Huffman Codes and Text Compression

- Representing letters →

A	B	C	D	∅
000	001	010	011	100

if we represent these with binary code how many digits should we use?

↳ encoding table

2^m > 5

↳ use 3 digits to represent 5 chars

ABCDABCBBA = m

↳ 000 001 010 011 000 ---- = 3×m bits

A B C D A

d = number of chars

n = number of items

total comp = n + d log d

Variable Sized Encoding →

A → 0

B → 10 → 0 ile başlayamaz

C → 110 → 10 ve 0 ile başlayamaz

D → 111 → 10, 110, 0 ile başlayamaz

ABC CAABA

0 10 110 110 0 0 100

ex:

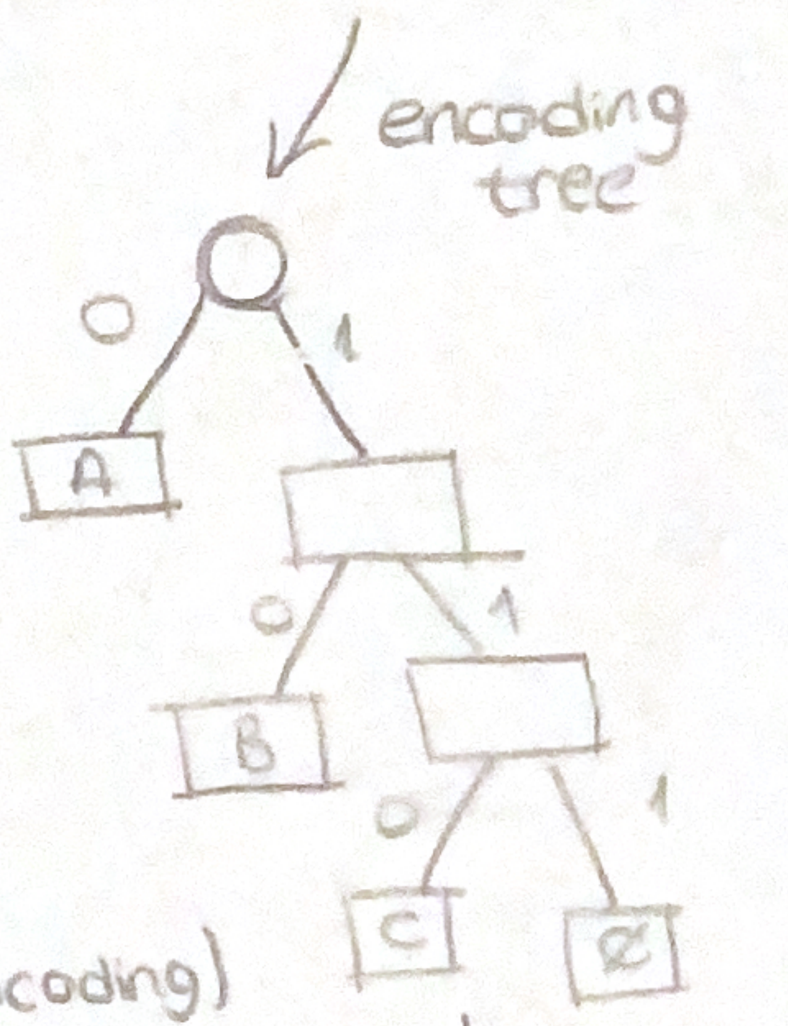
A	B	C	D	—
035	0.1	0.2	0.2	0.15

} letter frequencies

Fixed Encoding

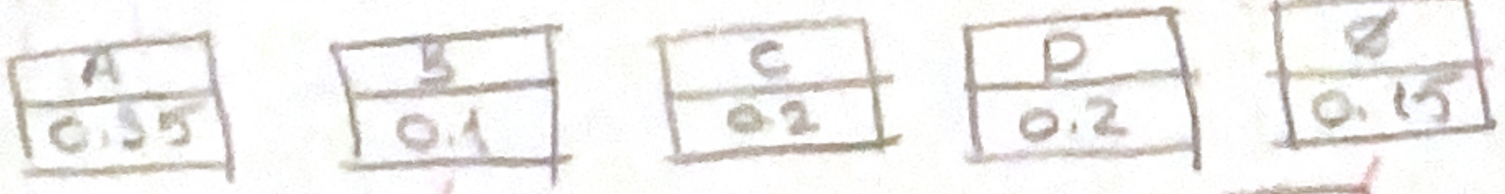
100 letter text → we need 3 bits

3×100 = 300 bits



If we can build up a variable encoding tree we can decrease the space (Huffman encoding)

- En küçük frecansları recursive olarak birleştir.



* complexity of construction d log d =

A → 11

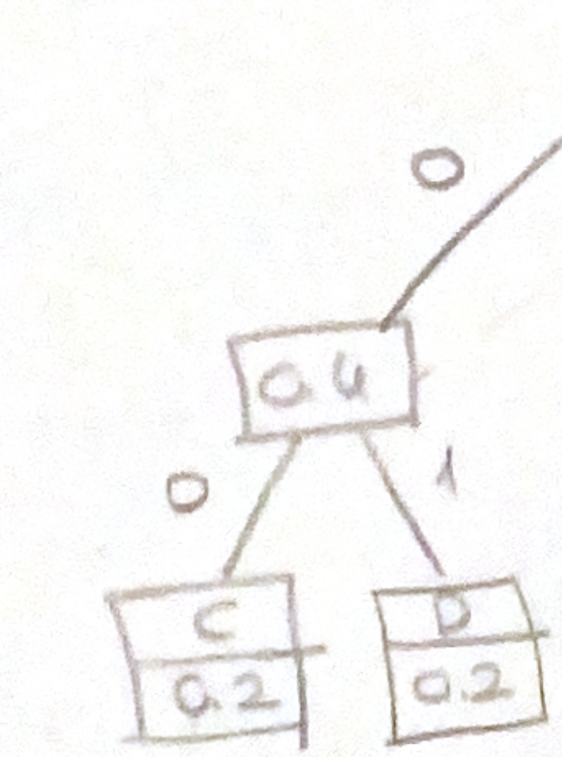
B → 100

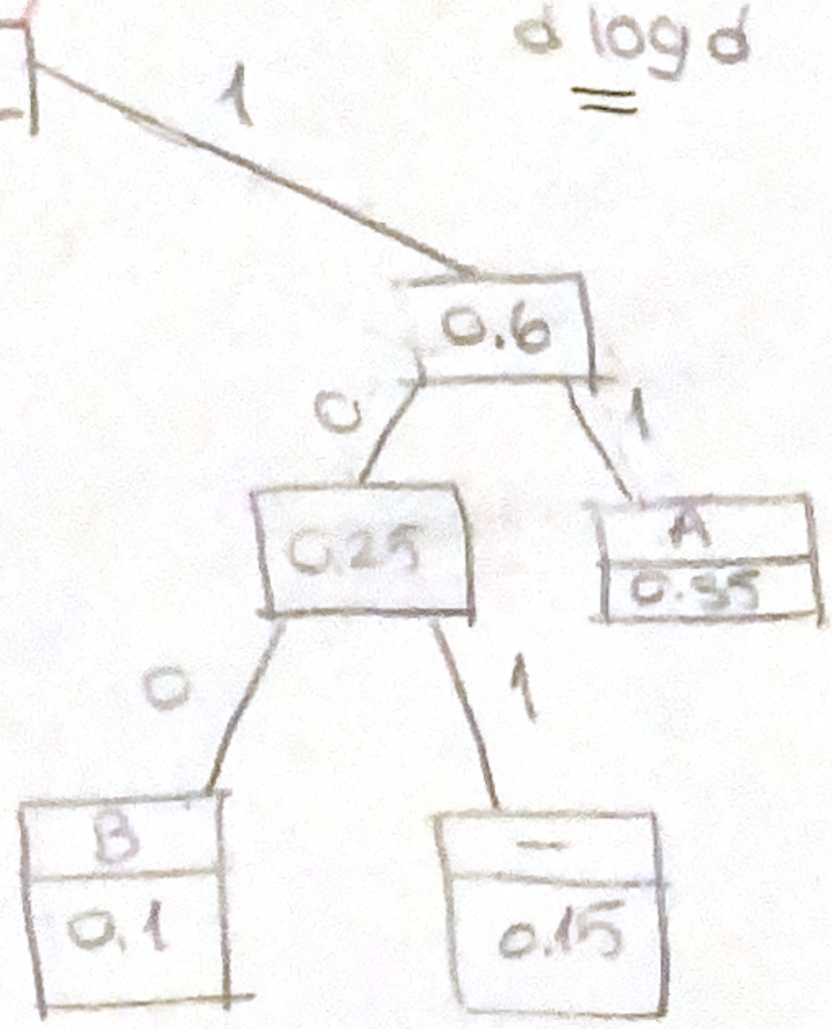
C → 00

D → 01

∅ → 101

100 letter → 35×2 A, 10×3 B, 20×2 C, 20×2 D, 15×3 ∅ → 225 bits (25% improvement)



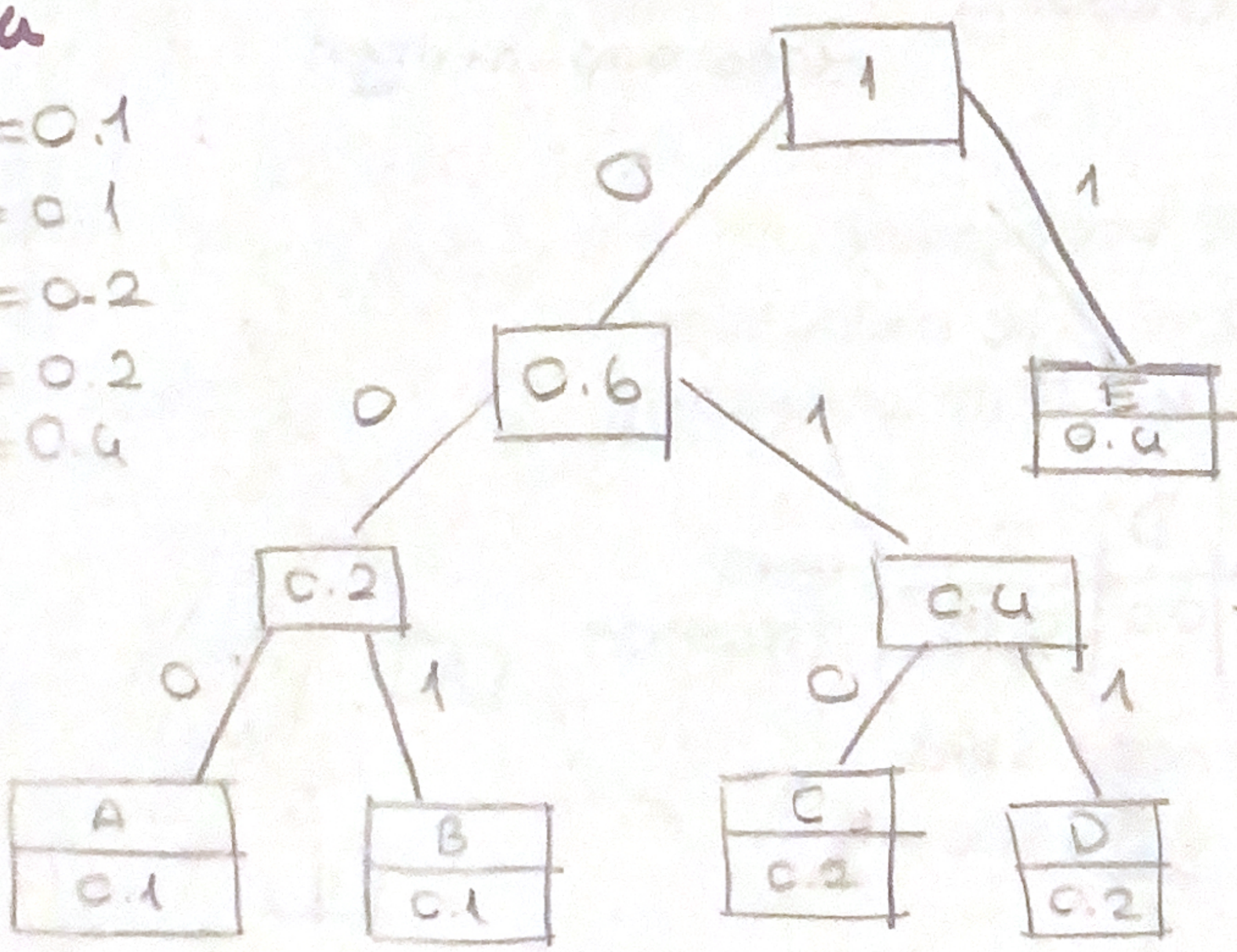


Huffman's Algorithm

1. Initialize n one node trees and label them with letters and ~~sequences~~. frequences
2. Repeat this step until every node is combined in a single tree. Combine them and make left & right subtree of a root which has the combined frequency.

ex.

A = 0.1
B = 0.1
C = 0.2
D = 0.2
E = 0.4



A = 000
B = 001
C = 010
D = 011
E = 1

100 letters

A = 3×10
B = 3×10
C = 3×20
D = 3×20
E = 1×40

220 bits

Fixed Encoding → 3 × 100 = 300 bits
Variable Encoding → 220 bits