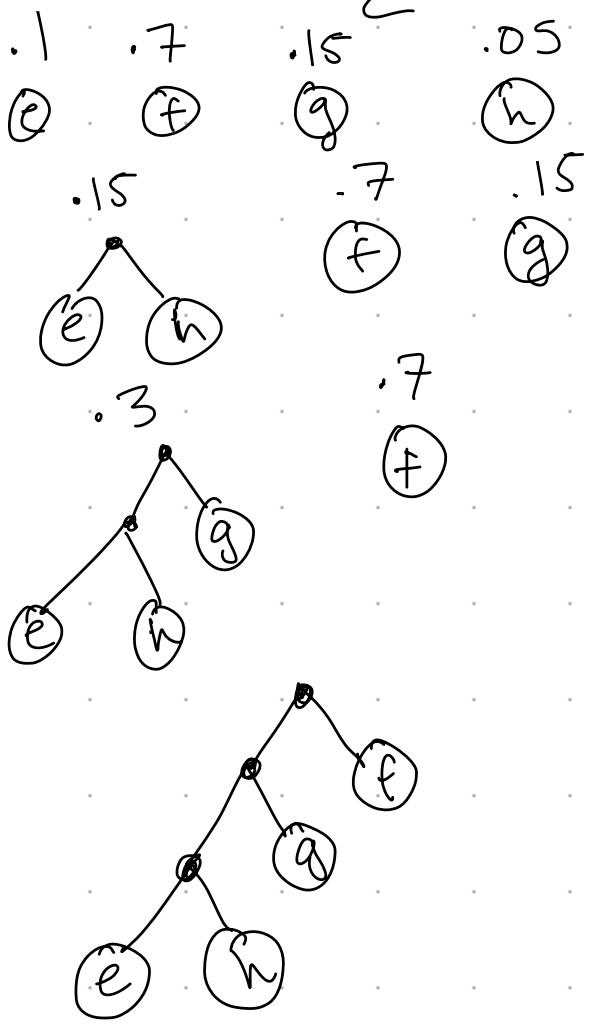
$\Sigma = \{e, f, g, h\}$ $p(e) = .1$ $p(f) = .7$ $p(g) = .15$ $p(h) = .05$

then

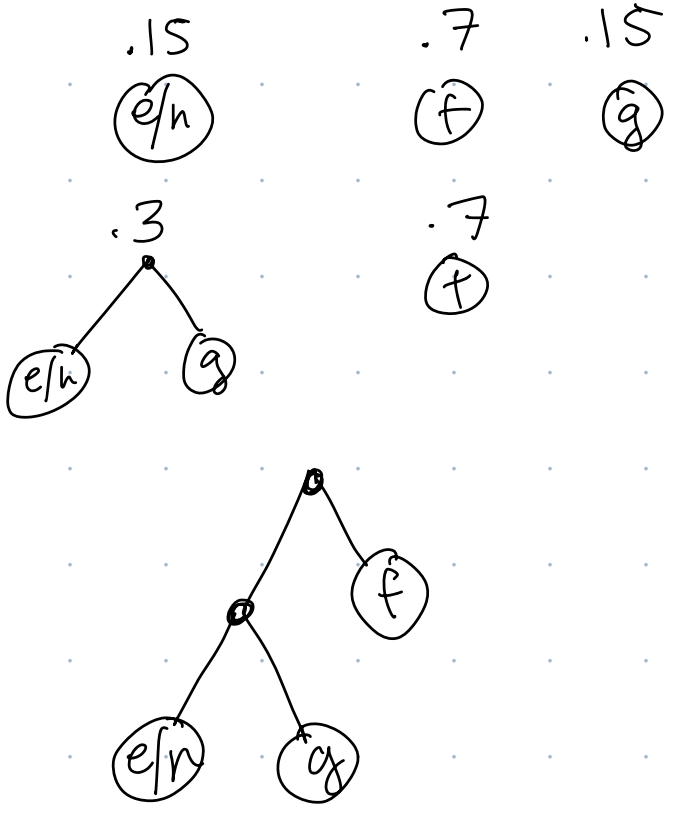
$\Sigma^- =$

Huffman

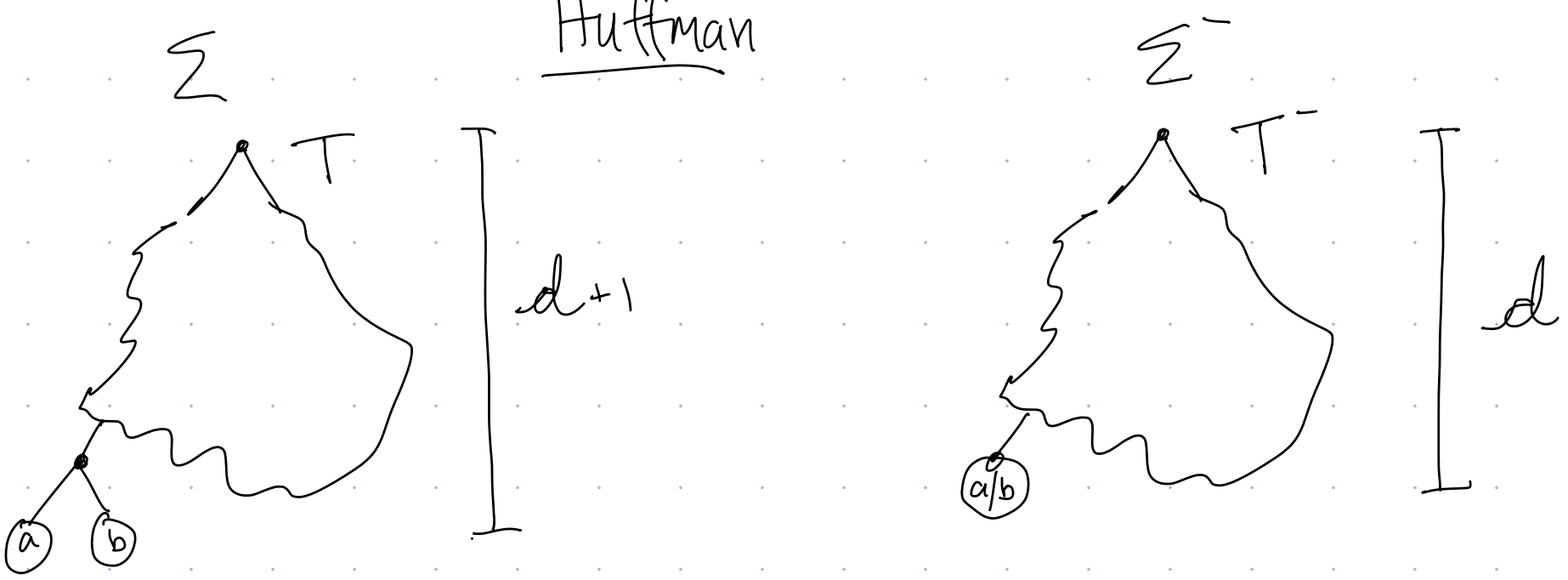
Σ



Σ^-



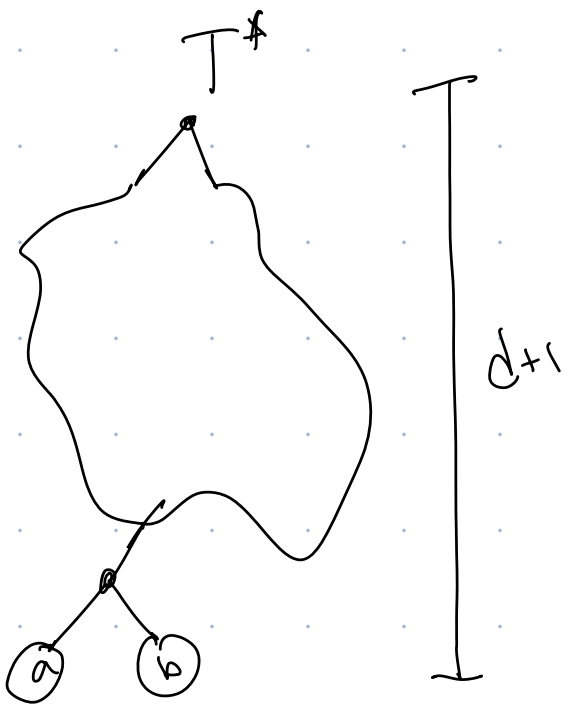
In general:



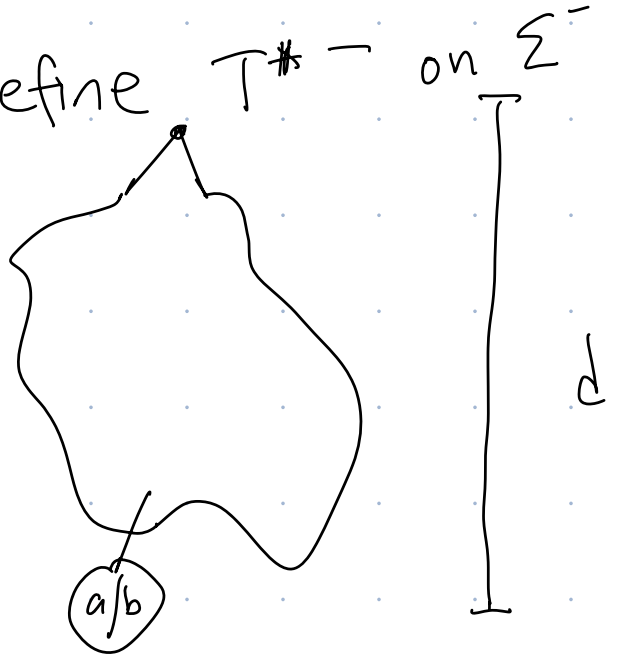
By inductive assumption, T^- gives an optimal binary code (b/c has k characters, created using Huffman)

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

$$L(T^*) = \sum_{i \neq a,b \in \Sigma} p(i)\,d(i) + \boxed{p(a)(d+1) + p(b)(d+1)}$$

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

$$\overset{\text{''}}{(p(a) + p(b))}\,d$$

So $L(T^*) - L(T^{*-}) =$ $p(a) + p(b)$

Similarly

$L(T) - L(T^-) =$ $p(a) + p(b)$

Thus

$+L(T^-)\quad L(T^*) - L(T^{*-}) = L(T) - L(T^-) \quad -L(T^*)$

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

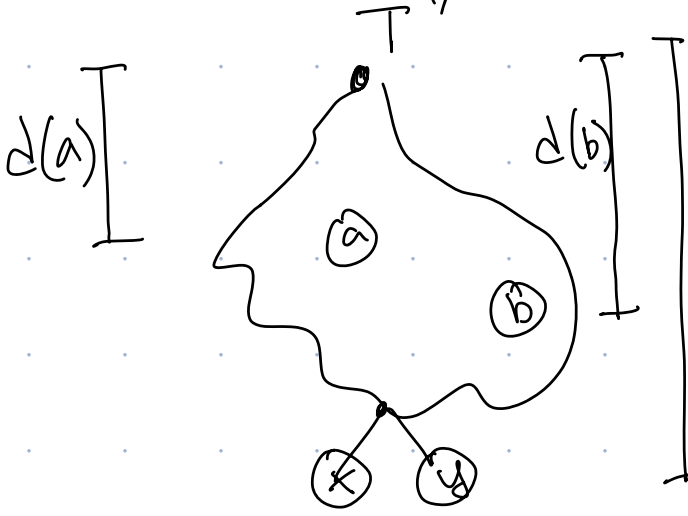
This is a contradiction because

Left side is positive because T^* by our assumption, is a tree with optimal average letter length, while T, our Huffman alg, does not. Right hand side is less than or equal to 0, because T^- by inductive assumption has minimal av. letter length, and T^{*-}

is any other tree on Σ. $\Rightarrow$ Contradiction. Thus Huffmans alg works correctly for any alphabet with k+1 letters, so we have proved the inductive step.

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

Let T' be any optimal prefix free tree where a, b are not siblings. Let x, y be siblings in T' at max depth.



If we exchange $x \leftrightarrow a$, and $y \leftrightarrow b$, the new tree T'_{ex} will have the same or smaller average letter length as T' because...