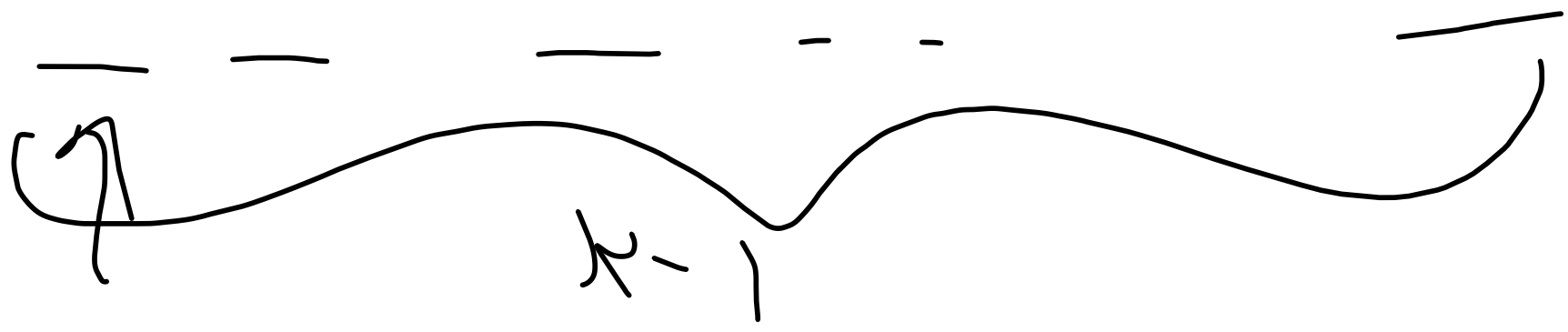
Clustering

Run Kruskal for n-k steps

Thm The k-clustering obtained by running Kruskal's alg. for n-k steps has maximum spacing

Pf Let C be the k-clustering consisting of $C_1, C_2, \ldots, C_k$.

Running Kruskal for $n-k$ steps is equivalent to running the full Kruskal alg. & deleting the $k-1$ most expensive edges



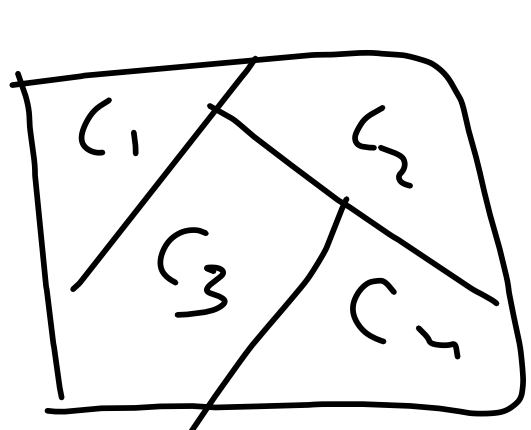
On the next step, Kruskal's alg. would have picked up the $k-1^{st}$ most expensive edge in the MST

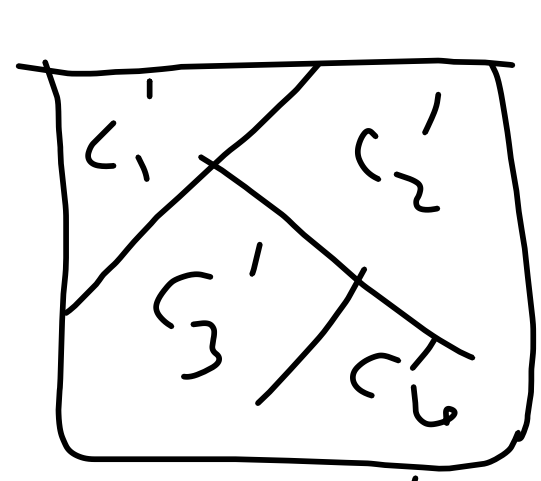
Clearly, the length of this edge is the spacing of C^* by δ.

Fix a k-clustering C' consisting of $C'_1, C'_2, \ldots, C'_k$

We shall show that the spacing of C' is at most δ^*.

If each C_i' is contained in some C_j, then $C = C'$.





If $C_1 \neq C_1'$, then $\exists x \in C_1 - C_1'$

W.l.s. assum $x \in C_3 \subseteq C_3'$

$C \subseteq C'$

$C_1 \subseteq C_1'$

$C_2 \subseteq C_2'$

$C_3 \subseteq C_3'$

$C_4 \subseteq C_4'$

Since $C \neq C'$, $\exists$ C_γ

s.t. C_γ is not a subset of

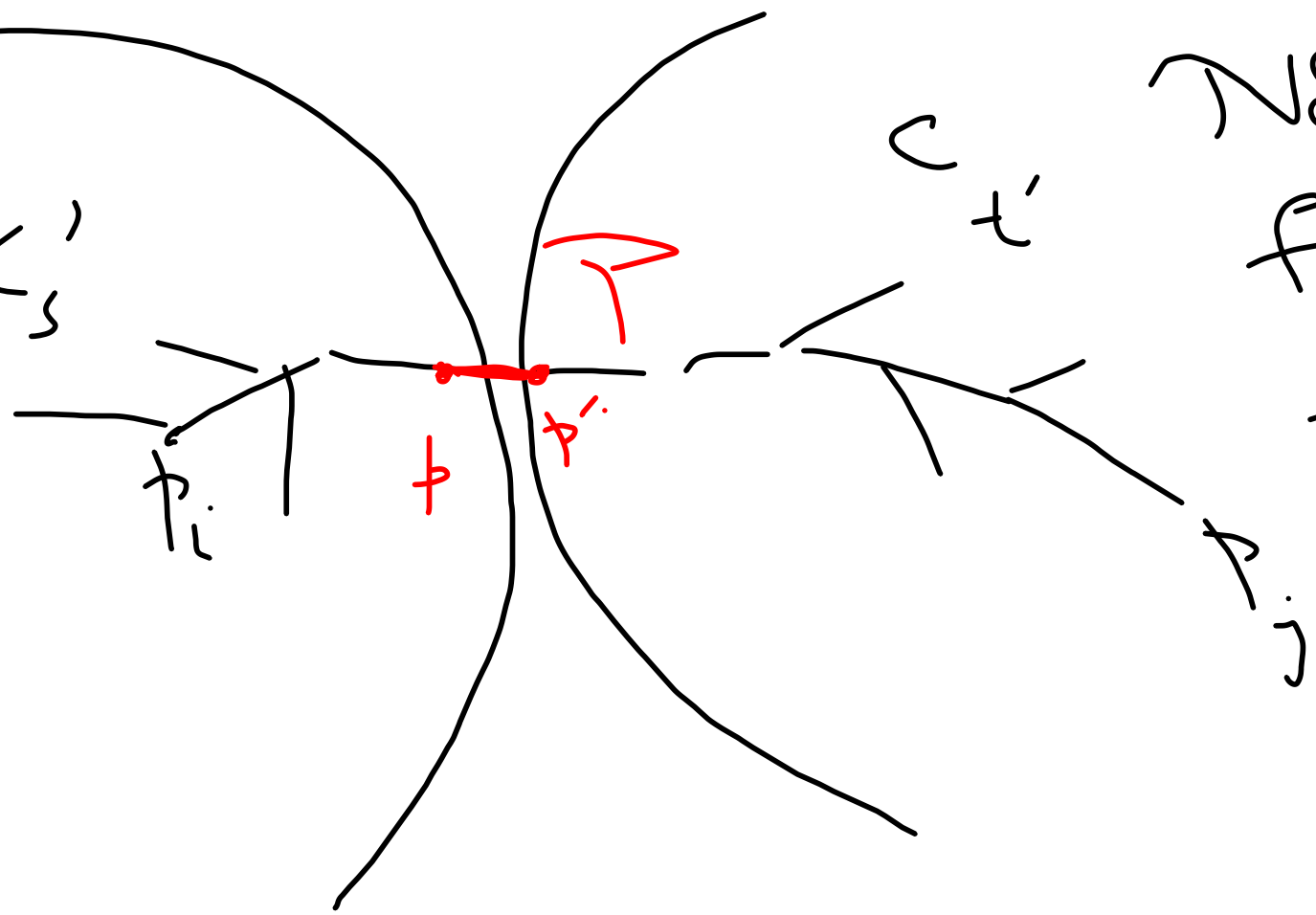
any C_j'. This means $\exists$ a pair

p_i, p_j in C_γ s.t. p_i, p_j belong to

different clusters of C'

Assume $p_i \in C_s'$ & $p_j \in C_t' \neq C_s'$

Consider a path P from p_i to p_j in G



Note that for each edge e in P

$d(e) \leq \delta^*$

Let p' be the first node on Γ that does not belong to C_s' & let p be the previous node on P

Clearly, $d(p,p') \leq \delta^*$.

Clearly, the spacing of C_s $\leq d(p,p') \leq \delta^*$.

Huffman Code & Data Compression

Since computers work on bit strings any text in a richer ~~language~~ alphabet, say the English alphabet, requires an encoding scheme into bit strings. For a fixed length encoding, each English letter, requires a 5-bit encodings

Suppose we try to encode in such a way that most frequently occurring letters use shorter encodings

e ← — → z

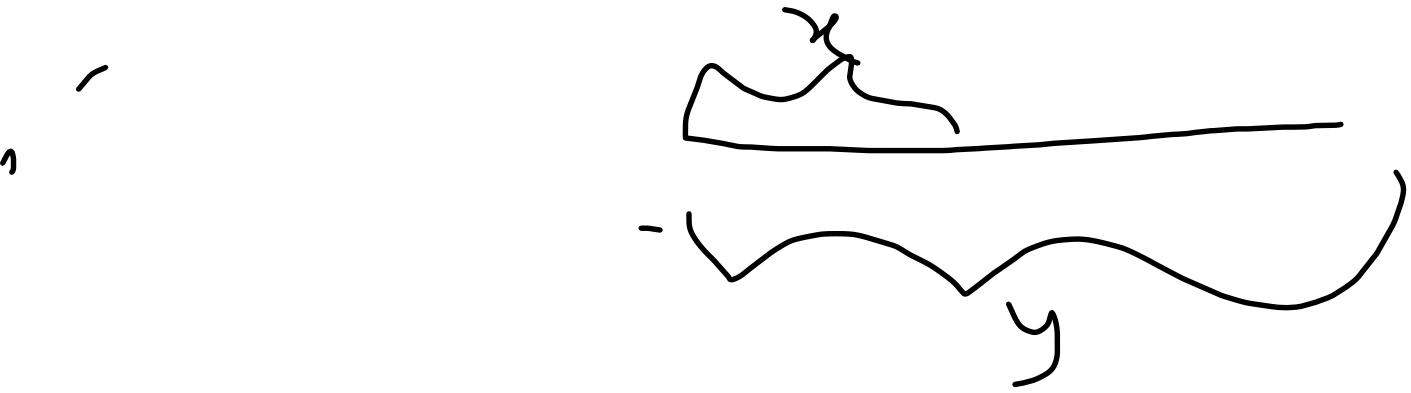
t ← — → a

45 + 5 x 55

- 320

. - . - .

But using variable length encoding
leads to some problem while decoding
Suppose $\exists$ x, y s.t. encoding of
x is a prefix of the encoding of y.



Defn A prefix code for an alphabet S is a fn $\gamma : S \longrightarrow \{0,1\}^*$ s.t. for any pair $x, y \in S$, neith $\gamma(x)$ nor $\gamma(y)$ is a prefix of the other

Suppose γ is a prefix code of an alphabet S. For each $x \in S$, let f_x be the fraction of the letters occurring in a text.

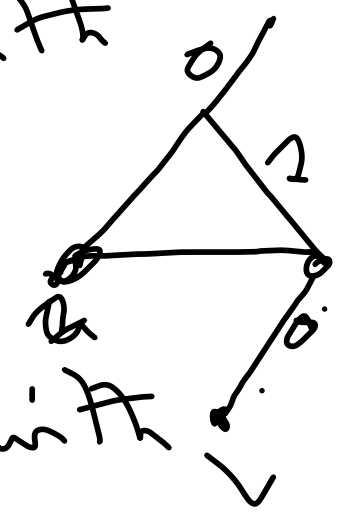
So if n be the no. of letters in the text, then nf_x is no. of times the letters are equal to x. Hence the total length of the encoding is $\sum nf_x|\gamma(x)|$. Hence the average no. of bits required per letter is

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

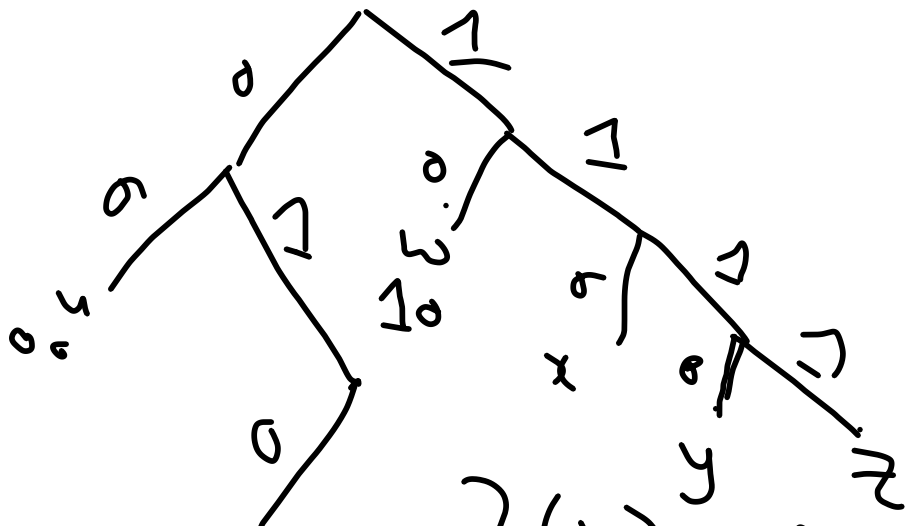
Our aim is to obtain a prefix code
γ s.t. $ABL(\gamma)$ is as small as possible.
Such a code is called an Optimal
Prefix Code.

Prefix Code from binary trees.

Let T be a binary tree with no. of leaves equal to |S|



Label each leaf with a letter of S.



For each leaf labeled by x, consider the path from the root to the leaf

$\gamma(u) = 00$

$\gamma(v) = 010$

$\gamma(w) = 10$.

$\gamma(x) = 110$.

$\gamma(y) = 1110$., $\gamma(z) = 1111$

Each time you traverse left, write down a 0 & each time you traverse right, write down a 1.

The resulting string is an encoding of x.

Clearly, this is a prefix code

For each prefix code one can construct a binary tree as follows.

All encodings starting with 0/1 are
the encodings of the leaves labelling
the leaves of the left/right subtree.
Recursively construct the left &
the right subtree.

Clearly, the length of the encoding of x
is the length of the path from the
root to the node labeled by x.

This is called the **depth of v**

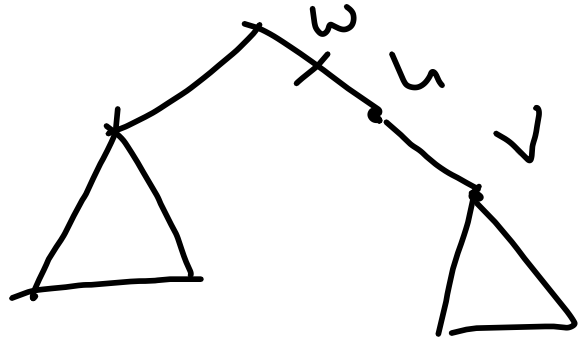
d is denoted by $\text{depth}_T(v)$

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

Defn Call a binary tree **full** if each internal node has exactly two children.

Lemma 1. The binary tree representing an optimal prefix code is full.

pf. Suppose there is a node u with only one child v. If u is a root node, remove u & make v the root node.



C.w. u has a parent w. Remove u & make v the child of w.

By this we obtain a tree T'

s.t. $ABL(T') < ABL(T)$ Contradiction.

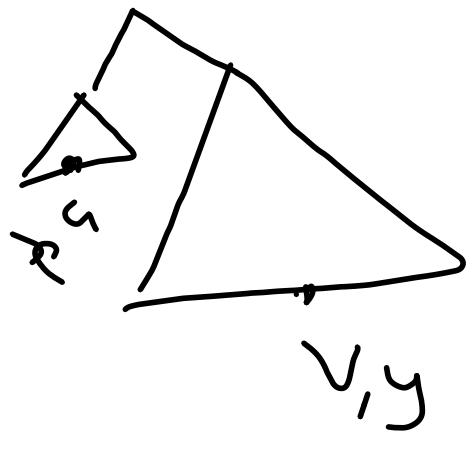
Lemma 2 For each $x \in S$, let f_x denote the fraction that x occurs. Let T be a binary tree representing an optimal prefix code.

Let u, v be two leaves of T

s.t. $depth_T(u) < depth_T(v)$

Let the labels of u, v be x, y resp.

Then $f_x \geq f_y$.



pf Let T' be the tree obtained from T by interchanging the labels of u & v

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

$$T \Longrightarrow f_x \, depth_T(x) + f_y \, depth_T(y)$$

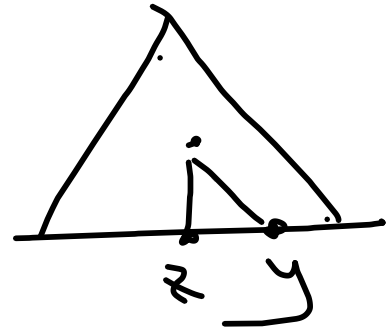
$$T' \longrightarrow f_x \, depth_{T'}(y) + f_y \, depth_{T'}(x)$$

$$= f_x \, depth_T(y) - f_x \, depth_T(x) + f_y \, depth_T(x) - f_y \, depth_T(y)$$

$$= (f_x - f_y)(depth_T(v) - depth_T(u)) \geq 0$$

Lemma 3 Let T^* be a binary tree rep^n an optimal prefix code. Let y be the label of a leaf with min. frequency. Then y has a sibling z in T^*.

pf



Since the tree is full, y has a parent w with another child z

If z has a child, then its depth would be greater than that of y Contradiction.