Greedy strategies
(Huffman codes)

→ Encode $\{a, \ldots, z\}$ using $\{0, 1\}$
will need 5 bits (32 5 bit strings)
if each letter is a 5 bit string

→ Can we optimize amount of data to transfer?

if n letters $\Rightarrow$ $5n$ bits

→ Variable length encoding

(Morse code) $\cdot$ —
0 1

freq letters $\begin{cases} e - 0 \\ t - 1 \end{cases}$

next freq $\{ a - 01$

But decoding ??

$0101 \longrightarrow aa$

;

$\swarrow \qquad \searrow$

et et $\qquad$ et a

. - - -

→ In practice, "slight pause" (ambiguous)

between letters.

→ so morse code is $\cdot$, —, and , pause

(3 alphabet)

---

want Variable length coding with Prefix code property

* $E(x)$ = encoding (x)

* $E(x)$ not a prefix for any $E(y)$]

prefix code property

(eg)

x	a	b	c	d	e

$E(x)$ | 11 | 01 | 001 | 10 | 000

001 000 001 11 01

c e c a b

(unambiguous decoding possible)

Optimal prefix codes

Given $f(x)$ = freq of x (statistical analysis) $\forall x$

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

→ n letter message

→ so $\approx n f(x)$ # of occurences of x

→ $$\sum_{x \in \text{alphabet}} \text{Length}[E(x)]\, n\, f(x)$$ = # of bits to encode n letter message.

(Expected #)

→ Avg # of bits / letter

$$= \sum_{x \in \text{alphabet}} |E(x)| \, f(x)$$

[Fixed length code: each letter is encoded by m bits $\forall$ letters in alphabet, $|E(x)| = m$]

Idea: shorter codes for more frequent letters....

Find $E(x)$ $\ni$: (1) $\sum_{x \in \text{alphabet}} |E(x)| \, f(x)$ is minimized

(2) has prefix code property

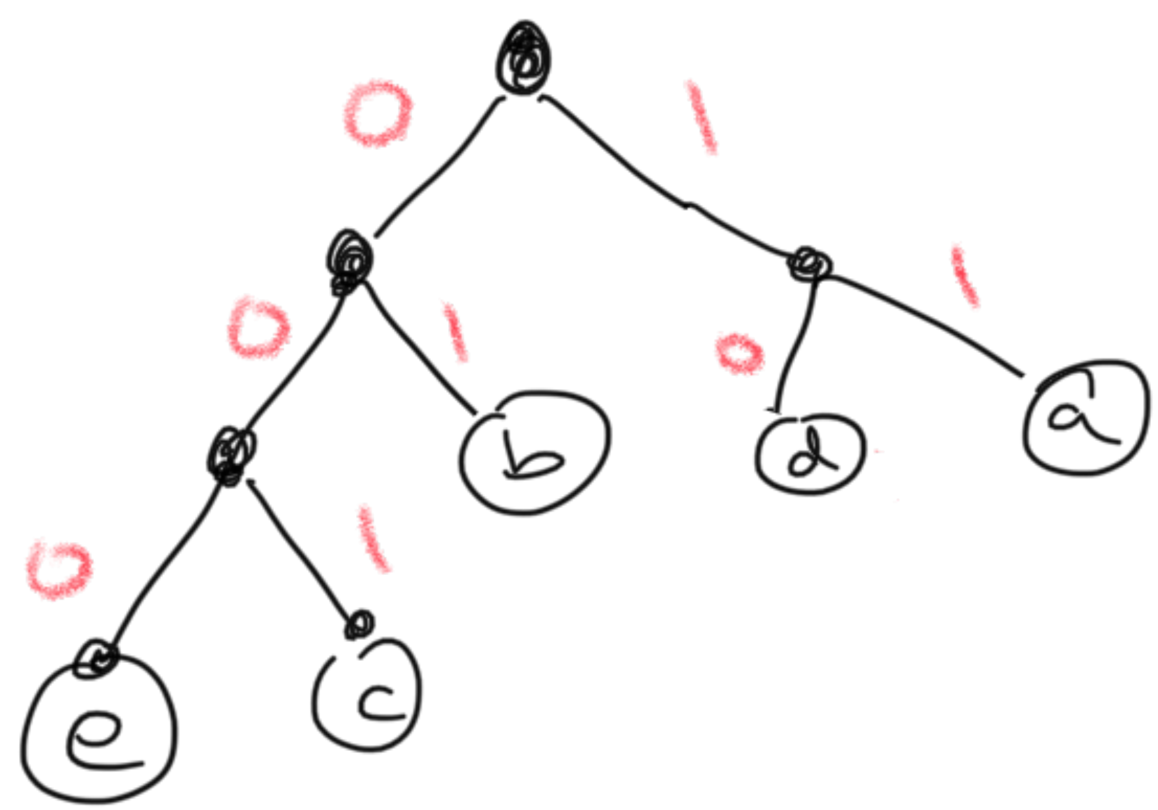
Codes as trees

binary tree: left $\swarrow$ 0, right $\searrow$ 1

path from root to a leaf : binary seq

a	b	c	d	e
11	01	001	10	000

$\Downarrow$



Prefix code $\Rightarrow$ each letter is at a leaf property

$$\text{depth}(\text{`}c\text{'}) = \text{length}(001)$$
$$= 3 = |E(\text{`}c\text{'})|$$

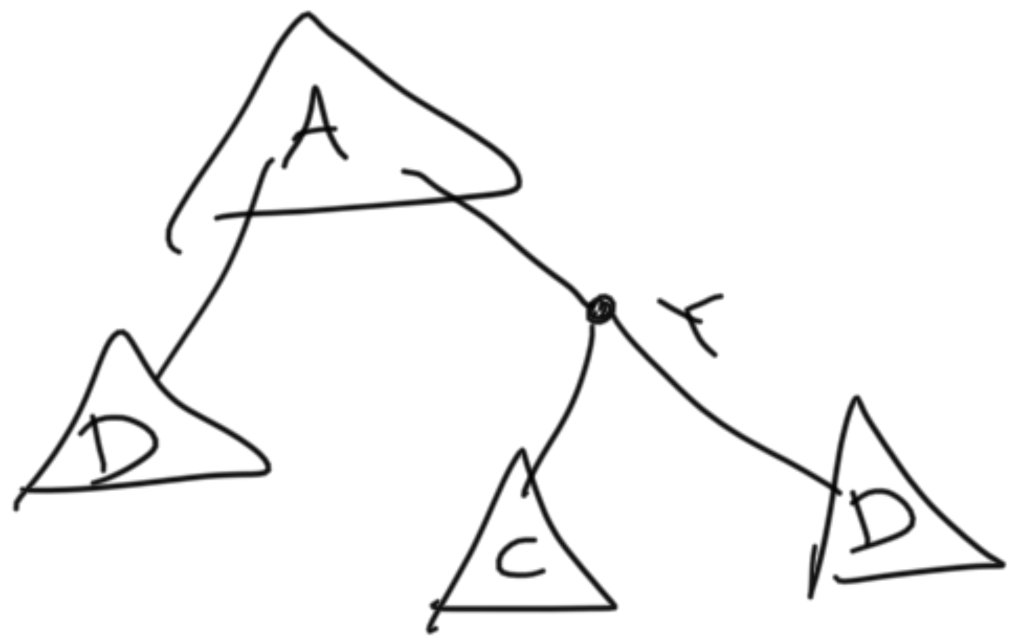
So want to put higher freq letters at lower depths

Full tree: 0 or 2 children for each node

I Optimal prefix code $\Rightarrow$ its tree has to be full

Pf

shrink tree...



$\underline{\text{II}}$ optimal prefix code $\Rightarrow$ if depth$(x) \leq$ depth(y) then $f(x) \geq f(y)$

$\underline{\text{Pf}}$: Else exchange x and y

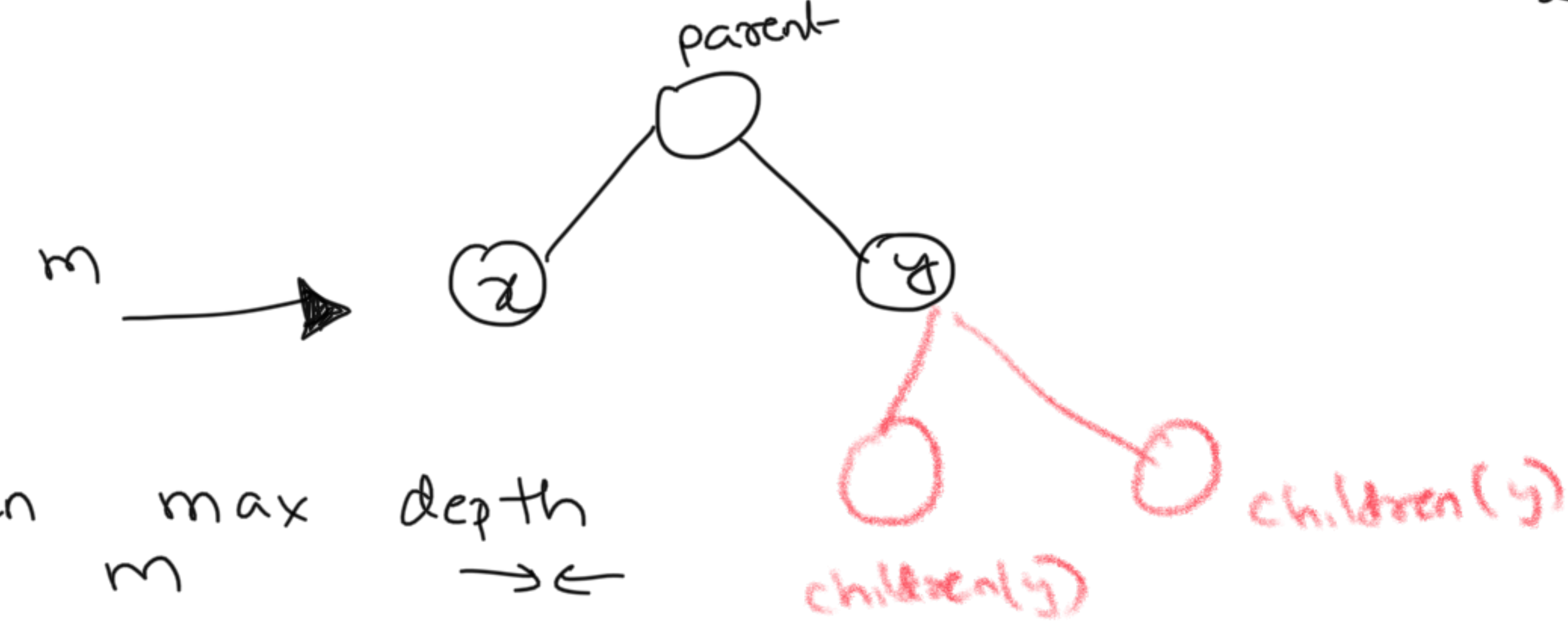
$\underline{\text{III}}$ Let max depth of tree $= m$ from optimal prefix code

Let x be a leaf @ depth m

Its sibling is also a leaf

Pf

since full,
x has
sibling y



Then max depth $> m$ $\rightarrow\leftarrow$

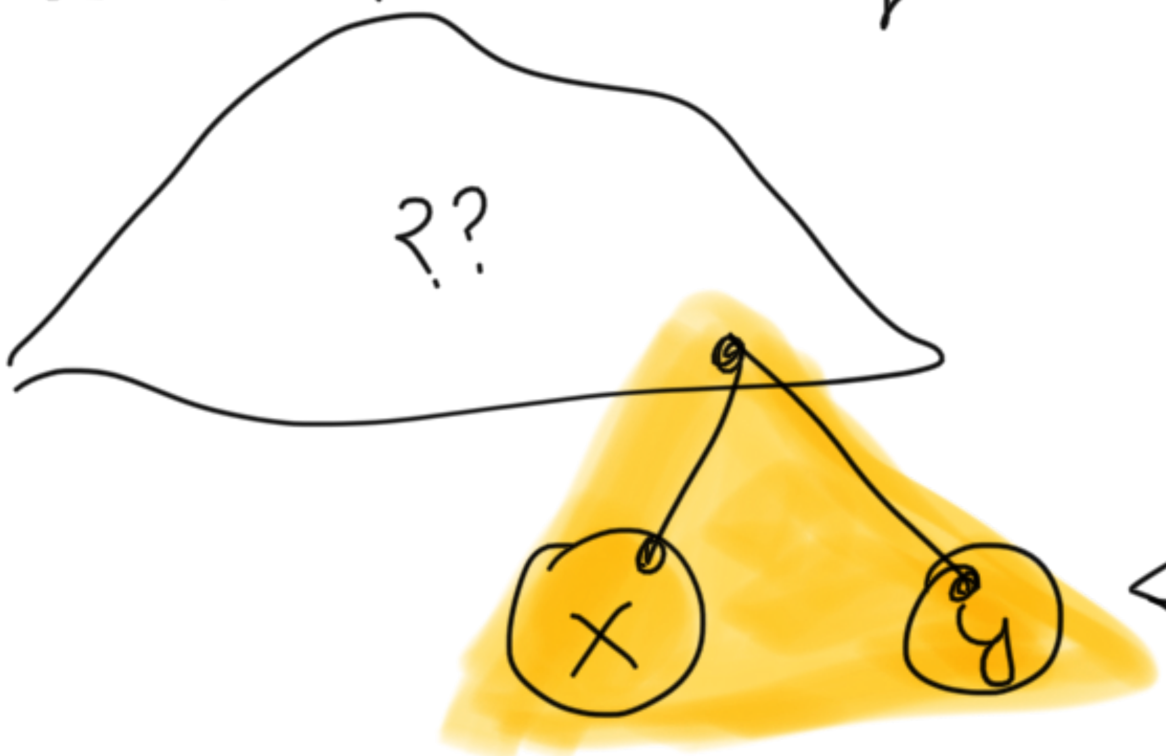
So Leaves @ max depths occurs in pairs

So leaves @ max depth $\rightarrow$ are lowest

Recursion (Huffman coding)

→ choose 2 lowest freq ... x, y ...

sibling leaves @ max depth



← treat as a unit

Alphabet A

create new alphabet letter xy

change alphabet

$A \to A' = A \setminus \{x, y\} \cup \{xy\}$

$f(xy) = f(x) + f(y)$

→ So recurse on A'

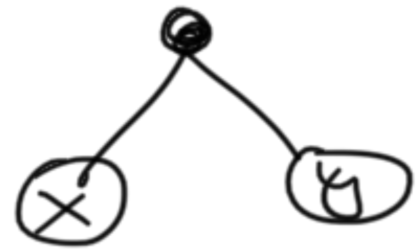
→ Base case : $|A| = 2 \Rightarrow E(x) = 0$

$E(y) = 1$

$A = \{x, y\}$

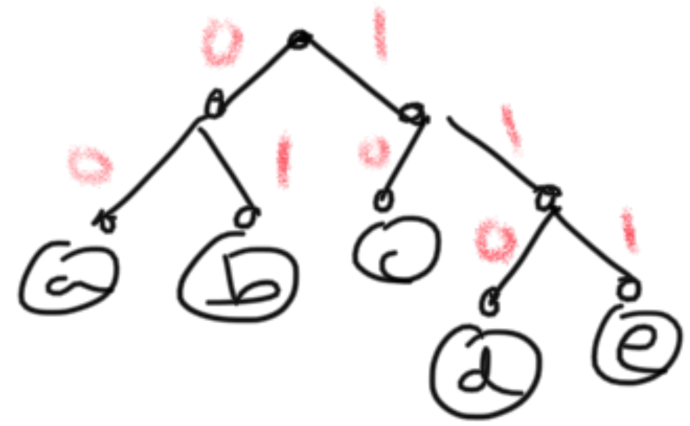
then go from A' $\longrightarrow$ A by

replacing xy with



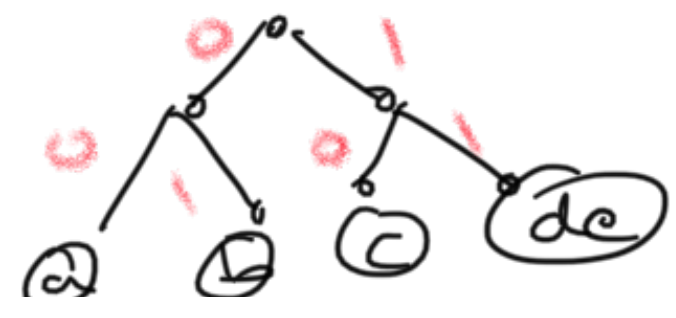
example

x	a	b	c	d	e
$f(x)$	0.32	0.25	0.2	0.18	0.05



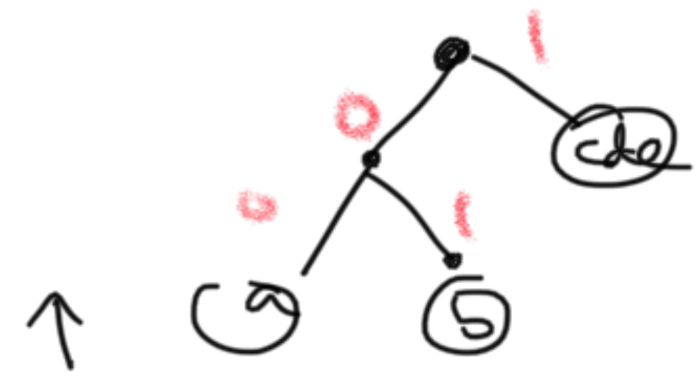
$\longrightarrow$

x	a	b	c	de
f	0.32	0.25	0.2	0.23



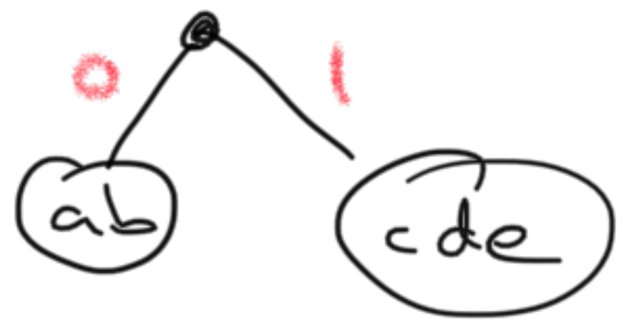
→

x	a	b	cde
f	0.32	0.25	0.43



→

x	ab	cde
f	0.57	0.43



why algo work?

→ Base case optimal

alphabet tree

→ Assume ok for k−1 letters $|A''|$ A' T'

→

$A\ ,\ T \rightsquigarrow A'\ ,\ T'$

x, y . . . . xy . . . . .

$$E = \sum f(\text{char})\,|E(\text{char})|$$

$$E' = \sum f(\text{char}')\,|E(\text{char}')|$$

$$E - E' = f(x)|E(x)| + f(y)|E(y)| - f(xy)|E(xy)|$$

$$= f(x)\,|E(x)| + f(y)\,|E(y)| - [f(x)+f(y)]\,|E(xy)|$$

$$= f(x)\,\ell + f(y)\,\ell - f(x)(\ell-1) - f(y)(\ell-1)$$

$$= f(x) + f(y)$$

$$= f(xy)$$

$|E(x)| = \ell$
$|E(y)| = \ell$
$|E(xy)| = \ell - 1$

* Suppose $\exists$ S (better tree) with $E_S > E_T$

↑
our tree

* x, y with lowest $f(x), f(y)$

must be @ max depth in S also

* wlog assume  in S also

(if not, move labels @ max depth leaves to make x and y as siblings)

* merge $x, y \longrightarrow xy$, $f(xy) = f(x) + f(y)$

to get S' (/ alphabet A' of $k-1$ letters)

S' / A' $\qquad \underbrace{T' / A'}_{\text{optimal}}$ $\qquad |A'| = k-1$

by induction

$$E_{S'} \geq E_{T'}$$

$$E_S = E_{S'} + f(xy) = E_{T'} + f(xy) = E_T$$

So T optimal as well

Implementation

* Extract lowest 2 freq
* merge, replace by combined freq

Each recursive step

current

* takes $O(|A|)$

# of recursive calls is $(|A| = k) - 1$

$= k-1$

* Store freq in array

* scan to find min 1, min 2 $\quad 2|A|$

$= O(|A|)$

$$k-1 + k-2 + \dots + 1 = \frac{k(k-1)}{2}$$

$$= O(k^2)$$

→ Can Do BETTER

→ maintain heap instead of array

→ finding min 1, min 2 → $O(\log |A|)$

+ insert new merged freq

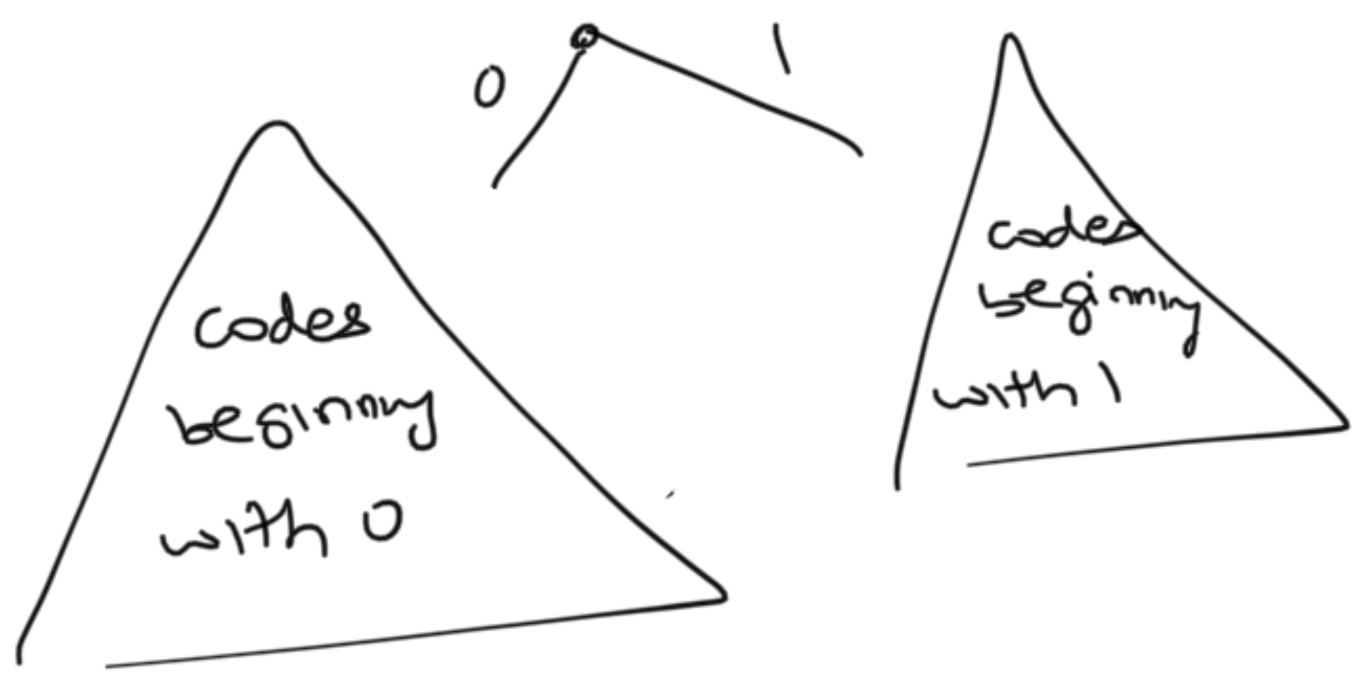
→ $O(k \log k)$

* Greedy: At each stage, choose 2 min freq to be leaves at that stage

Shannon - Fano: Divide and conquer approach
(1950)

$\Sigma f = F$

*



*

1 - subtree

0 - subtree

A_1

A_2

so $A = A_1 \sqcup A_2$, A_1 total freq $\approx A_2$ total freq

Recursively solve each partition

DOESN'T WORK

Fano's class: Huffman, grad student. After few years, came up with greedy sol.