Greedy Algorithms (Contd.)

Thu: 4pm - 6pm
Office hours
A3 117.

Prefix trees:

"A quick brown dog jumps over a lazy fox".

↳ No. of bits needed

↳ 8 × # of char in the text.

26×2	letters
10	digits
2	char
64	} 8-bits to represent.

Space → 0

0/011

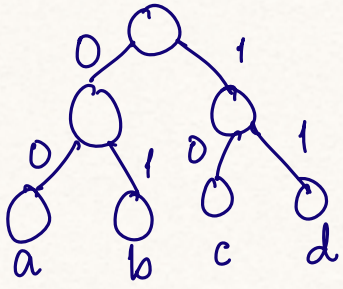
aabbcdd

a → 00 2/7
b → 01 → 2/7
c → 10 → 1/7
d → 11 → 2/7

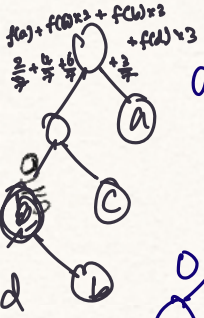
$(\frac{2}{7}+\frac{2}{7}+\frac{2}{7}+\frac{1}{7})2 = 1$

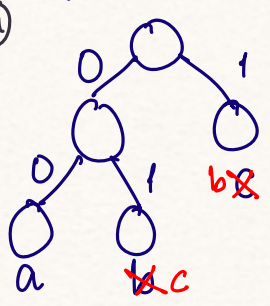
S ← Set of letters/ alphabet

T ← Binary prefix trees.



aabcc





$\frac{2}{5}\times2+\frac{1}{5}\times2+\frac{2}{5}\times1$

$\frac{4+2+2}{5}$

$\sim \frac{8}{5}$

a → 00
b → 01
c → 1

For every letter the bit representation is given by the root to leaf path.

For any pair of letters u, v, Enc(u) is not a prefix of Enc(v) and vice versa.

F: Frequency list

$F: S \to \mathbb{N}$

Average bit length of a tree (T)

$$= \sum_{x \in S} f(x) \cdot \text{depth}_T(x)$$

$f(x)$ ← Normalized

$$\sum_{x \in S} f(x) = 1$$

"Less freq. elements can be farther"

Obs: Frequent elements must be placed closer to the root.

Problem: $S \leftarrow$ Alphabet, $F \leftarrow$ freq. list

a, b, c, d, e
.32, .25, .20, 0.18, 0.05

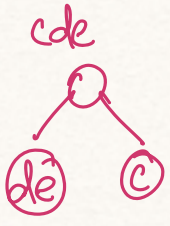
Want: An optimal prefix tree.

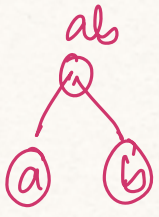
Huffman Coding

- Sort the elements in non decr. order of freq.
- Pick two least freq. elements. $u, v \in S$.
  - create a node w with u and v as children.
  - Remove u, v from S and add w to S.
    set $f(w) = f(u) + f(v)$.
- Repeat until S is empty.

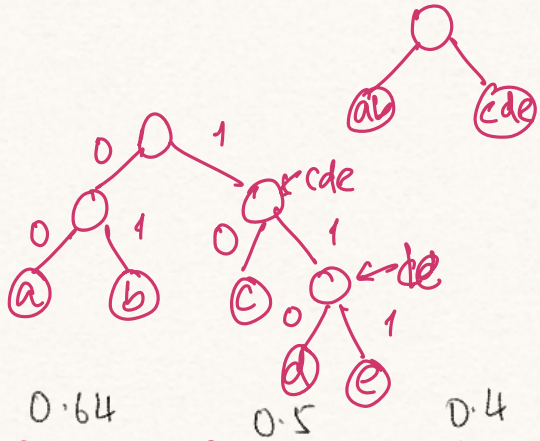
a, b, c, d, e
.32, .25, .20, 0.18, 0.05
{a, b, c, de} ← 0.23

0.32 0.20 0.25 0.43
S = {a, b, cde}





S = {ab, cde}
0.57 0.43



a	0 0	0.32
b	0 1	0.25
c	1 0	0.20
d	1 1 0	0.18
e	1 1 1	0.05

0.64 0.5 0.4 0.54

ABL = 0.32×2 + 0.25×2 + 0.20×2 + 0.18×3 + 0.05×3 (0.15)
= 2.23.

$S \longrightarrow S_1 \quad S_2$

$cf(S_1) \sim cf(S_2)$

↳ Tree need not be balanced.

$S_1 \longrightarrow S_{11} \quad S_{12}$

$S_{11} \longrightarrow S_{111} \quad S_{112}$

$S_{12} \longrightarrow S_{121} \quad S_{122}$

Correctness:

Lemma: Huffman coding produces an optimal prefix tree.

Proof: Induction on $|S|$. → Base case: $|S|=2$ ↖ Then 1-bit repr for each.

Induction step: $|S|=k$.

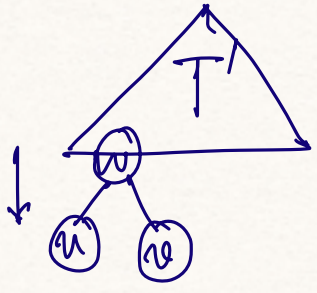
Algorithm generates a tree T.

For the sake of contradiction, T is not optimal.

Then, $\exists$ tree Z s.t $ABL(T) > ABL(Z)$. —— (*)

↳ T from T' by replacing w by u and v.

$T' \rightarrow$ Has $k-1$ letters and I.H tells us that T' is optimal.



$$ABL(T') = f(w)\cdot depth_{T'}(w) + \sum_{x \neq w} f(x)\cdot depth_{T'}(x)$$

$$ABL(T) = (f(u)+f(v))\cdot(depth_{T'}(w)+1) + \sum_{x\neq w} f(x)\cdot depth_{T'}(x)$$

$$= f(w) + f(w)\cdot depth_{T'}(w) + \sum_{x\neq w} f(x)\cdot depth_{T'}(x)$$

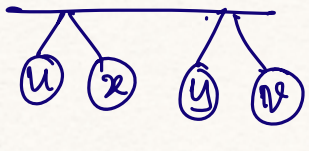
$$= f(w) + ABL(T').$$

Qu: Are u and v siblings in Z?

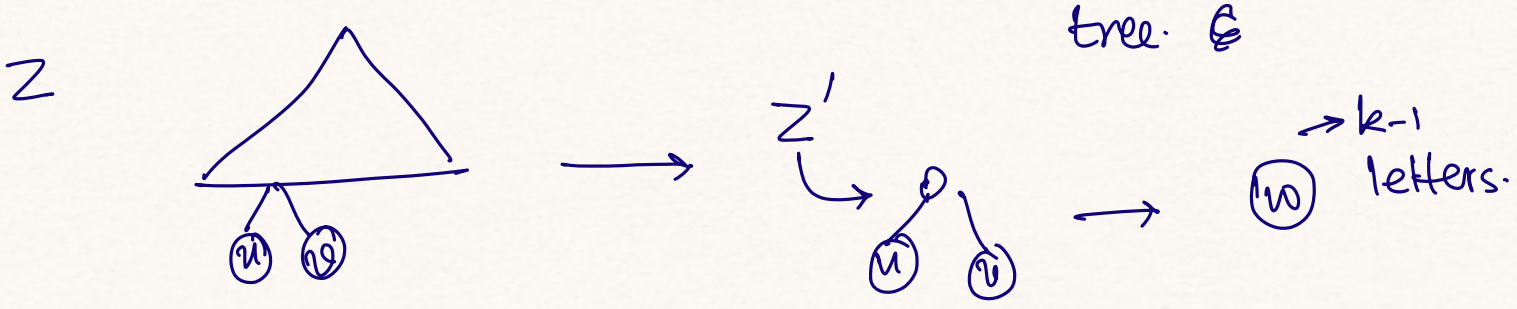
Or do they occur at same depth?

(if they donot occur at same depth?)

Contradicts optimality of Z.



By swapping u and x, we get a better ABL tree. ☹



Claim: $ABL(z') = ABL(z) - f_w$.

$$\sum_{x \neq w} f(x) \cdot depth_{z'}(x) + f(w) \cdot depth_{z'}(x)$$

$$\sum_{x \neq u,v} f(x) \cdot depth_z(x) + f(u) \cdot depth_z(u) + f(v) \cdot depth_z(v)$$

From I.H

$ABL(z') \geq ABL(T')$ ← T': $k-1$ letters.

$ABL(z) - f(w) \geq ABL(T) - f(w)$

$ABL(z) \geq ABL(T)$.

↳ Contradicts (☹).