Greedy Day 2 2/24/22

1. Planting Trees
2. Spreading News
3. Huffman Coding
4. Welcome

↓ 4 more specific solutions which use a greedy alg.

Planting Trees (kattis: plantingtrees)

n trees to plant
ith tree takes a_i days to mature
Plant 1 tree a day.
Goal: Min # days for all trees to mature

n = 4 3, 8, 1, 4

8	4	3	1
Day 1	Day 2	Day 3	Day 4
9	6	6	5

ans 9

Greedy: Plant trees in order from longest to mature to shortest to mature.

exchange argument! ⇒ anything that exchanges the longest ~~tree~~ tree only potentially gets worse.

Spreading News

Can't try all orders of calling subordinates.

Which order is the best?

⇒ Recursively call on each subtree

⇒ Store all answers from recursive calls

⇒ IDENTICAL PROB PLANTING TREES!

3 | 4 | 2 | 5

⇒ 5 | 4 | 3 | 2

1 2 3 4

6 6 6 6

6

Huffman Coding

Optimal Variable Length Coding Scheme

	fixed	freq
a	000	200
b	001	10
c	010	13
d	011	47
e	100	18
f	101	63
g	110	24
h	111	25
		400

$3 \times 400 = 1200$ bits

What if some codes shorter, others longer? Can we save space?

Problem

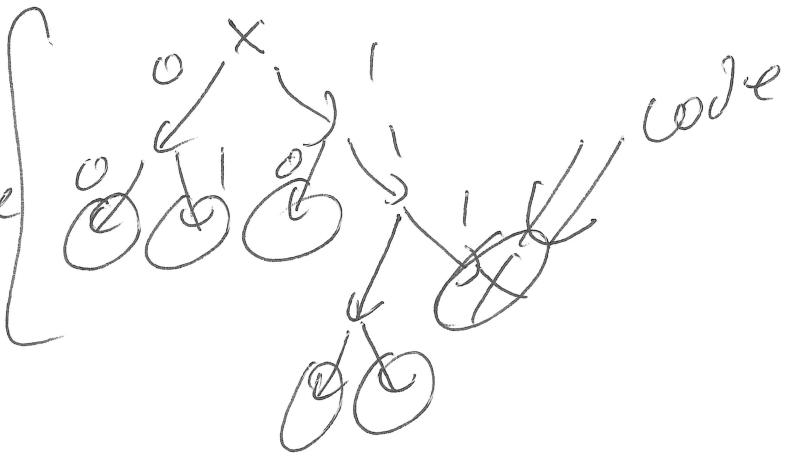
When do I know a code is done if they all aren't the same length?

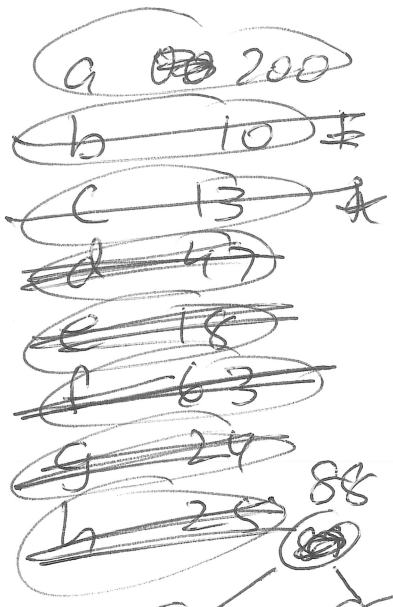
↳ SOL: Not allowed to have codes x and y if x is a prefix of y.

01, 0110 → [Prefix free]

Prefix of one code can't be another code

elegant way of saying prefix free is that ALL leaf nodes binary tree to correspond to codes!





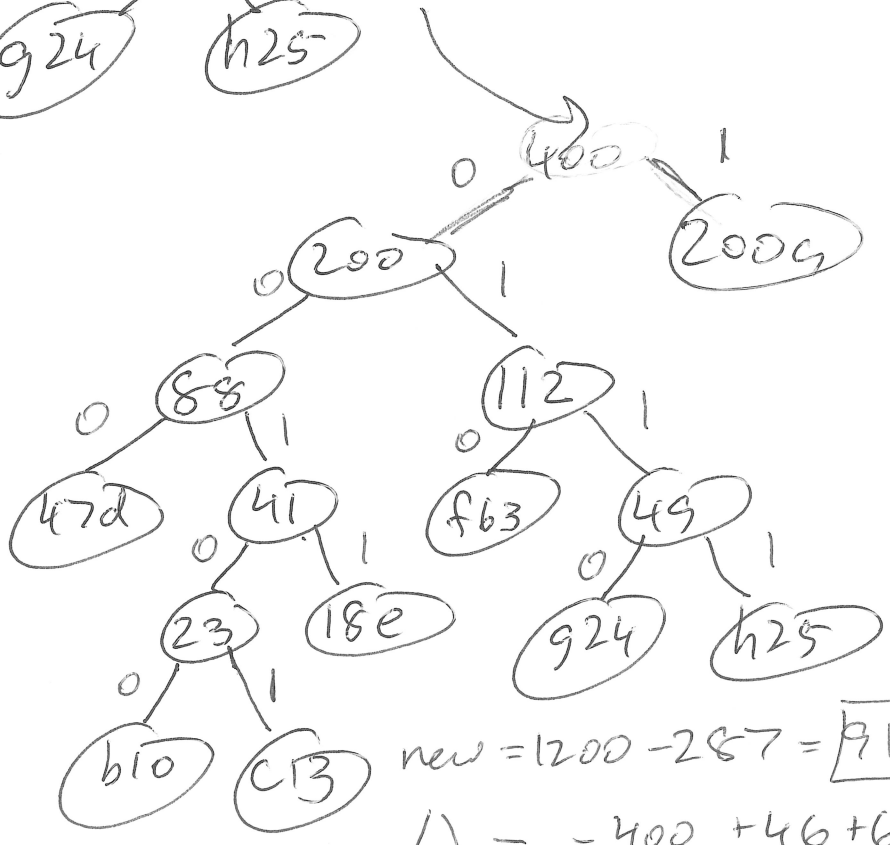
Algorithm

each char+freq is a tree on its own, so we start w/n trees size 1

Add each tree to a priority queue sorted by sum of freq.

```
While ( # trees > 1 ) {
    tree1 = pq.poll()
    tree2 = pq.poll()
    treeNew = merge (tree1, tree2)
    pq.offer (treeNew)
}
```

113



a	1	(-2)
b	00100	(+2)
c	00101	(+2)
d	000	(0)
e	0011	(+1)
f	010	(0)
g	0110	(+1)
h	0111	(+1)

new = 1200 − 287 = **913**

Δ = −400 + 46 + 67 = **−287**

bits used = 1200 − 2(200) + 2(10) + 2(13) + 1(18) + 1(24) + 1(25)

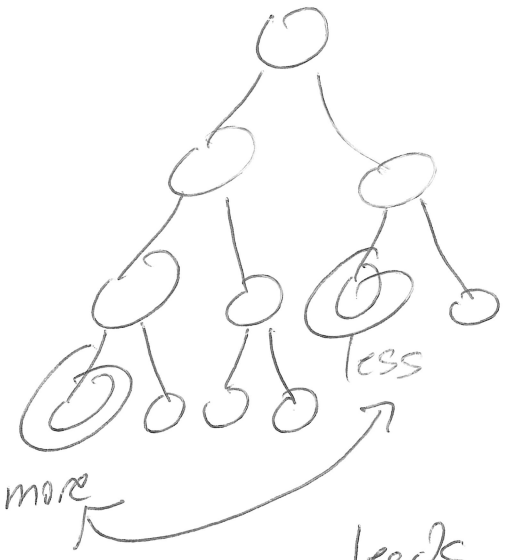
Why greedy?

I don't consider all trees. I greedily pick the 2 "smallest" to merge.

Why is this optimal?

My tree

Competing Tree



exchanging leads to less cost.

Welcome Party

Hybrid Problem "two dimensions"

If you brute force both dimensions, your algorithm will take too long. But, if you brute force one dimension, then you can make an observation to run a greedy on the other dimension.

mask = 27 = 11011 (16 8 4 2 1; e d b a) { mask represents which last name clubs we have

Idea: Brute force all last name clubs (possible # subsets = 2^{18} since all last names are 'A'-'R'.

Once we know the last name clubs loop through all the names (greedy) + place anyone w/a last name out of "range" into the corresponding first name club.

Run-Time = 2^{18} (possible last letters) $\times$ 300 (n names) $\sim$ 75 million