Huffman Codes

File of characters, drawn from some alphabet.

We wish to code this file as a binary string.

Want to do it as efficiently as possible
(In particular length of the binary string should be Small)

A simple way of coding is character by character, each character is assigned a binary string (called code for that char.)

(i) Fixed length code

<u>Example</u>: 100 characters in the alphabet

then each character may be assigned a binary string of length 7 $(2^7 = 128)$

(ii) Variable length code

Some character in the file may be occurring more frequently than others.

The characters which occur more frequently are assigned shorter code-word, less frequently occurring characters are assigned longer code-words.

code words

Example:

a , b , s , x , z

40% 20% 25% 10% 5%

a → 0
s → 10
b → 110
x → 1110
z → 1111

Code

Length for 100 char file $= 40\times1 + 25\times2 + 20\times3 + 10\times4 + 5\times4$
$= 40 + 50 + 60 + 60 = 210$

(Fixed length code will have length ≥ 3 and 300 bits are required to code this file)

[Code: character $\longrightarrow$ binary string]

Prefix code

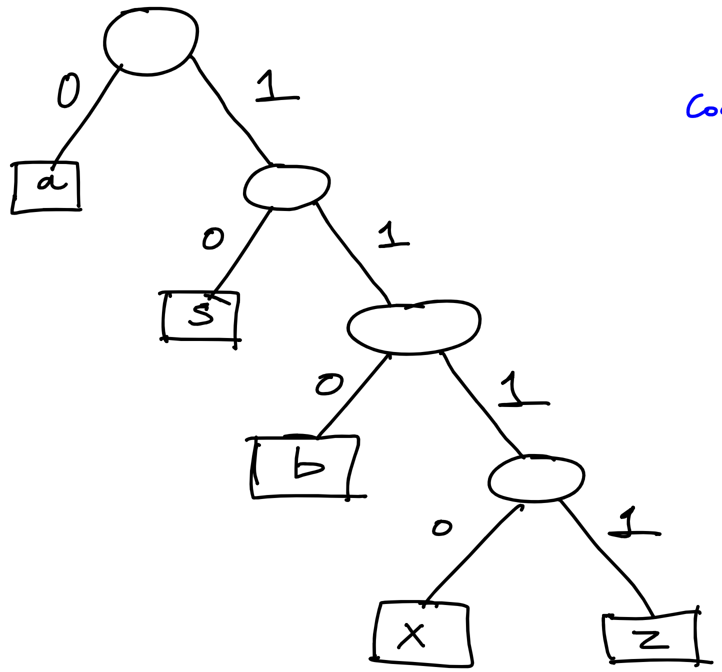
No code-word is prefix of another code-word

[Ex: Code words 0, 1, 01, 10 do not form a prefix code.]

The code given on the previous page is a prefix code.

Prefix codes can be represented by a binary tree. Each leaf is labeled by a character, and code for that character is the binary string representing the path from root to that leaf.

(Lchild direction $\rightarrow$ 0
Rchild direction $\rightarrow$ 1)



Decoding: start reading the coded file, from the root of the tree.

when 0 is read move to lchild of current node in tree,

when 1 is read move to rchild of current node in the tree.

when one hits a leaf then output character labeling the leaf.

Start reading rest of the file by going back to root of this tree.

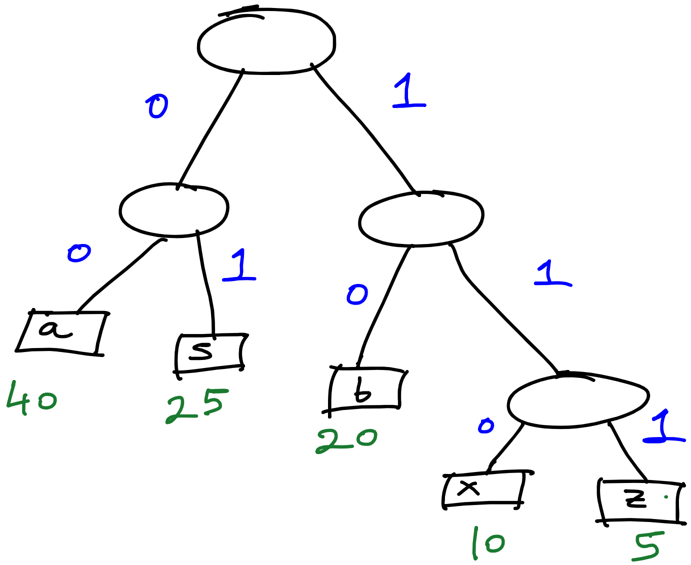
SSXa

encode → 1010111100

decode → S1011100

→ SS11100 → SSX0 → SSXa

(simple one-pass algorithm for decoding)



$a \rightarrow 00$

$s \rightarrow 01$

length of the coded file (no. of bits)

$$= 40\times 2 + 25\times 2 + 20\times 2 + 10\times 3 + 5\times 3$$

$$= 80 + 50 + 40 + 30 + 15$$

$$= 215$$

Not optimal

<u>Problem</u>: Is to find optimal code

Problem: Given characters and their frequencies

c_1	c_2	$\cdots$	c_i	$\cdots$	c_n
f_1	f_2	$\cdots$	f_i	$\cdots$	f_n

Define prefix code for $\{c_1 \cdots c_n\}$ which results in minimum <u>length coded file</u>.

$$\left(\sum_{i=1}^{n} f_i \cdot |code(c_i)| \right)$$

In other words, if prefix code is represent by tree T.

T

$$wt(T) = \sum_{i=1}^{n} f_i \cdot \overbrace{d_T(c_i)}^{|code(c_i)|}$$

$d_T(c_i)$ is the depth of the leaf containing c_i.

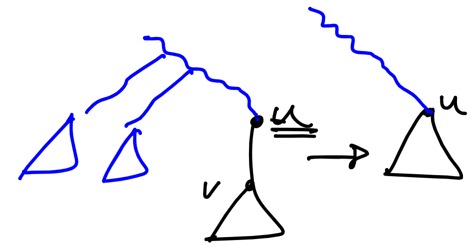
We need to find such tree with the minimum weight.

We will see a greedy method to define Huffman code.

How to reduce the given problem into subproblem?

<u>Observation 1:</u> (Let $n>1$) In the tree representing optimal Huffman code, every internal node has exactly two children.

<u>proof:</u> If there is a node u with exactly one child v then tree rooted at v may be planted at u, resulting in shorter codes for some characters. The new tree has lesser weight than the original tree. This is a contradiction as the original tree was optimal.



$\Longrightarrow$ there is no node with exactly one child. $\square$

<u>Observation 2</u> In an optimal Huffman tree there is at least one internal node which has two leaf nodes as its children.

<u>Pf</u>:

$$n_0 = n_2 + 1 \quad \text{—— (1)}$$

If every internal node has at most one leaf node as its child then (also each leaf node is a child of some internal node)

$\implies n_0 \leq n_2$. A contradiction to equation (1) above

$\square$

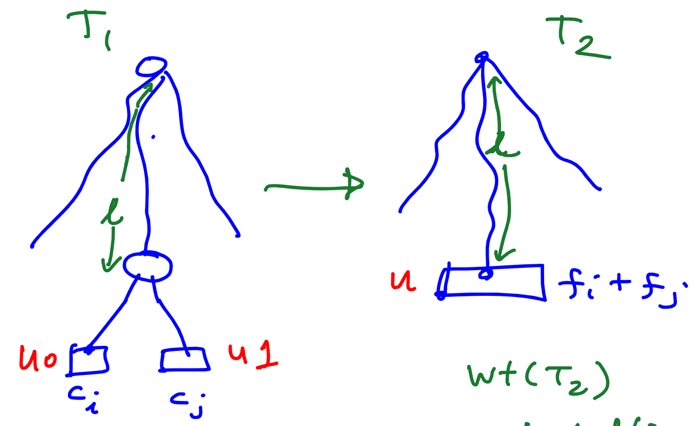
This observation gives us a way to express solution to our problem in terms of solution to a subproblem.

Given $C = \{(c_1, f_1), (c_2, f_2) \ldots (c_n, f_n)\}$

choice (f_i, f_j)

We can construct a subproblem

$$C' = [C - \{(c_i, f_i), (c_j, f_j)\}] \cup \{(c_{ij}, f_i + f_j)\}$$



$wt(T_2) = w + l(f_i + f_j)$

$wt(T_1) = w + (l+1)(f_i + f_j)$

$wt(T_1) = wt(T_2) + (f_i + f_j)$

How to construct solution to C from C'.

(i) If solution S' of C' assigns code u to c_{ij} then

$$\begin{cases} Code_S(c_i) = u0 \\ Code_S(c_j) = u1 \\ Code_S(d) = code_{S'}(d), \text{ for all } d \in C - \{c_i, c_j\} \end{cases}$$

(ii) $\underset{\text{minimize}}{wt(S)} = wt(S') + \underbrace{(f_i + f_j)}$ ← choice

Now we have expressed solution to the given problem in terms of solution to a subproblem and a choice

Several choices possible, corresponding to pairs f_i, f_j

Greedy choice choose f_i, f_j with minimum value.