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

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

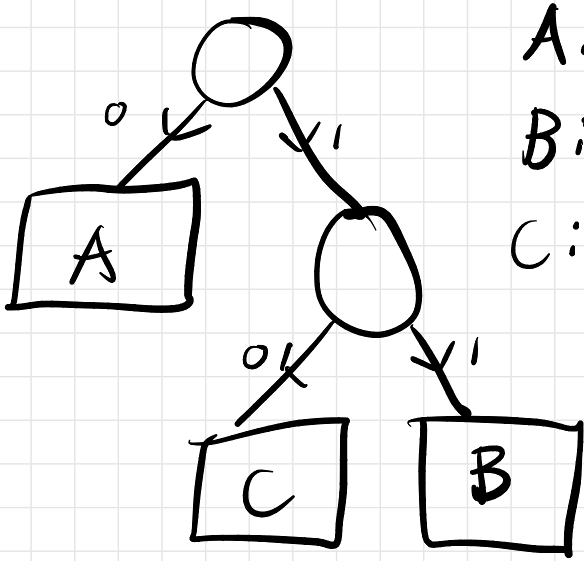
7-bit ASCII ✓

UTF-8 ✓

Morse code ✗

E: .

S: ...



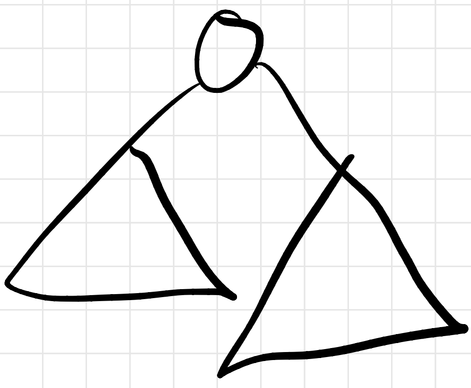
A: 0
B: 11
C: 10

represent using a binary tree with characters at leaves

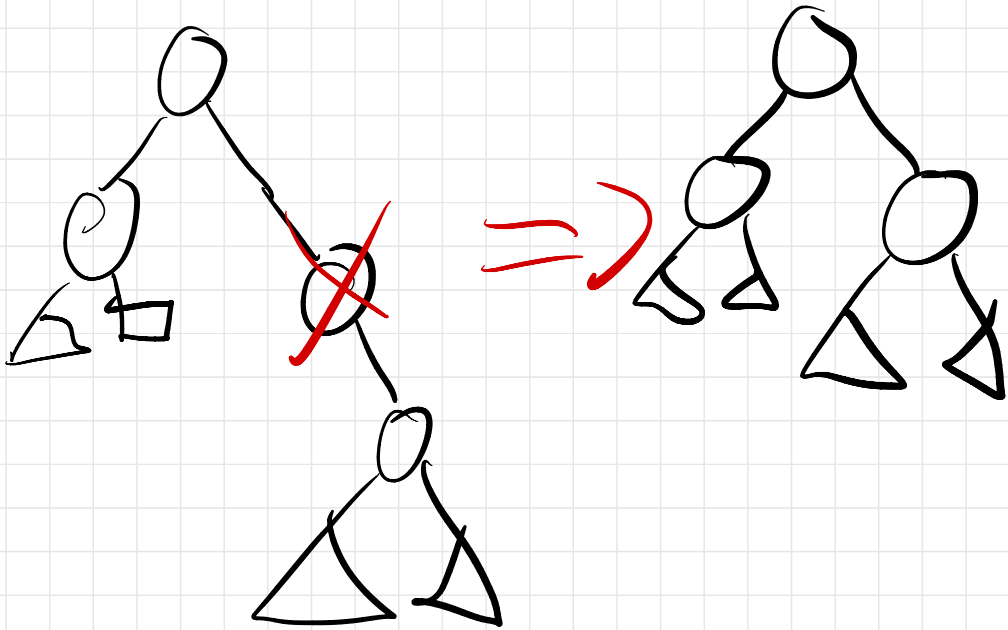
not ~~nec~~ necessarily binary search trees; no order to the characters

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 $\sum_{i=1}^{n} f[i] \cdot \text{depth}(i)$

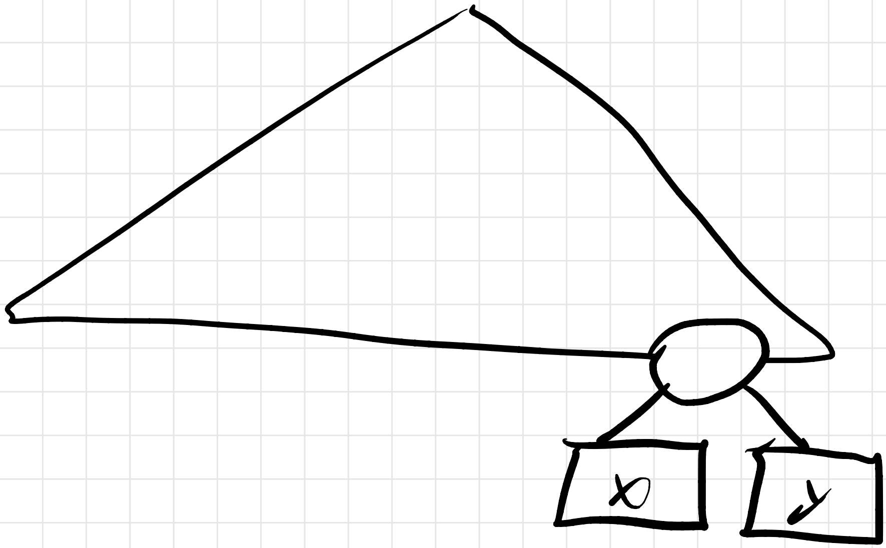


Observation 1: optimal code tree is <u>full</u> (every node has 0 or 2 children)



2: Every node of max depth is a leaf.

⇒ 3: The max depth nodes have leaf siblings.



New backtracking strat:

Guess two siblings leaves. $x+y$
Treat parent as a character representing both $x+y$.
Recurse over the $n-1$ remaining characters.

Huffman['51]:

- set two least frequent characters as sibling leaves
  - treat them as a single merged/parent character & recurse.

Σ:	A	B	C	D	E	F
f:	45	13	12	16	9	5

CAFE...

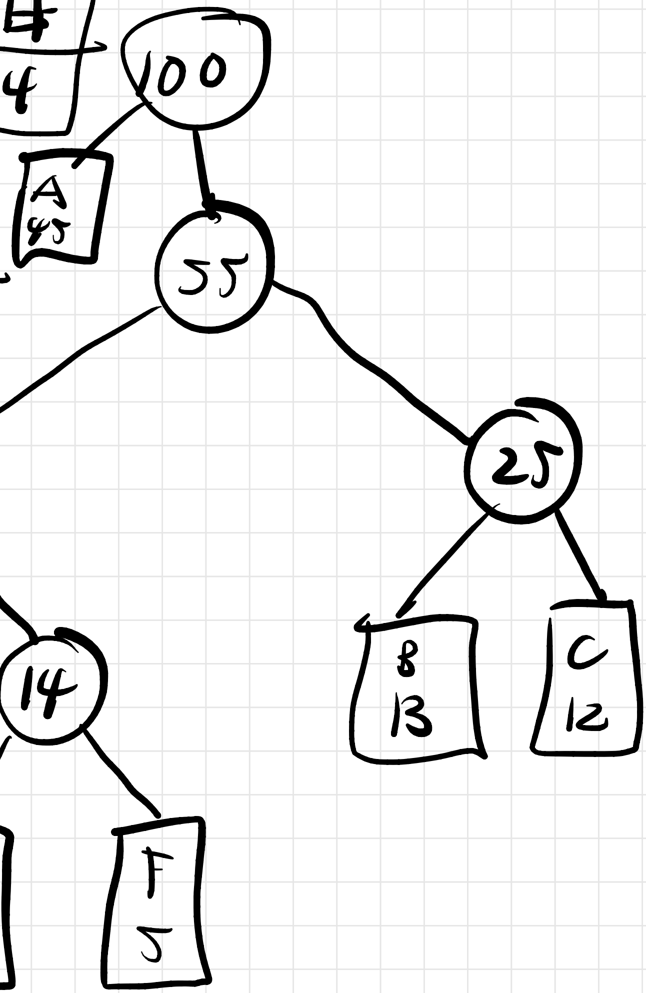
111 0 1011 1010 ...

A	B	C	D	EF
45	13	12	16	14

A	BC	D	EF
45	25	16	14

A	BC	DEF
45	25	30

A	BCDEF
45	55



A: 0 C: 111 E: 1010

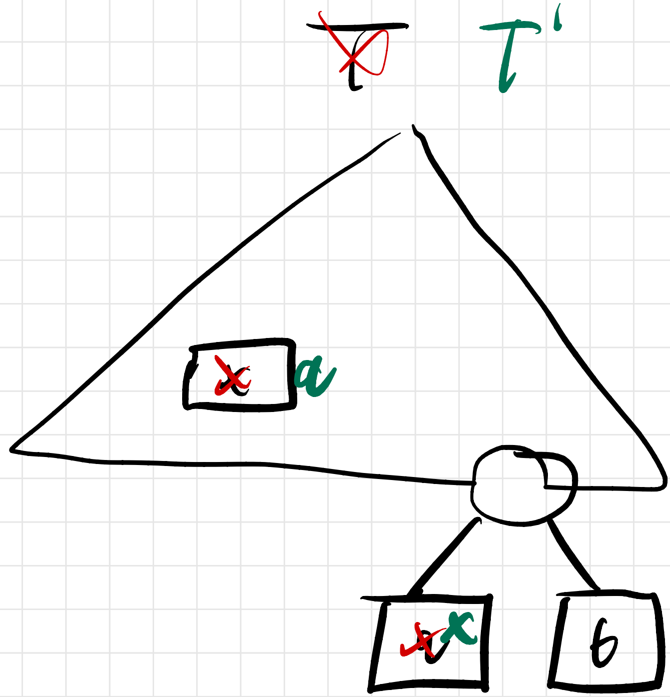
B: 110 D: 100 F: 1011

Lemma: Let x & y be two least frequent characters. There is an optimal tree with x & y as siblings (& they have max depth).

Proof: Let T be an optimal code tree. $d :=$ depth of T. So there are sibling leaves a & b at depth d.

$x :=$ least frequent character

if $x \neq a$ & $x \neq b$ ...

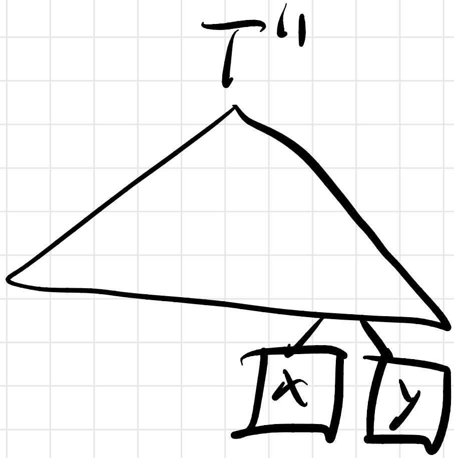


$T' :=$ swap a + x in T

$$\begin{aligned}
\text{cost}(T') &= \text{cost}(T) \\
&\quad + f[x]\cdot(\text{depth}_T(a) - \text{depth}_T(x)) \\
&\quad - f[a]\cdot(\text{depth}_T(a) - \text{depth}_T(x)) \\
&= \text{cost}(T) + (f[x] - f[a])\cdot \\
&\qquad (\text{depth}_T(a) - \text{depth}_T(x)) \\
&\leq \text{cost}(T) + 0 = \text{cost}(T)
\end{aligned}$$

If $b \neq y$, swap them too for T''.

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



T was optimal, so T'' must be as well!

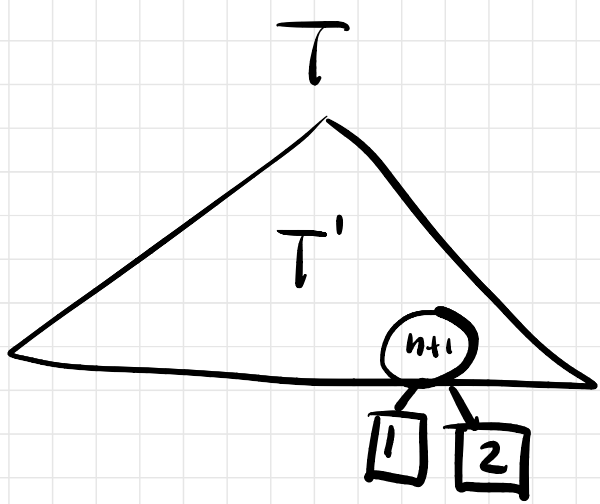
Theorem: Huffman codes are optimal.

If $n=2$, yes.

o.w. assume $f[1]$ & $f[2]$ have least frequencies.

$T :=$ any code tree over $f[1..n]$ with 1 & 2 as siblings.

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



Treat parent of 1 & 2 as character $n+1$. $f[n+1] := f[1] + f[2]$

T' is a code tree for $f[3 .. n+1]$.

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

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

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

$$= \mathrm{cost}(T') + f[1] + f[2] + (f[1] + f[2] - f[n+1]) \cdot ?$$

$$= \mathrm{cost}(T') + \underbrace{f[1] + f[2]}_{(\mathrm{depth}_T(1) - 1)}$$

Want to minimize $\mathrm{cost}(T')$ which we do by I.H.

O(n log n) to build using a priority queue (min heap)

```
BuildHuffman(f[1..n]):
  for i ← 1 to n
      L[i] ← 0; R[i] ← 0
      Insert(i, f[i])
  for i ← n+1 to 2n − 1
      x ← ExtractMin( )          ⟨⟨find two rarest symbols⟩⟩
      y ← ExtractMin( )
      f[i] ← f[x] + f[y]         ⟨⟨merge into a new symbol⟩⟩
      Insert(i, f[i])
      L[i] ← x; P[x] ← i         ⟨⟨update tree pointers⟩⟩
      R[i] ← y; P[y] ← i
  P[2n − 1] ← 0
```

```
HuffmanEncode(A[1..k]):
  m ← 1
  for i ← 1 to k
      HuffmanEncodeOne(A[i])

HuffmanEncodeOne(x):
  if x < 2n − 1
      HuffmanEncodeOne(P[x])
      if x = L[P[x]]
          B[m] ← 0
      else
          B[m] ← 1
      m ← m + 1
```

```
HuffmanDecode(B[1..m]):
  k ← 1
  v ← 2n − 1
  for i ← 1 to m
      if B[i] = 0
          v ← L[v]
      else
          v ← R[v]
      if L[v] = 0
          A[k] ← v
          k ← k + 1
          v ← 2n − 1
```