7: Huffman

Thursday, March 25, 2021 3:58 PM

Binary Codes

Binary code: $f:\Sigma \to \{0,1\}^*$ where Σ is set of symbols

e.g. Morse code, ASCII, Braille, UTF-16

↑unicode using only 0,1

Q: Suppose you have a message where the letter "a" occurs 50% of the time, b occurs 30% of the time, and c occurs 20% of the time. What is the best binary encoding of a, b, and 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

Remember, you need to be able to decode: convert from binary back to symbols.

A)
a → 00
b → 01
c → 10

Avg. length per letter: 2

B)
a → 0
b → 1
c → 01

Ambiguous
01 = ab ?
or
01 = c ?

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

Not ambiguous.
Avg. length per letter:
$.5\cdot 1 + .3\cdot 2 + .2\cdot 2$
$= 1.5$

Average Length: $L(f) = \sum_{i\in\Sigma} \underbrace{\ell(f(i))} \cdot p(i)$

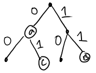
↑ length of encoding of letter i

↑ probability of letter i occurring

Binary Codes ↔ Binary Trees

a → 0
b → 11
c → 01

⟺



ℓ = depth from root

- $\forall i \in \Sigma$, there is a vertex with label i
- $f(i)$ is list of edge labels on path from root to vertex i

Ambiguity arises if 2 letters on same path

"Prefix-free": all letters @ leaves

no ambiguity

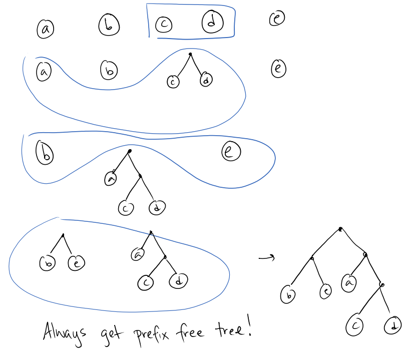
Optimal Binary Code Problem

Input: Σ (an alphabet of symbols)
$p : \Sigma \to \mathbb{R}^+$ (probabilities for each letter)

Output: $f : \Sigma \to \{0,1\}^*$ prefix-free binary encoding that minimizes average length:

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

Approach to Create Prefix-free Trees: Merge Trees



Always get prefix free tree!

Huffman's Algorithm

Why Greedy?

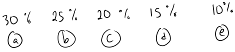
```
For each $i \in \Sigma$
    • Create a tree with one node, label node "i"
    • Give tree weight $p_i$        ← weight is f-value
While (more than 1 tree)
    • Merge 2 trees with smallest weight        ← greedy choice
    • Set Weight of merged tree to be sum of weights
      of trees involved in merger
                                    ↖ f-value changes
                                      each round
```

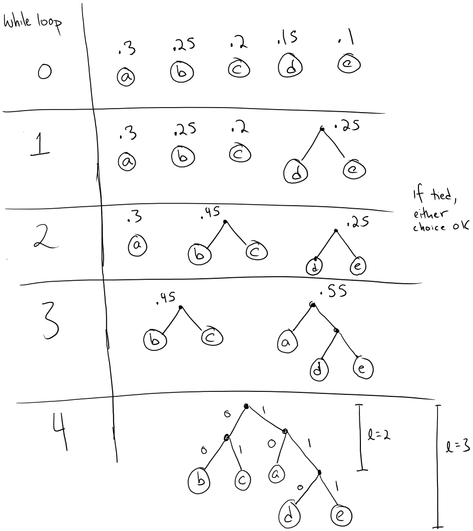
Consider

Letter	Probability
a	0.3
b	0.25
c	0.2
d	0.15
e	0.1

- Create optimal code/tree using Huffman
- What is av. length of your code?
- If $|\Sigma| = n$, what is the runtime of Huffman?

(Do min heap later...
not enough Knew)





If tied,
either
choice ok

$$L = .1\cdot 3 + .15\cdot 3 + .75\cdot 2 = 2.25$$

Huffman's Algorithm

```
O(n) { For each $i \in \Sigma$
         • Create a tree with one node, label node "i"
         • Give tree weight $p_i$
```

```
O(n) While ( more than 1 tree )
        · Merge 2 trees with smallest weight          O(n) to find smallest weight
        · Set Weight of merged tree to be sum of weights
          of trees involved in merger
```

$O(n^2)$

Huffman's Algorithm

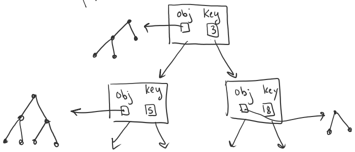
```
    For each $i \in \Sigma$                                  } Initialize n elts O(n)
        · Create a tree with one node, label node "i"
        · Give tree weight $p_i$
O(n) ← While ( more than 1 tree )
O(log n) {  · Merge 2 trees with smallest weight        } Remove min (x2) O(log n)
            · Set Weight of merged tree to be sum of weights
              of trees involved in merger
            · Reinsert merged tree into heap     O(log n)
```

$O(n \log n)$

Repeatedly find smallest element of a dynamic set? Use min heap! / **Priority Queue**

Min Heap / Priority Queue
- Initialize n elements : $O(n)$
- Find min elt : $O(1)$
- Remove min elt : in $O(\log n)$
- Insert a new elt : in $O(\log n)$

Min Heap / Priority Queue



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

van Emde Boas tree

What is the societal impact of Huffman's Algorithm?

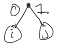
Societal Impact?? Good!

- Faster communication → more communication (limited bandwidth)

Correctness: Huffman's Alg Always Produces a Prefix-free code with smallest possible average length

Proof :

- Prefix-free ⟹ Because of merging approach
- Minimum Av. Length ⟹ induction on $|\Sigma| = n$.

Base case: If $n = 2$, get  . Optimal!

Inductive step: Assume optimal (Huffman produces a code with smallest possible av. length) for any input with $|\Sigma| = k$.

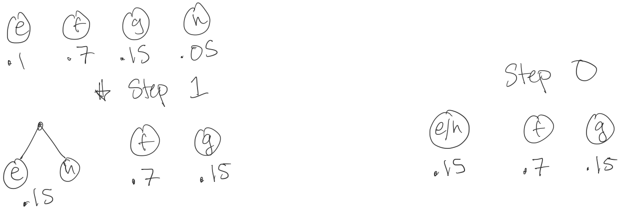
Consider an input with $|\Sigma| = k+1$. Let a, b be letters with smallest frequency. Consider $\Sigma^- = \Sigma - \{a,b\} + \{a/b\}$ ← new symbol "a/b"
with $p(a/b) = p(a) + p(b)$

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

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

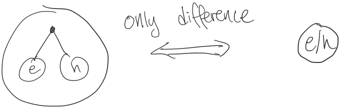
Huffman (Σ):

Huffman (Σ')

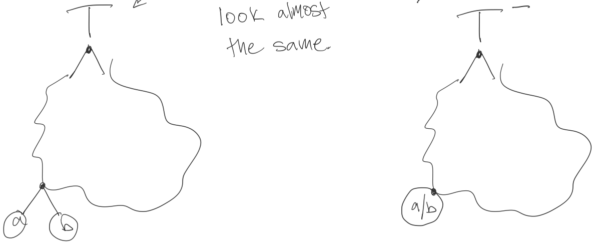


Future merges will be identical

⇓



Thus, Trees produced by Huffman for Σ an Σ^- look almost the same.



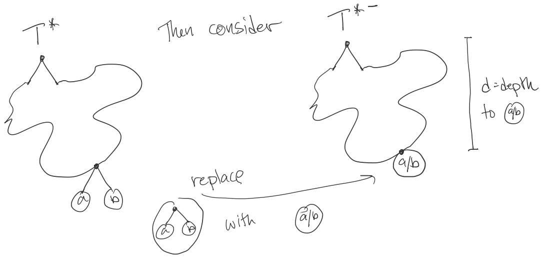
Since $|\Sigma^-| = k$, by inductive assumption, T^- is optimal.

Suppose for contradiction T is not optimal.

Lemma 1: There is an optimal tree with a, b siblings (Obviously Huffman's alg makes a, b siblings, but maybe there is a better approach that doesn't) (will prove later).

Let T^* be an optimal tree with a, b siblings. So

$T^* \neq T$.



Then:

$$L(T^*) = \sum_{i \neq a,b} p(i)\, \overset{\text{depth}}{d}(i) + \overset{(p(a)+p(b))(d+1)}{p(a)(d+1) + p(b)(d+1)}$$

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

(explain notation)

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

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

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

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

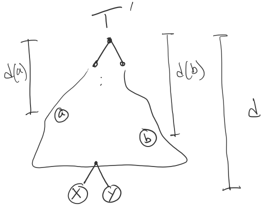
This is a contradiction, because ⇒ positive, since T^* is optimal T is not

so then ⇒ positive, which means T^{*-} better than T^-, a contradiction since T^- is optimal by our inductive assumption

Lemma 1: There is an optimal tree for Σ with a, b siblings.

Let T' be any optimal tree without a, b siblings.

Let x, y be sibling leafs in T' at maximum depth.



If we exchange positions x ⟷ a and y ⟷ b, then new tree T'_{ex} will have the same or smaller average length as T' because... Thus, ...