binary codes: assign a string of 0's and 1's to every character in some alphabet.

is prefix-free if no code word is the prefix of another.

7-bit ASCII ✓
UTF-8 ✓
Morse code ✗

E: .
S: ..
represent using a binary
tree with characters at leaves
not necessarily
binary search trees; no
order to the characters
Given array \( f[1..n] \) of frequency counts. Character \( i \) appears \( f[i] \) times in some message, \( n \geq 2 \).

Want a binary code tree minimizing
\[
\sum_{i=1}^{n} \min \{ \text{depth}(i) \}
\]
Observation: optimal code tree is full (every node has 0 or 2 children)

2: Every node of max depth is a leaf.

⇒ 3: The max depth nodes have leaf siblings.
New backtracking strat:
Guess two siblings leaves.
Treat parent as a character representing both $x \lor y$.
Recurse over the $n-1$ remaining characters.
the course.

Single merged parent characters

threat them as a set of two least frequent:

Huffman (717)
Lemma: Let \( x \oplus y \) be two least frequent characters. There is an optimal tree with \( x \oplus y \) as siblings (i.e., they have max depth).

Proof: Let \( T \) be an optimal code tree. \( d := \text{depth at} \ T \). So there are sibling leaves \( a + b \) at depth \( d \).

\( x \) := least frequent character.

\( a \neq a \neq x \neq b \)....
$T' := \text{swap } a \leftrightarrow x \text{ in } T$

$\text{cost}(T') = \text{cost}(T) + \sum f[x] \cdot (\text{depth}(a) - \text{depth}(x))$

$- \sum f[a] \cdot (\text{depth}(x) - \text{depth}(a))$

$= \text{cost}(T) + \left( \sum f[x] - \sum f[a] \right) \cdot \left( \text{depth}(a) - \text{depth}(x) \right)$

$\leq \text{cost}(T) + 0 = \text{cost}(T)$
If $xy \neq yx$, swap them too.

for $T''$

$\text{cost}(T'') \leq \text{cost}(T)$

$T''$

$T$ was optimal, so $T''$ must be as well!
Theorem: Huffman codes are optimal.

If $n = 2$, yes.
o.w. assume $f[1] + f[2]$ have least frequencies.

$T_i$: any code tree over \(f[1, \ldots, n]\) with 1+2 as siblings.

$T'_i$: $T \setminus \{1, 2\}$.

\( T' \) is a code tree for \( f(3, n+1) \).

\[
\text{cost}(T) = \sum_{i=1}^{n+1} f[i] \cdot \text{depth}_T(i)
\]

\[
= \sum_{i=3}^{n+1} f[i] \cdot \text{depth}_T(i) + f[1] \cdot \text{depth}_T(1) + f[2] \cdot \text{depth}_T(2)
- f[n+1] \cdot \text{depth}(n+1)
\]

\[
= \text{cost}(T') + f[1] \cdot \text{depth}(1) + f[2] \cdot \text{depth}(2)
- f[n+1] \cdot \text{depth}(n+1)
\]

\[
\]
\[
\text{cost}(T') + \sum \left( f[1] + f[2] \right)
\]

\[
\text{(depth}_{T})^{(1)}
\]

Want to minimize \text{cost}(T)

which we do by I.H.
$O(n \log n)$ to build using a priority queue (min heap)