Name: Grishma Maheshbhai Thumar

Student ID: 922950012

Course: CSC 510, Spring 2025

Project: Homework 5

1. Longest Common Subsequence (LCS)

s_1 = T C G G C G T A G A

s_2 = C A A G C A T A T G G

DP Table for it is:

Each cell c[i,j] is the length of the LCS of s_1[1…i] and s_2[1…j].

		C	A	A	G	C	A	T	A	T	G	G
	0	0	0	0	0	0	0	0	0	0	0	0
T	0	0	0	0	0	0	0	1	1	1	1	1
C	0	1	1	1	1	1	1	1	1	1	1	1
G	0	1	1	1	2	2	2	2	2	2	2	2
G	0	1	1	1	2	2	2	2	2	2	3	3
C	0	1	1	1	2	3	3	3	3	3	3	3
G	0	1	1	1	2	3	3	3	3	3	4	4
T	0	1	1	1	2	3	3	4	4	4	4	4
A	0	1	2	2	2	3	4	4	5	5	5	5
G	0	1	2	2	3	3	4	4	5	5	6	6
A	0	1	2	3	3	3	4	4	5	5	6	6

Reconstruction steps (starting at i = 10, j = 11):

c[10,11]=6; s_1[10]=A ≠ s_2[11]=G; since c[9,11]=6 ≥ c[10,10]=6, move **up** to (9,11).

c[9,11]=6; s_1[9]=G = s_2[11]=G → **match** "G"; move to (8,10).

c[8,10]=5; s_1[8]=A ≠ s_2[10]=G; since c[8,9]=5 ≥ c[7,10]=4, move **left** to (8,9).

c[8,9]=5; s_1[8]=A ≠ s_2[9]=T; since c[8,8]=5 ≥ c[7,9]=4, move **left** to (8,8).

$c[8,8]=5$; $s_1[8]=A = s_2[8]=A \rightarrow$ **match** "A"; move to (7,7).

$c[7,7]=4$; $s_1[7]=T = s_2[7]=T \rightarrow$ **match** "T"; move to (6,6).

$c[6,6]=3$; $s_1[6]=G = s_2[6]=A$? No; compare $c[5,6]=2$ vs $c[6,5]=3 \rightarrow$ **left** to (6,5).

$c[6,5]=3$; $s_1[6]=G = s_2[5]=C$? No; compare $c[5,5]=2$ vs $c[6,4]=2 \rightarrow$ **up** to (5,5).

$c[5,5]=2$; $s_1[5]=C = s_2[5]=C \rightarrow$ **match** "C"; move to (4,4).

$c[4,4]=2$; $s_1[4]=G = s_2[4]=G \rightarrow$ **match** "G"; move to (3,3).

$c[3,3]=1$; $s_1[3]=G$… no match $\rightarrow$ move up/left until (2,1), then match the "C" at (2,1).

Reversing the matched characters yields one LCS of length 6.

"C G C T A G"

2. Optimality of the Cashier's (Greedy) Algorithm for {1, a, ab}

Let an optimal solution use x_1 1-cent coins, x_a a-cent coins, and $x_{(ab)}$ (ab)-cent coins.

(a)

Proposition 2.1.

In any optimal solution, the number of 1-cent coins satisfies

$$\#(\text{1-cent}) \leq a-1.$$

Proof (exchange argument).

If an optimal solution ever used $\geq$ a pennies, you can replace **a** of those 1-cent coins by a single a-cent coin. That swap:

- reduces the total number of coins by (a – 1), and
- still makes exactly the same total value.

This contradicts optimality, so you cannot have a or more pennies.

(b)

Proposition 3.1.

In any optimal solution, the number of a-cent coins satisfies

$$\#(\text{a-cent}) \leq b-1.$$

Proof (exchange argument).

If an optimal solution ever used $\geq$ b of the a-cent coins, you can replace b of those a-cent coins by a single (ab)-cent coin. That swap:

- reduces the coin count by (b – 1), and
- keeps the total value unchanged.

Hence you can never have b or more a-cent coins in an optimal solution.

(c)

Lemma 4.3 (contrapositive style).

For any amount $C \geq ab$, *every optimal solution must include at least one ab-cent coin.*

Proof.

We prove the contrapositive:

If an optimal solution contains **no** ab-cent coins, then C<abC < abC<ab.

Under that assumption, it uses only 1- and a-cent coins. By parts (a) and (b):

$$\#(\text{1-cent}) \leq a-1, \qquad \#(\text{a-cent}) \leq b-1.$$

Thus

$$C = 1\cdot\#(1) + a\cdot\#(a) \leq (a-1) + a\,(b-1) = ab - 1 < ab.$$

This contradiction shows that **any** optimal solution for $C \geq ab$ must indeed contain an ab-cent coin.

3. Which of these systems is always greedy-optimal?

After testing each (and in cases (a) and (d) found counterexamples by comparing the greedy count to the true minimum via dynamic programming).

(a) {1, 13, 60}

- **Claim:** *Not* always optimal.
- **Counterexample:** Make change for **65** cents.
  - **Greedy:** take 60 → 5 remaining → take 5×1 → **6 coins** total.
  - **Optimal:** take 5×13 → **5 coins** total.
  - ⇒ Greedy uses more coins (6 > 5), so fails at C = 65.

(b) {1, 14, 98}

- **Claim:** Always optimal.
- **Reason:** 14 is a multiple of 1, and 98 is a multiple of 14 (98 = 14·7).
- **Exchange-argument sketch (exactly like the 3-coin proof):**
  1. No optimal solution can use ≥ 14 pennies (swap 14 × 1 ¢ → one 14 ¢ coin).

2. No optimal solution can use ≥ 7 of the 14 ¢ coins (swap 7 × 14 ¢ → one 98 ¢ coin).
3. Thus whenever you need ≥ 98 ¢, an 98-¢ coin appears (contrapositive).

- That exactly matches the pattern proven in class for US coinage .

(c) {1, 9, 63, 315}

- **Claim:** Always optimal.
- **Reason:** Each denomination divides the next:
  - $9 = 9 \cdot 1$
  - $63 = 7 \cdot 9$
  - $315 = 5 \cdot 63$
- We can repeat the same three-step exchange argument twice (first on 1,9,63, then on 1,9,63,315) to show the greedy choice is always part of an optimal solution.

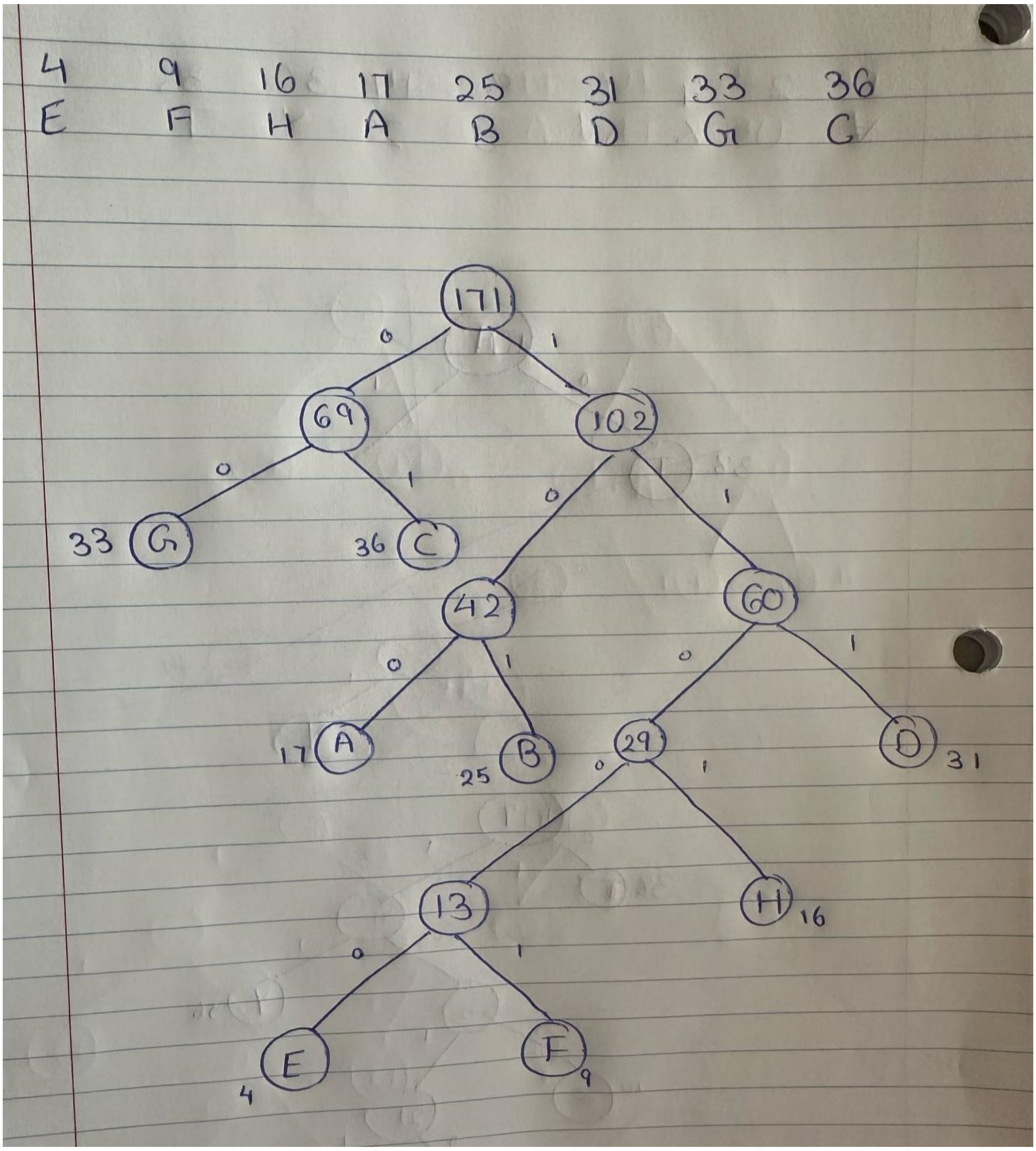
(d) {1, 7, 19, 45}

- **Claim:** *Not* always optimal.
- **Counterexample:** Make change for **57** cents.
  - **Greedy:** take 45 → 12 remaining → take 7 → 5 remaining → take 5×1 → **7 coins** total.
  - **Optimal:** take 3×19 → **3 coins** total.
  - ⇒ Greedy uses more (7 > 3), so fails at C = 57.

4. Huffman Codes

By repeatedly merging the two least-frequent nodes and placing the **heavier** of the two as the **left** child.

(a) Frequencies A 17, B 25, C 36, D 31, E 4, F 9, G 33, H 16



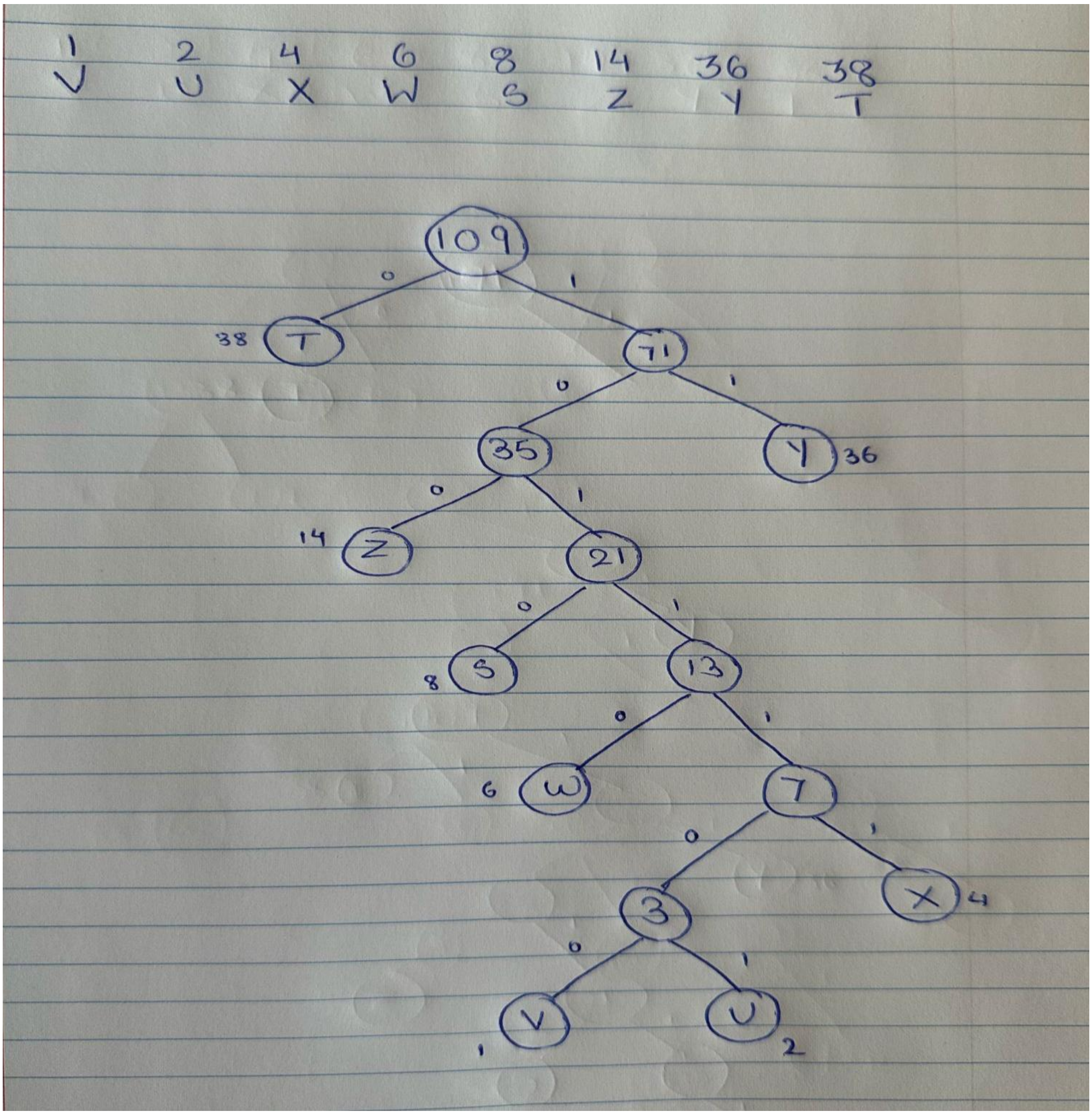
Character	**Frequency**	**Code**
C	36	01
G	33	00
D	31	111
B	25	101
A	17	100

Character	Frequency	Code
H	16	1101
F	9	11001
E	4	11000

Total encoded length =

$36 \cdot 2 + 33 \cdot 2 + 31 \cdot 3 + 25 \cdot 3 + 17 \cdot 3 + 16 \cdot 4 + 9 \cdot 5 + 4 \cdot 5 = 486$ bits.

(b) Frequencies S 8, T 38, U 2, V 1, W 6, X 4, Y 36, Z 14



Character	Frequency	Code
T	38	0
Y	36	11
Z	14	100
S	8	1010
W	6	10110
X	4	101111
U	2	1011101
V	1	1011100

Total encoded length =

$38 \cdot 1 + 36 \cdot 2 + 14 \cdot 3 + 8 \cdot 4 + 6 \cdot 5 + 4 \cdot 6 + 2 \cdot 7 + 1 \cdot 7 = 259$ bits.