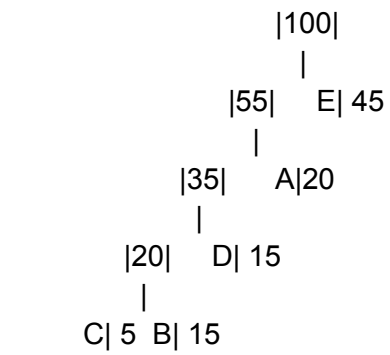
# Lab 5

`Let A = {a/20, b/15, c/5, d/15, e/45}`

Frequency:

A = 20
B = 15
C = 5
D = 15
E = 45

Min frequency -> 5 + 15 = 20 (merge the two smallest)
Merge new node with next lowest (d) -> 20+15 -> 35
Merge with A -> 35+20 -> 55
Merge with last -> 55 + 45 -> 100




e=0
a=10
d=110
b=1110
c=1111

Average = (0.45 * 1) + (0.2 * 2)+(0.15 * 3)+(0.15 * 4)+(0.05 * 4) =2.1 bits/symbol

2. `string: BABAABAAA.`

`A = 65 B =66 Black = input, Red = Output, Green = newDictionaryEntry`

1. B - \N - \N
2. BA - 66(B) - 256 = “BA”
3. A - \N - \N
4. AB - 65(A) - 257 = “AB”
5. BA - 256(BA) -/N
6. AB - \N -\N
7. BAA - 257(AB) - 258 = “BAA”
8. A - \N - \N
9. AA - 65(A) - 259 = “AA”
10. A - 259(AA) - \N

Sequence = [66,65,256,257,258,259]

3. Delta compression works by storing only the differences between successive versions of data instead of storing the entire dataset.

**How It Works**

1. The first version of a file is stored completely.
2. Subsequent versions store only the changes between the old and new versions.
3. When retrieving a full version, the base version is reconstructed along with the stored differences.

**Application**

One common use case is version control systems like Git, where only the differences between commits are stored, significantly reducing storage requirements.