Data Structure

Lecture#13: Binary Trees 4 (Chapter 5)

U Kang
Seoul National University

In This Lecture

- Learn the Huffman tree data structure
- Learn the concept of trie

Huffman Coding Trees

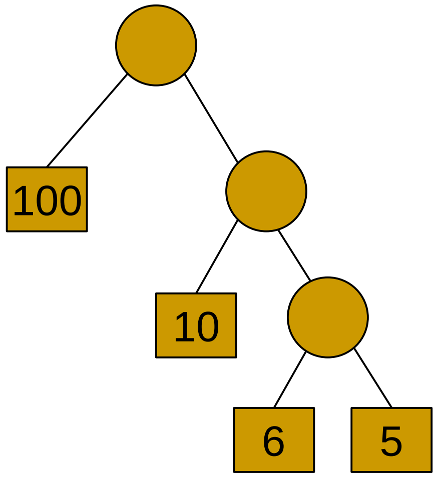
- ASCII codes: 8 bits per character.
  - Fixed-length coding.
- Can take advantage of relative frequency of letters to save space.
- Variable-length coding
  - E.g., 1 bit for ‘e’, 6 bit for ‘z’

Z	K	M	C	U	D	L	E
2	7	24	32	37	42	42	120

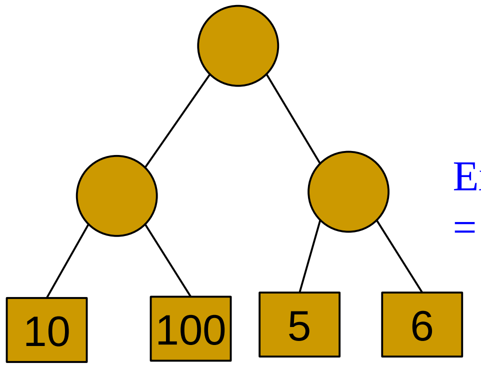
- How can we assign variable length codes to letters?

Huffman Coding Trees

- Goal: build a binary tree with minimum external path weight.
  - Each leaf corresponds to a letter
  - Weighted path length of a leaf: weight × (depth of the leaf)
  - External path weight: sum of weighted path lengths for the leaves



External path weight = 153

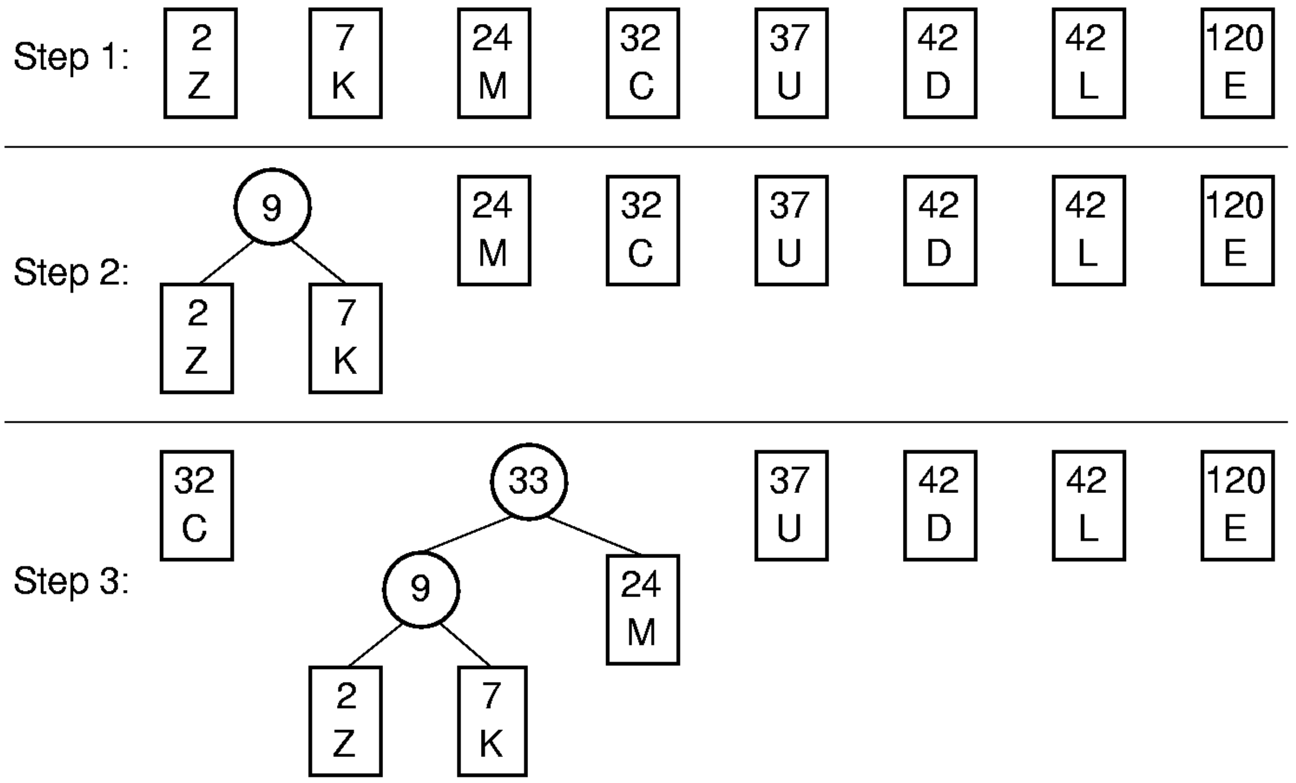


External path weight = 242

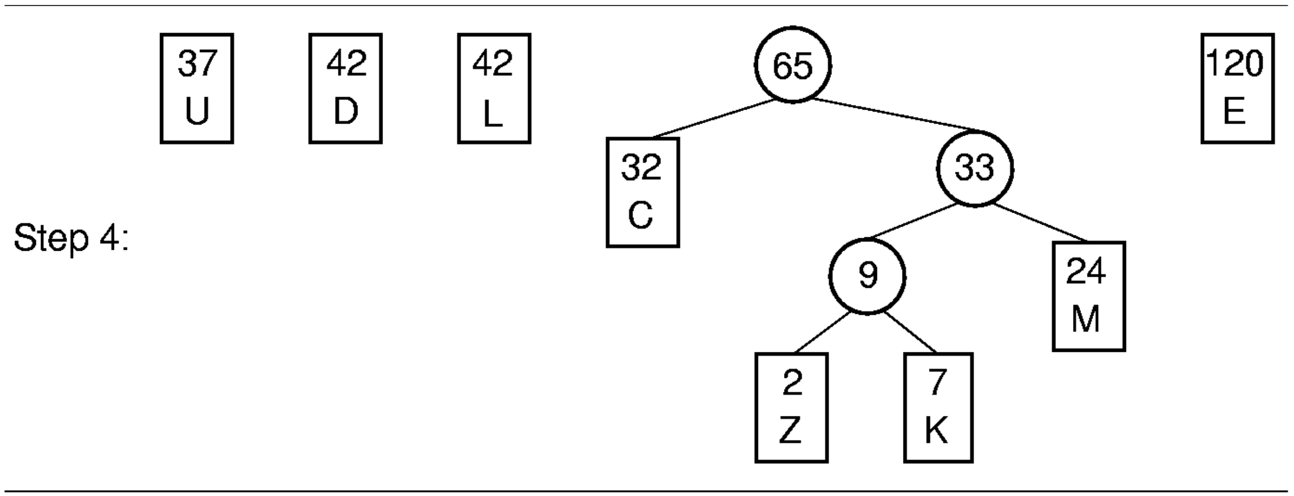
Huffman Coding Trees

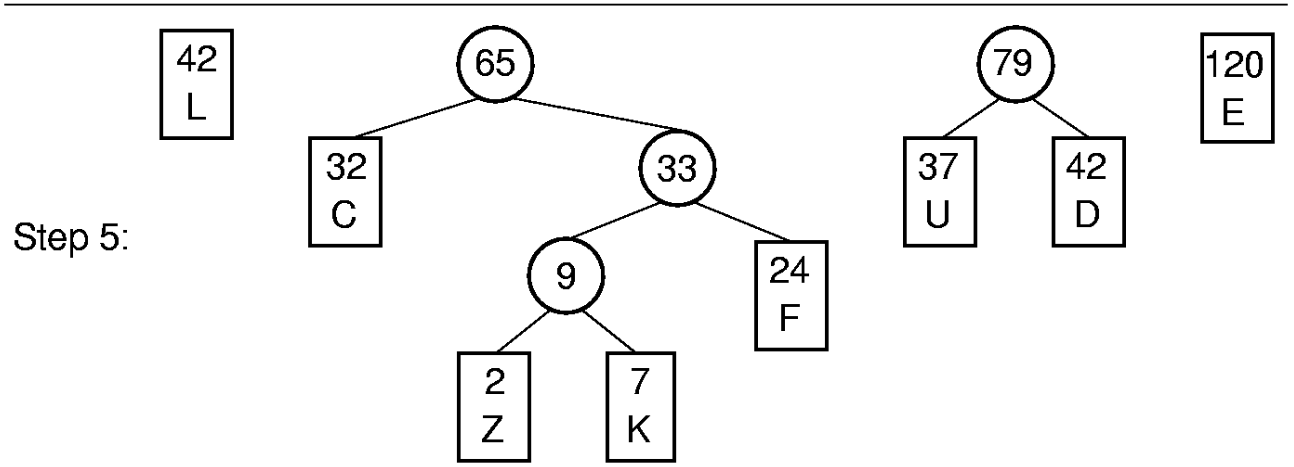
- How can we get the tree with minimum external path weight?
- Theorem: Huffman coding tree gives the minimum external path weight

Huffman Tree Construction (1)

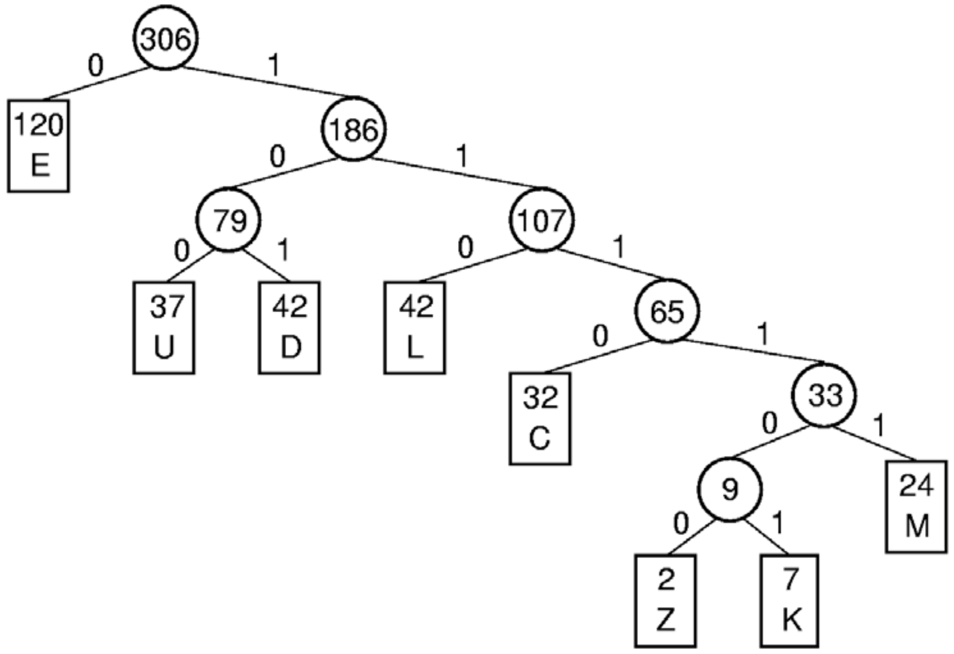


Huffman Tree Construction (2)





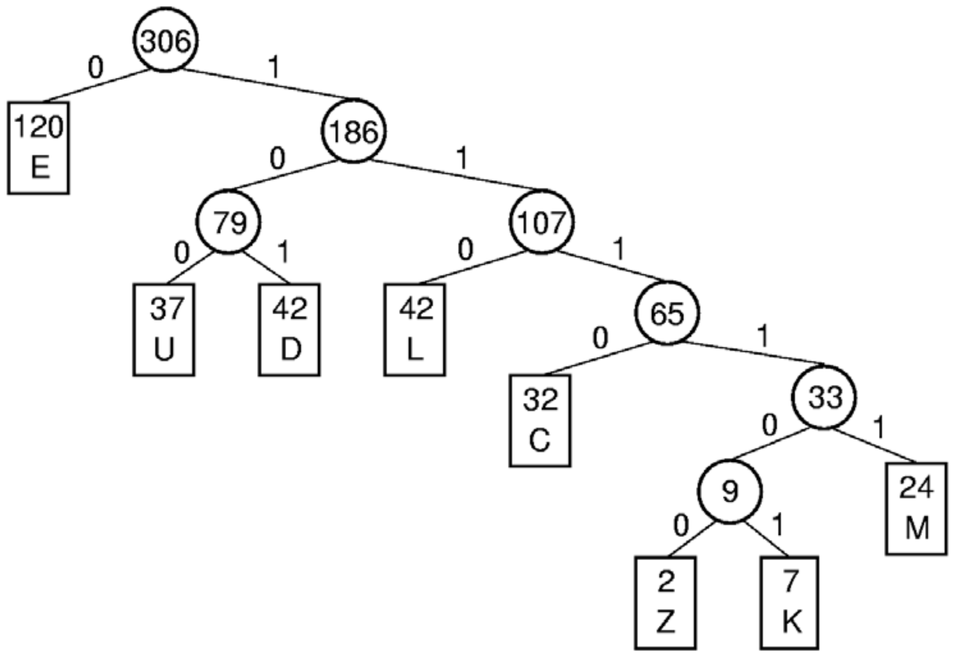
Assigning Codes



Letter	Freq	Code	Bits
C	32		
D	42		
E	120		
K	7		
L	42		
M	24		
U	37		
Z	2		

Assign 0 to left branch, 1 to right branch
Follow the path from the root to each letter

Assigning Codes



Letter	Freq	Code	Bits
C	32	1110	4
D	42	101	3
E	120	0	1
K	7	111101	6
L	42	110	3
M	24	11111	5
U	37	100	3
Z	2	111100	6

Assign 0 to left branch, 1 to right branch
Follow the path from the root to each letter

Using Huffman Codes

- To encode a string using Huffman codes, just concatenate the Huffman code for each letter
  - Code for DEED: 10100101

- Given a Huffman-encoded string, can we correctly decode it?

Coding and Decoding

- Given a Huffman-encoded string, can we correctly decode it?
  - Yes! Thanks to the ‘prefix’ property of Huffman code

- A set of codes is said to meet the <u>prefix property</u> if no code in the set is the prefix of another.
  - Thus, each code is uniquely identified

- Decode $\underline{101}\,\underline{100}\,\underline{1110}\,\underline{111101}$:

Expected Cost

- Expected cost = Average cost per letter =

$$\frac{c_1 f_1 + c_2 f_2 + \cdots + c_n f_n}{f_T}$$

  c_i = bits for letter i
  f_i = freq for letter i
  f_T = all frequncies

- The expected cost is 2.57 bits/letter which is smaller than log 8 = 3 bits/letter

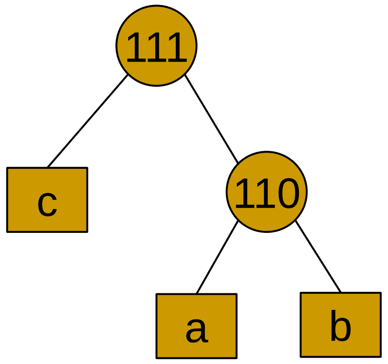
Letter	Freq	Code	Bits
C	32	1110	4
D	42	101	3
E	120	0	1
K	7	111101	6
L	42	110	3
M	24	11111	5
U	37	100	3
Z	2	111100	6

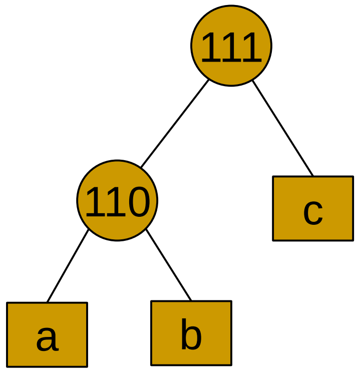
Uniqueness of Huffman Tree

- Is Huffman tree unique?

Uniqueness of Huffman Tree

- Is Huffman tree unique?
  - No. We can arbitrarily choose to make a node right or left child of the new parent.
  - E.g., let a, b, and c appear 100, 10, and 1 times, respectively. Both of the two trees shown below are valid Huffman trees. Nonetheless, the external path lengths for the two trees are the same.





Optimality of Huffman Tree

Details

- (Theorem) Huffman coding tree gives the minimum external path weight
- (Proof)
  - (Lemma 1) There exists an optimal tree that contains two characters with least frequency as sibling nodes whose depth is at least as deep as any other leaf nodes in the tree.
    - (Proof by contradiction) Assume the conclusion of the Lemma is false. Let L={x,y} be the set of two least frequency nodes. Let y be not the deepest, and z ∉ L be the deepest. Swapping y and z decreases the external path weight, thus contradiction. Thus, both x and y should be the deepest.
    - What if x and y are not siblings? Assume x ∈ L and z ∉ L are siblings; then we can swap y and z to make another optimal tree where x and y are siblings.
  - Proof is by induction on n, the number of letters
  - Base case: for n = 2, Huffman tree is optimal.

Optimality of Huffman Tree

Details

EPL: external path length

- (Proof: continued)
  - Induction hypothesis: for n-1 letters Huffman tree is optimal.
  - Induction step (proof by contradiction)
    1. Let T be a Huffman tree from n letters.
    2. Let x and y be letters with least frequencies in T. x and y are siblings since T is a Huffman tree.
    3. Let v be the parent of x and y in T. Let T’ be a tree by replacing v with a leaf node v’ whose weight is w(x)+w(y). Note that T’ is also a Huffman tree with n-1 letters. From I.H., T’ is optimal.
    4. Assume T is not optimal; i.e., let Z be an optimal tree whose EPL is smaller than T. From Lemma1, we know that Z contains x and y as the deepest siblings. Create tree Z’ by replacing the parent of x and y with a new node v’’ whose weight is w(x)+w(y). Then, EPL(Z) = EPL(Z’) + w(x) + w(y) ≥ EPL(T’) + w(x) + w(y) = EPL(T) , where the inequality uses the result of step 3. This is a contradiction to the assumption that T is not optimal. Thus, T should be optimal.

Search Tree vs. Trie

- In a BST, the root value splits the key range into everything less than or greater than the key
  - The split points are determined by the data values
- View Huffman tree as a search tree
  - All keys starting with 0 are in the left branch, all keys starting with 1 are in the right branch
  - The split points are determined by the data structure, not the data values
  - Such a structure is called a Trie

What you need to know

- Huffman tree
  - Goal, how to construct Huffman tree
  - Encoding/decoding using Huffman tree
- Compare trie vs. tree

Questions?