Huffman Encoding

Huffman encode letter so that it has minimum data

- Most frequent data ⟶ Least number of bits
- No ambiguity & Each encoding must be unique

Example

a occurs 50 times
b occurs 100 times
c " 200 "
d " 250 "
e " 300 "

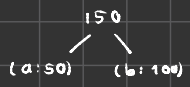
} key, value pairs —Insert into piority Q→ (a : 50)(b : 100)(c : 200)(d : 250)(e : 300)

Algorithm

① Pop the first two data

Piority Q : (c : 200)(d : 250)(e : 300)

② Sum number occurance as the parent of the subtree



③ Insert the root back into piority Q.

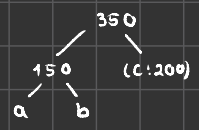
Piority Q : (: 150)(c : 200)(d : 250)(e : 300)

④ Repeat until only one data in Piority Q

Pop

Piority Q : (d : 250)(e : 300)

Sum



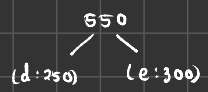
Insert

Piority Q : (d : 250)(e : 300)(: 350)

Pop

Piority Q : (: 350)

Sum



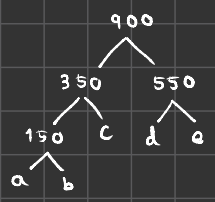
Insert

Piority Q : (: 350)(: 550)

Pop

Piority Q :

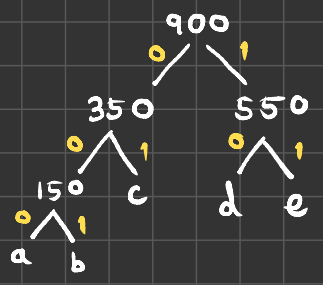
Sum



Insert

Piority Q : (: 900)

Result



Encoding

a : 000
b : 001
c : 01
d : 10
e : 11

} No ambiguity, all unique