S. KIMMEL

Goals

- Describe why Huffman's algorithm is correct
- Analyze runtime of Huffman's alg.
- Describe relationship between algorithm + data structure

Midterm

- Up to PS 8 (no huffman, no shortest path)
- Post to Canvas Discussion tonight to influence Thurs. review
- Same system as first midterm, 1 handwritten cheat sheet (1 side), etc.

Huffman's Algorithm

Initialize each $i \in \Sigma$ as tree with associated probability

While (> 1 tree to be merged)

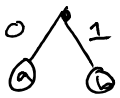
- Find 2 trees with smallest probability
- Merge into new tree with new probability = sum of old probabilities

S. KIMMEL

Thm: Huffman's Algorithm Produces a Tree T with minimum average length

$$L(T) = \sum_{i \in \Sigma} p_i [\text{depth node } i]$$

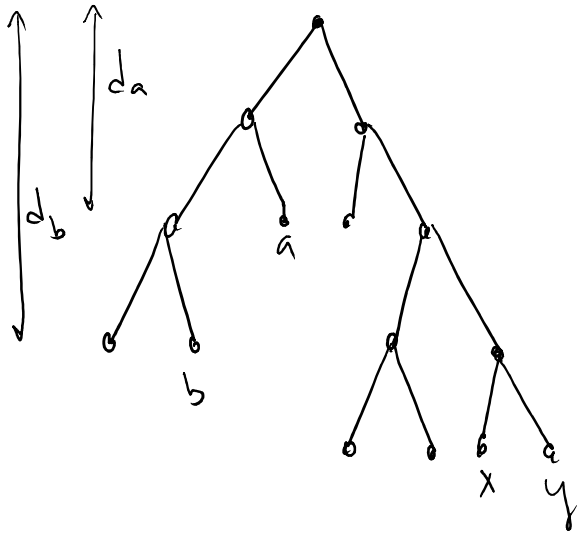
Pf: Induction on $n = |\Sigma|$. Assume $n \geq 2$

Base Case: If $n = 2$, 

is optimal

Inductive Step: Let a, b be the letters with the lowest frequency.

1. There is a tree with optimal L s.t. a, b are siblings. [Use exchange!] For contradiction, suppose T^* is optimal tree.

T^*: 

Let x, y be siblings at deepest level of tree

Let T be tree where $x \leftrightarrow a$
$y \leftrightarrow b$

S.KIMMEL

Q What is $L(T^*) - L(T)$?

$$L(T^*) = \sum_{i \in \Sigma - \{a,b,x,y\}} p_i d_i + p_a d_a + p_b d_b + (p_x + p_y) d_{x/y}$$

$$L(T) = \sum_{i \in \Sigma - \{a,b,x,y\}} p_i d_i + p_a d_{x/y} + p_b d_{x/y} + p_x d_a + p_y d_b$$

$$L(T^*) - L(T) = p_x(d_{x/y} - d_a) + p_y(d_{x/y} - d_b) + p_a(d_a - d_{x/y}) + p_b(d_b - p_{x/y})$$

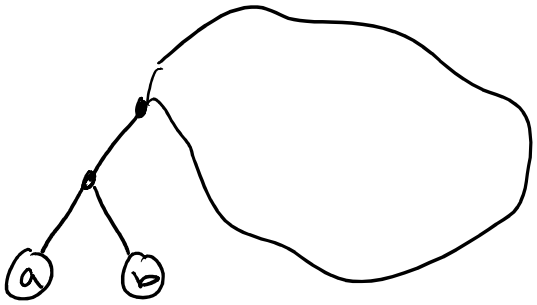
$$= (p_x - p_a)(d_{x/y} - d_a) + (p_y - p_b)(d_{x,y} - d_b)$$

≥ 0 ≥ 0 ≥ 0 ≥ 0

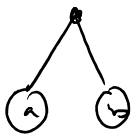
So new tree T must also have optimal average length, but $T \in X_{ab}$. So there is always an optimal tree with a, b siblings.

S.KIMMEL

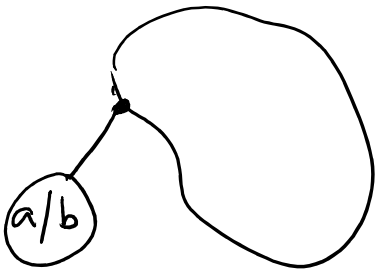
From 1, without loss of generality, there is an optimal tree T where a, b are siblings



Let T' be tree that is same as T, but with



replaced by (a/b)
one letter
(a/b) w/ probability $p_a + p_b$



S. KIMMEL

Q: What is $L(T) - L(T')$

$d_{a/b}$ is depth of node (a/b)

A: $L(T') = \sum_{i \in \Sigma - \{a,b\}} p_i d_i + (p_a + p_b) d_{a/b}$

$$L(T) = \sum_{i \in \Sigma - \{a,b\}} p_i d_i + (p_a + p_b)(d_{a/b} + 1)$$

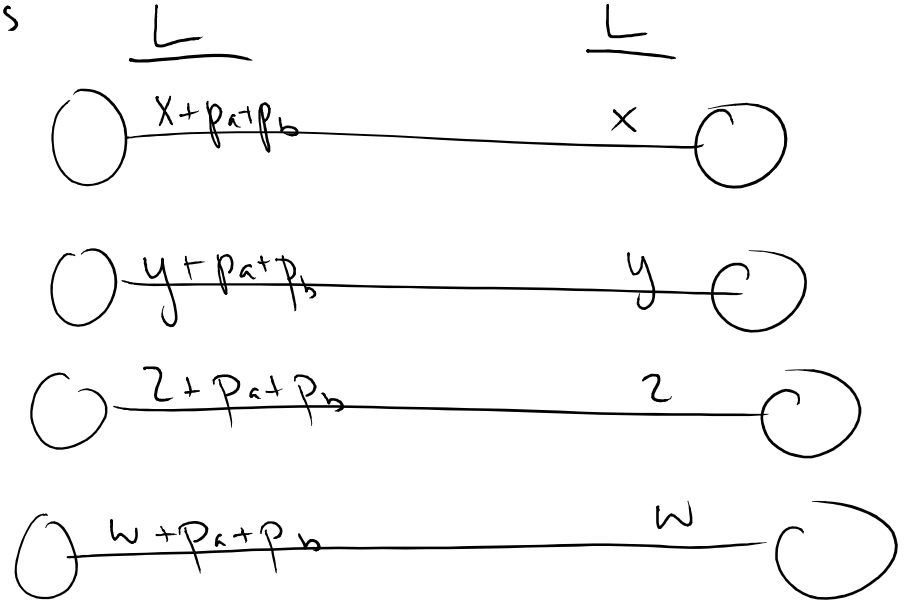
$$L(T) - L(T') = p_a + p_b$$

X_{ab}

Set of all trees where a, b are siblings

X'_{ab}

Set of all trees with symbol (a/b)



Tree with minimum L in X_{ab}, can be found by finding tree with minimum L in X'_{ab}

By inductive assumption, Huffman does this!

and replacing (a/b) with (a) (b)

Runtime

Initialize each $i \in \Sigma$ as tree

While (> 1 tree to be merged)

- Find 2 trees with smallest probability
- Merge into new tree with new probability = sum of old probabilities

Q. What is the runtime?

A) $O(n)$ B) $O(n \log n)$ C) $O(n^2)$ D) $O(n^2 \log n)$

S.KIMMEL

Runtime

Initialize each $i \in \Sigma$ as tree ← $O(n)$

While (> 1 tree to be merged) ← $O(n)$ reps

- Find 2 trees with smallest probability ← Extract twice $O(\log n)$
- Merge into new tree with new probability → $O(1)$
  = Sum of old probabilities
- Reinsert into heap : $O(\log(n))$

Create 2 new pointers

Q. What is the runtime?

A) $O(n)$ B) $O(n \log n)$ C) $O(n^2)$ D) $O(n^2 \log n)$

↑ Keep finding min over in over in a changing data structure

Use min-heap

- Initialize n elements in $O(n)$
- Extract min elt in $O(\log n)$
- Insert a new elt in $O(\log n)$

Using different data structure, can acheive $O(n \log\log n)$

van Emde Boas tree