02/04 [Section 4.8 of the book]

Huffman Codes and Data Compression

Consider storing or communicating a large text file.

How many bits do we need to store or communicate?

26 alphabets, punctuation symbols, blank space

≤ 32 symbols.

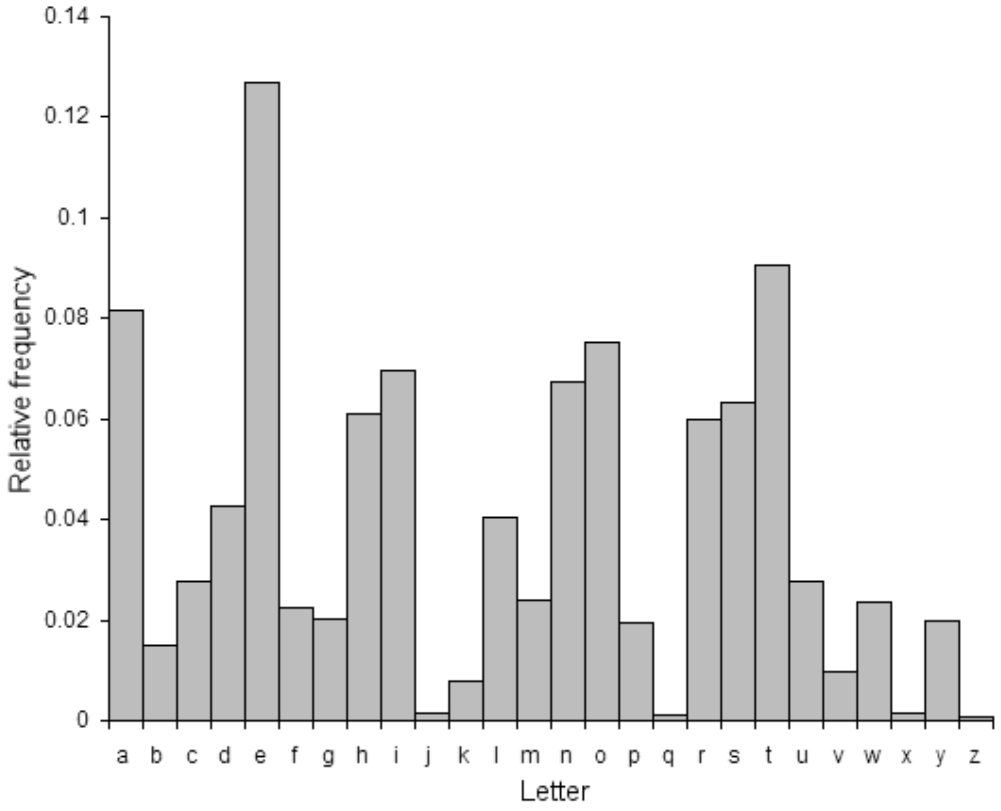
Encode each symbol using 5 bits.

Text file has n symbols

$\Rightarrow$ Use $5n$ bits. ✓

Question: Can we do better? How?

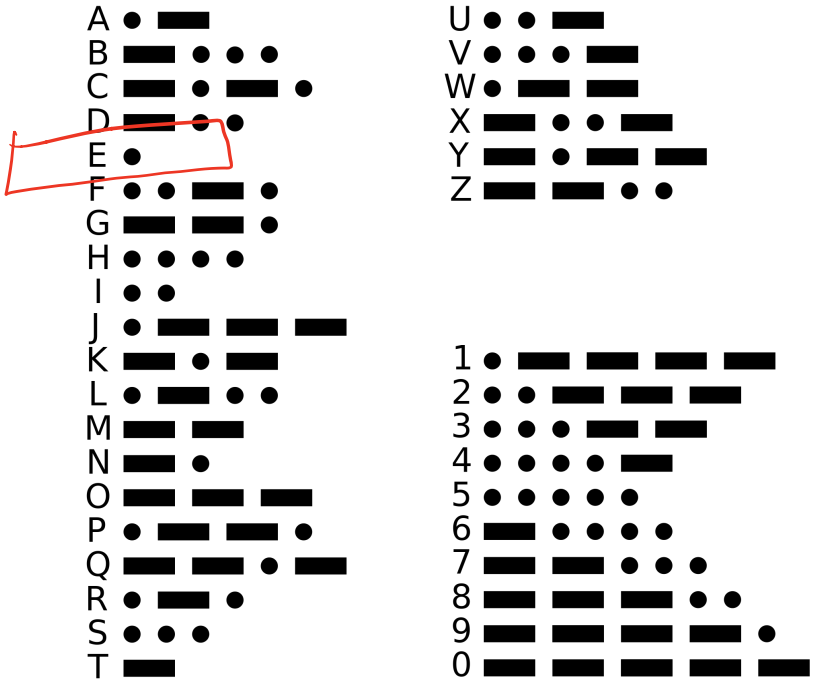
✓ Observation: Some letters in the alphabet are used more frequently.



→ This idea was already used in Morse code.

International Morse Code

1. The length of a dot is one unit.
2. A dash is three units.
3. The space between parts of the same letter is one unit.
4. The space between letters is three units.
5. The space between words is seven units.



Source: Wiki

(i) a a ✓
(ii) e t e t
(iii) e t a
(iv) a e t
(v) e k

Morse: adds a 'pause' after each symbol.

→ A Ternary encoding.

→ Prefix code: general way to solve this problem.

02/06

Prefix codes

Let S be a set of symbols

Encoding function ≡ Prefix code

$\gamma : S \rightarrow \{0,1\}^*$ binary strings of arbitrary length.

Property of γ:

For any distinct $x, y \in S$,

$\gamma(x)$ is not a prefix of $\gamma(y)$.

How to encode a string using γ:

$x_1 x_2 \cdots x_i \cdots \quad (x_i \in S)$

$\gamma(x_1)\ \gamma(x_2) \cdots \gamma(x_i) \cdots$

Decoding procedure:

(i) Scan from left to right.

(ii) As soon as the bit-string makes sense, map it to some $x_i \in S$.

(iii) Delete bits seen so far.

e.g. $a \to 01$, $b \to 100$, $c \to 101$

Is this a prefix code?

(a) Yes (b) No

(ii) $a \to 01$, $b \to 10$, $c \to 101$

No

Notation: For any string x, $|x|$ to denote length of x.

Designing good prefix codes.

Symbols : S

Frequencies: For each $x \in S$,

f_x ($f_x \in [0,1]$)

Given γ, ← prefix code

$$ABL(\gamma) = \sum_{x \in S} f_x |\gamma(x)|$$

← average bits per letter

Example: $a \to 01$, $b \to 100$, $c \to 101$

$f_a = 0.4$ $\quad f_b = 0.3$ $\quad f_c = 0.3$

$$ABL(\gamma) = 2 \times (0.4) + 3(0.3) + 3(0.3)$$
$$= 2.6 .$$

Text T has n symbols; $ABL(\gamma) = \delta$.

H

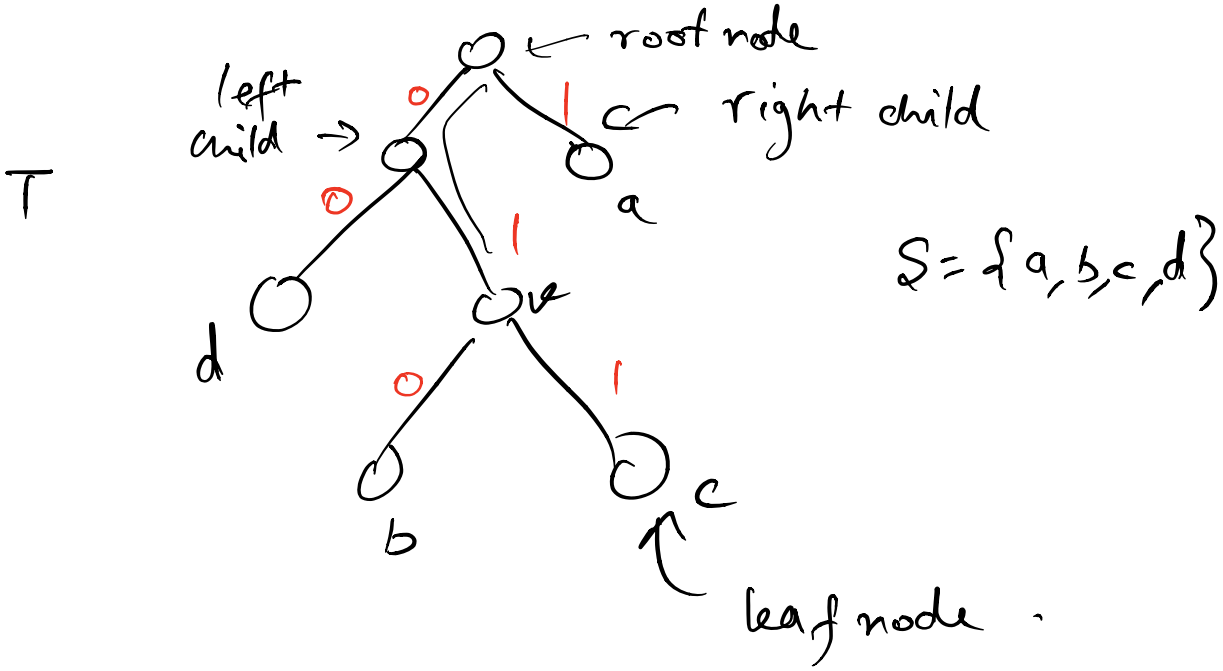
$$\gamma(H) = \delta n .$$

$$\hookrightarrow \sum_{x \in S} f_x \cdot n \, |\gamma(x)| = n \cdot ABL(\gamma)$$
$$= n \cdot \delta .$$

Task: Design γ with minimum ABL.

Prefix codes as labelled binary trees.

Labelled binary tree



All the leaf nodes are labelled by distinct symbols from S.

For any node v,

$$\text{depth}_T(v) = \text{length of path (edges) from root to } v.$$

→ Given a labelled binary T with distinct symbols from S, it specifies a

prefix code γ.

In the example: $\gamma(a) = 1$,

$\gamma(b) = 010$, $\gamma(c) = 011$, $\gamma(d) = 00$

General process: Walk from root to leaf labelled x. Whenever going left add 0 to the end, else add 1.

Obs: γ is a prefix code.

Pf: $x, y \in S$.

If $\gamma(x)$ is a prefix of $\gamma(y)$,

y is a node in a subtree with x as the root node.

But x, y are leaf nodes.

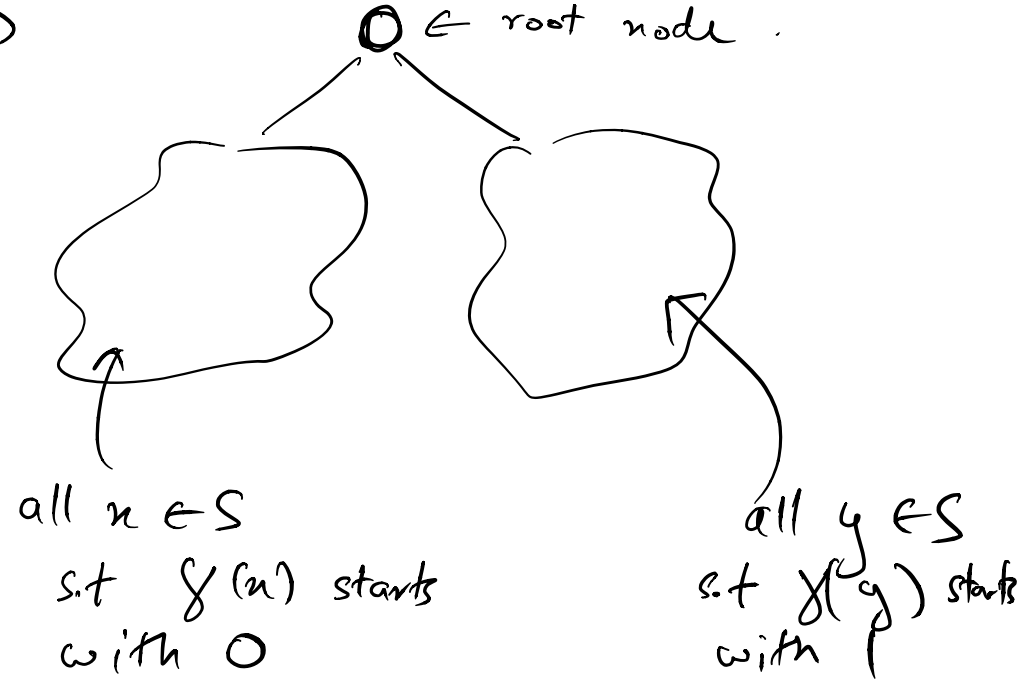
So this is impossible.

Thus, $\gamma(x)$ is not a prefix of $\gamma(y)$.

Given: γ is prefix code for S.

Construct a labelled binary tree

→



Recursively apply on the clouds.

→ From now on, let's try to build a labelled binary tree.

$$ABL(T) = \sum_{x \in \text{Leafnodes}(T)} f_x \, \text{Depth}_T(x)$$

If T is constructed out of γ,

Obs. $ABL(\gamma) = ABL(T)$.

Natural strategy [Fano and Shannon]

Input: S, set of frequencies $x \in S$, f_x.

Algorithm: Partition S into S_1 and S_2 s.t

$\sum_{x \in S_1} f_x$ is as

Close to $1/2$ as

possible.

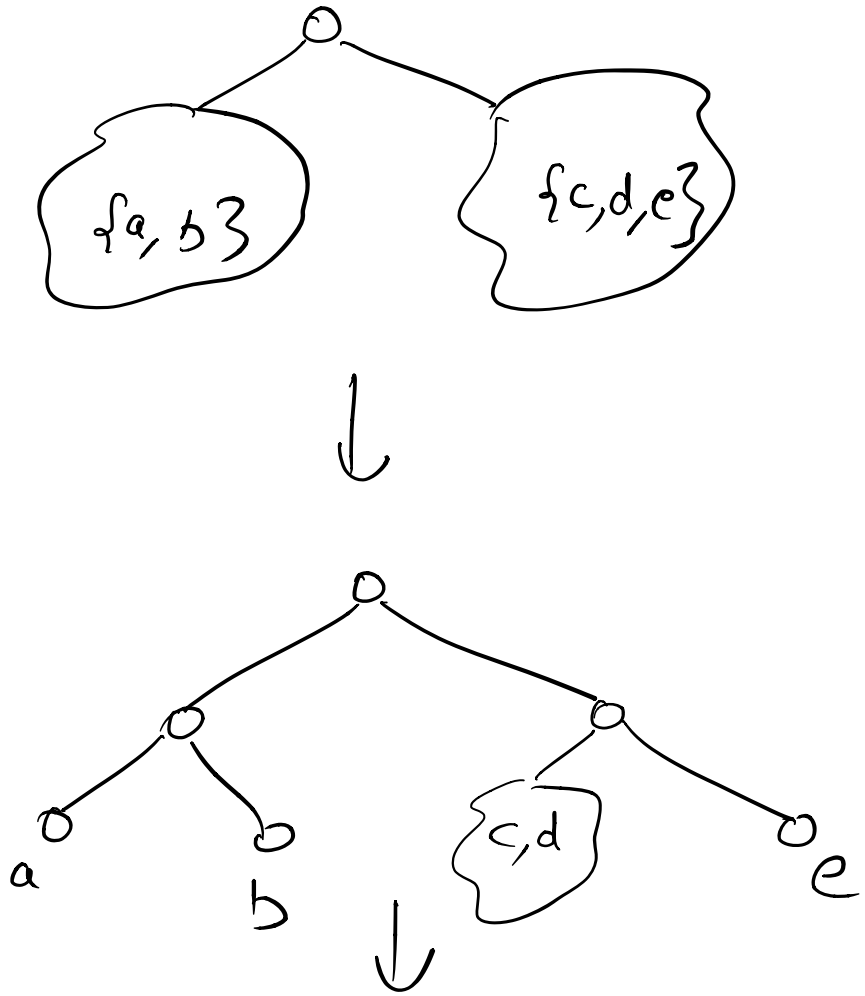
Recurse on S_1 and S_2.

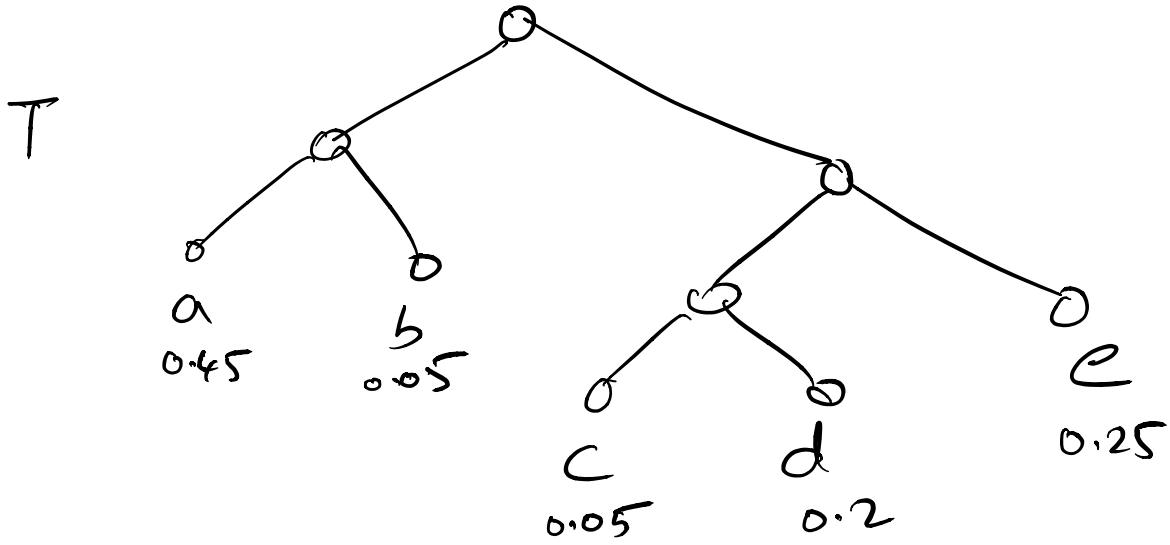
Example: $S = \{a, b, c, d, e\}$

$f_a = 0.45$ $f_b = 0.05$

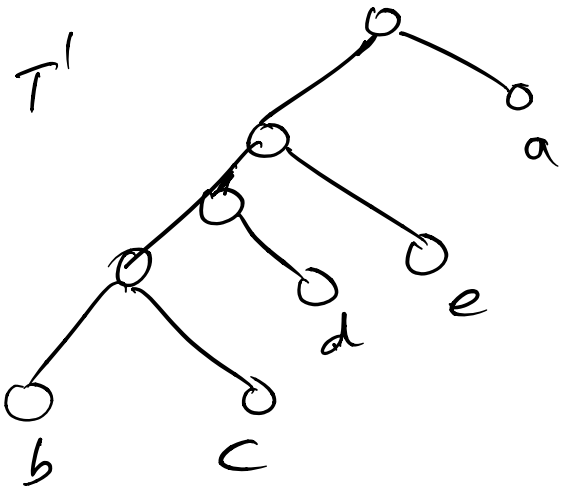
$f_c = 0.05$ $f_d = 0.2$

$f_e = 0.25$





$ABL(T) = 2.25$



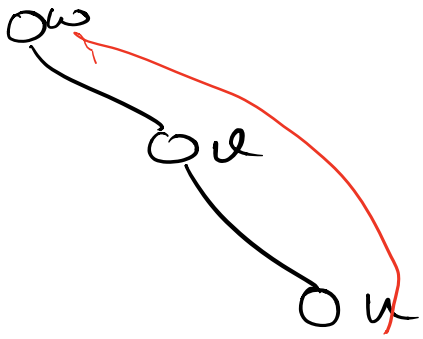
$ABL(T') = 1.95$

Huffman codes.

Let T be an optimal prefix code.

Obs T is full. (i.e each non-leaf node has two children)

Pf: Use exchange argument.



[see book]

Obs: Let x, y be leaf nodes in T.

If $f_x < f_y$, then

$\text{depth}_T(x) \geq \text{depth}_T(y)$.

Pf: Exchange argument.

$T \rightarrow T'$

$ABL(T) - ABL(T')$

→ Suppose you are given an optimal tree without labels.

Task: Find labelling.

→ Sort according frequencies

→ Label least frequent symbols with larger

depth.

Algorithm

(i) If S has 2 letters

encode each letter using 1 bit.

(ii) Else

Let v, w be the least frequent letters in S.

$$S' = (S \setminus \{v, w\}) \cup \{\alpha\}$$

(α: new letter)

$$f_\alpha = f_v + f_w$$

(iii) Recursively solve on S' to get tree T'.

(iv) Construct tree T from T' by: removing the label α from the leaf, adding a left and right child, labelling u and w.

endif.

02/08

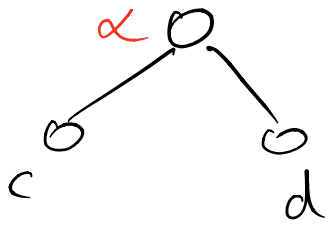
Analysis of the algorithm.

Let's start with an example:

$a \rightarrow 0.3$ $b - 0.3$ $c \rightarrow 0.2$

$d \rightarrow 0.2$

$\alpha \rightarrow \{c, d\}$ $f_\alpha = 0.4$



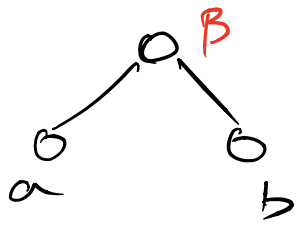
$a \rightarrow 0.3$

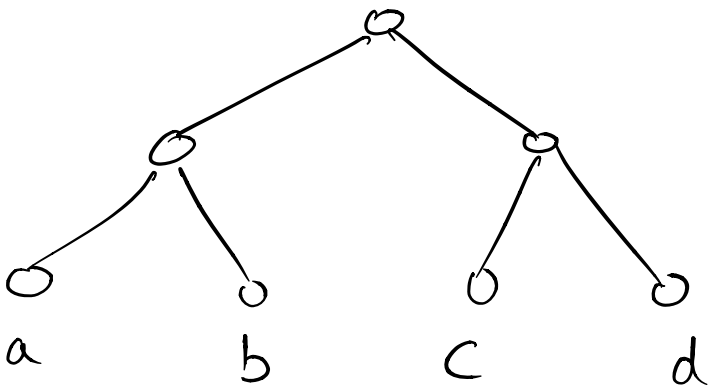
$b \rightarrow 0.3$

$\alpha \rightarrow 0.4$

$\beta \rightarrow \{a, b\}$

$f_\beta = 0.6$





Proof of correctness

Induction on $|S| = K$.

Base case: $K = 2$. Direct from algorithm

Inductive step: Let's assume correctness for $K-1$.

Let v, w be the least frequent letters in S.

$$S' = (S \setminus \{v, w\}) \cup \{\alpha\}$$ ← new symbol

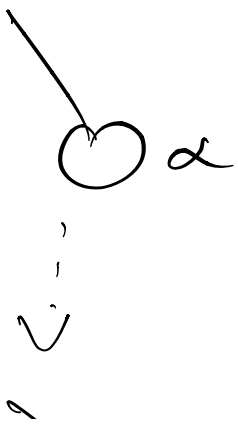
$$f_\alpha = f_v + f_w$$

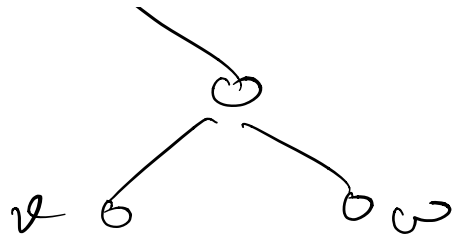
Let T' be the output of the algorithm on S'.

By induction, T' is the optimal prefix code for S'.

<u>Obs</u> $ABL(T) - ABL(T') = f(v) + f(w)$

<u>Pf</u>: Only leaf nodes that change in T' is that labelled with α.





□

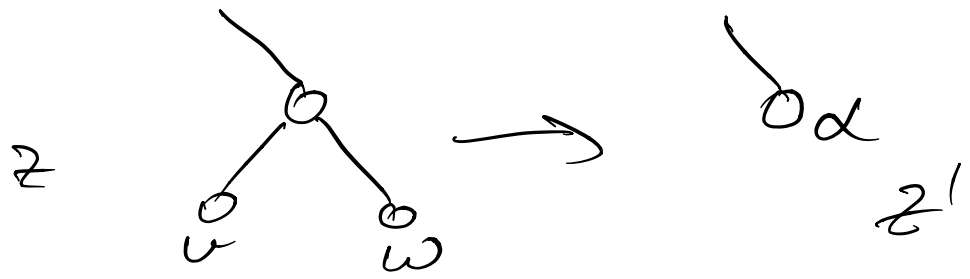
Obs: There is an optimal tree with v and w as sibling leaf nodes.

Pf: (a) optimal tree is full
(b) Follow greedy labelling.

□

Let Z be an optimal tree with v, w as sibling leaf nodes.

Let Z' be a tree with labels from S'. [Same way as T' is constructed from T.]



$$ABL(z) - ABL(z') = f(v) + f(w)$$

$$ABL(T') \leq ABL(z')$$

(by inductive hyp.)

$$\Rightarrow ABL(T') + f(v) + f(w) \leq ABL(z') + f(v) + f(w)$$

$$\Rightarrow ABL(T) \leq ABL(z)$$

$$\Rightarrow ABL(T) = ABL(z)$$ (since z is optimal)