Static Huffman Coding

- we are given frequency distribution $F = (f_1, \ldots, f_n)$
  determine codelengths $L = (L_1, \ldots, L_n)$
  to minimize resulting encoding length $= F \cdot L = \sum f_i L_i$

- Huffman Algorithm (original version)

  initialize list with ordered set of frequencies
  while *len*(list) > 1 **do**
    merge 2 smallest values (i, j) into one value (x)
    represent x by creating a parent node having children i and j
    insert x in proper place within the list

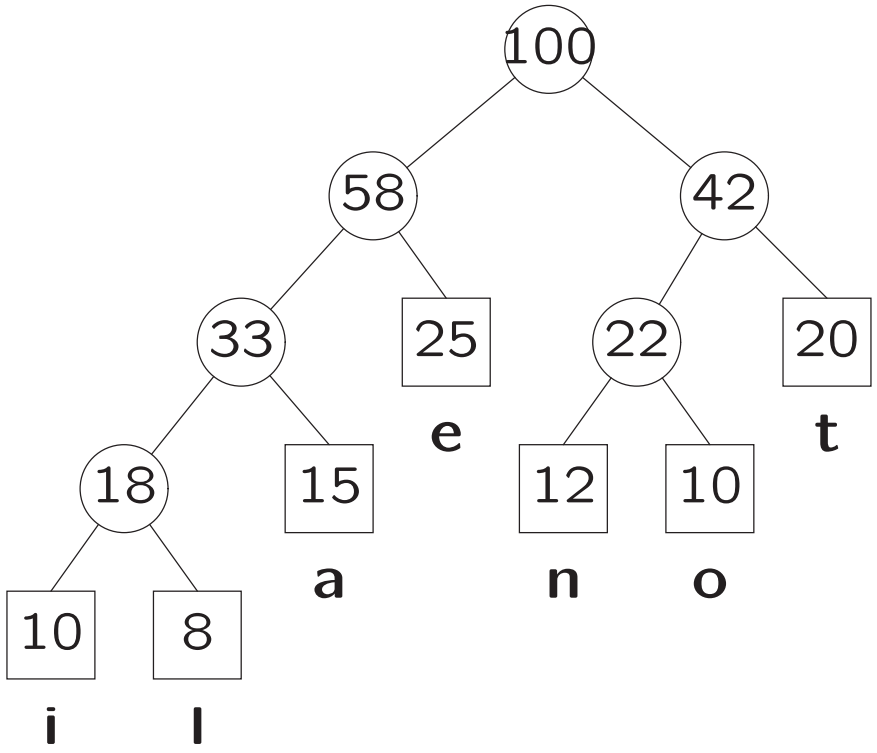
- avoid insertion search: use 2 lists (leaf+internal)

  while lists contain more than one value **do**
    merge 2 smallest values (i, j) into one value (x)
    represent x by creating a parent node having children i and j
    place x at end of 'internal' list

Static Huffman Coding

Example:

e	t	a	n	o	i	l
25	20	15	12	10	10	8



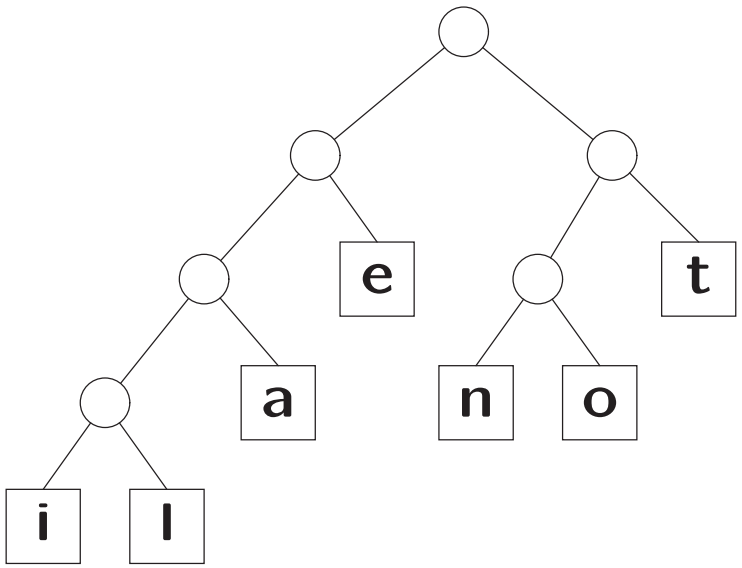
Static Huffman Coding

Decoding:

- encoder transmits the codetree in some form
- decoder iterates determining next char by
  - start at top (root) of tree
  - branch L/R (for 0- or 1-bit) until a Leaf
  - output character associated with Leaf

Example:

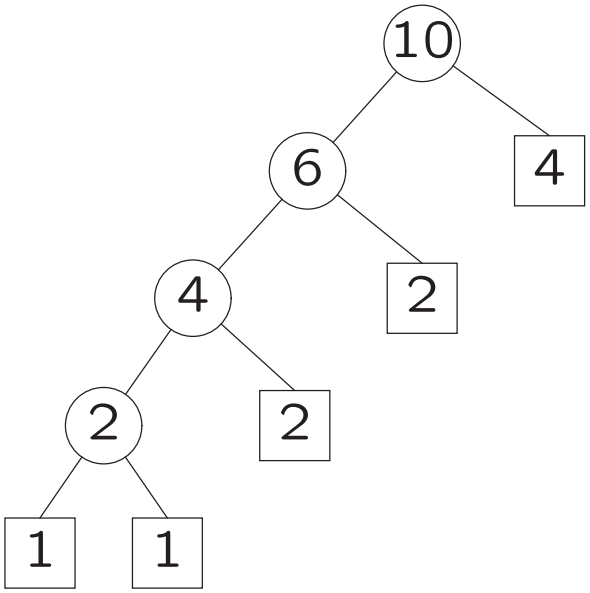
```
1  1  1  0  1  0  0  0  0  0  0  0  1
   t        o           i           l
```

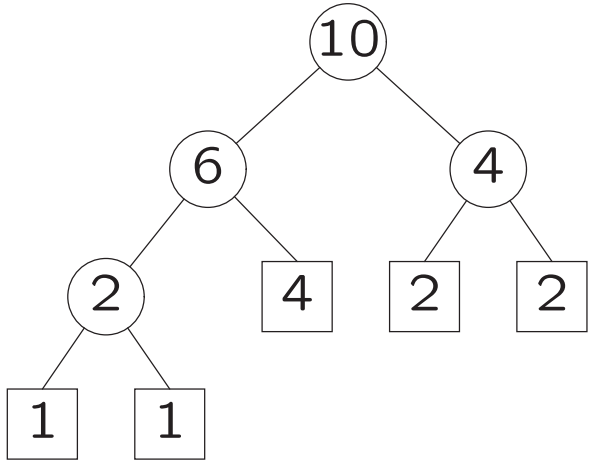


Static Huffman Coding

- prefer merging leaf will minimize variance

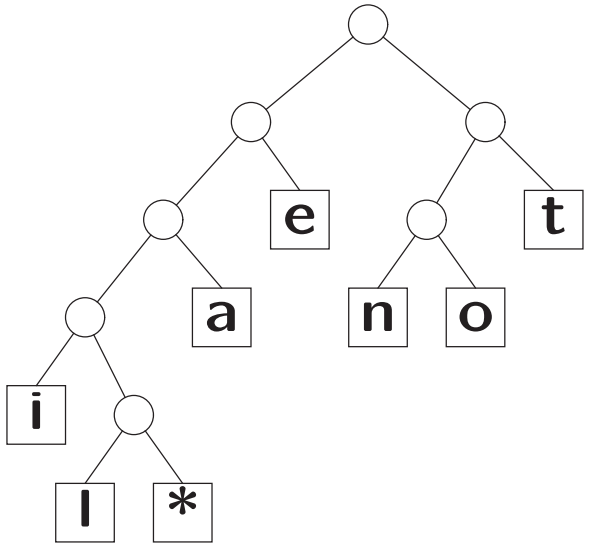
Example: `1, 1, 2, 2, 4`



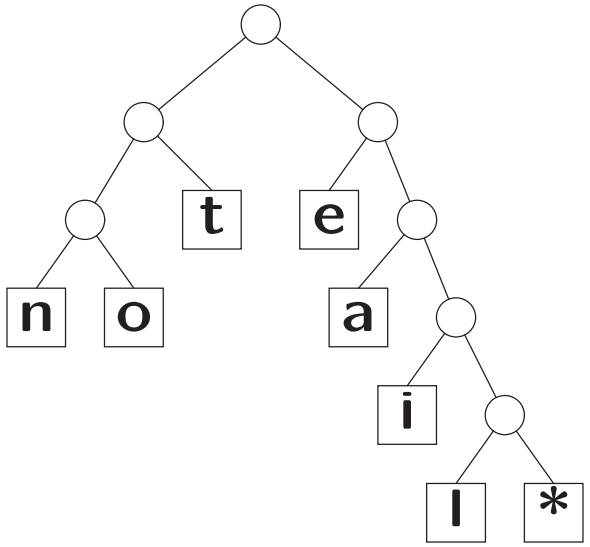


Code Tree Representation

- can construct Huffman codetree that contains EOF node

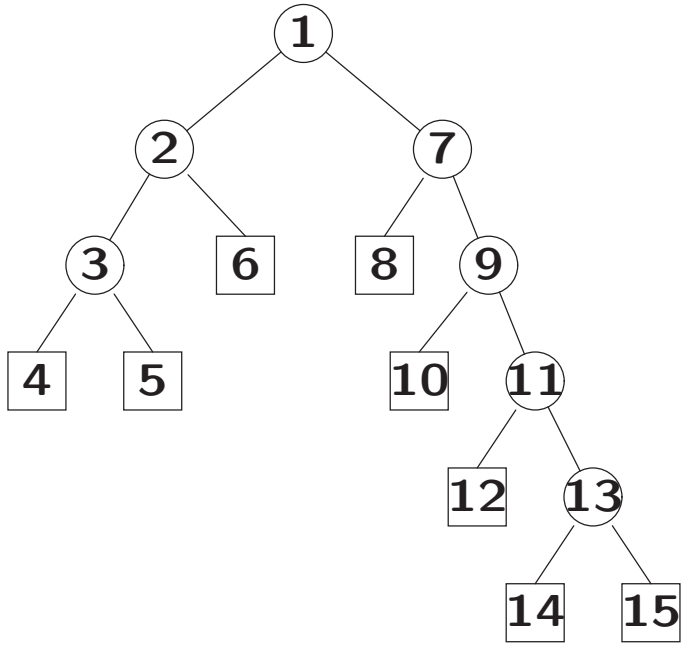


- carefully construct Huffman codetree that contains EOF node by ordering siblings to cause EOF node to be rightmost leaf



Code Tree Representation

- diagram shows preorder numbering of tree nodes

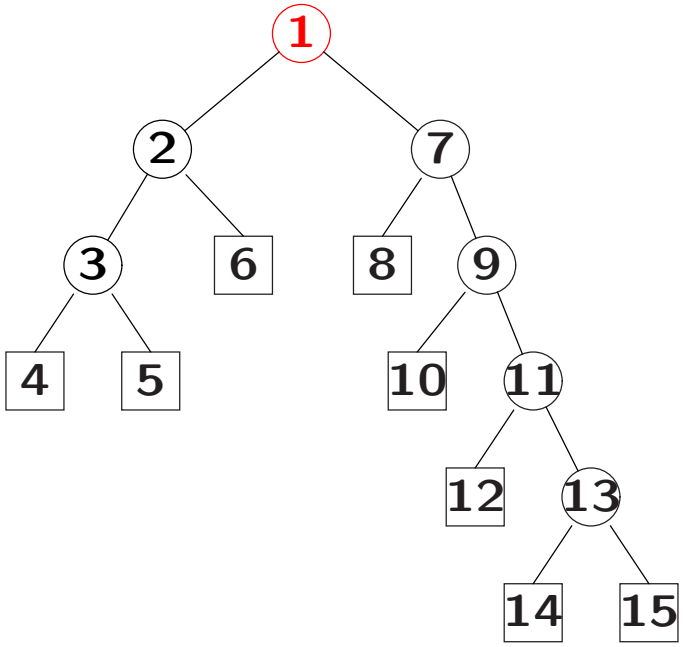


- bit string showing nodes are leaf (square) or internal (circle) enables reconstruction of tree structure

① ② ③ 4 5 6 ⑦ 8 ⑨ 10 ⑪ 12 ⑬ 14 15

Code Tree Representation

- diagram shows preorder numbering of tree nodes

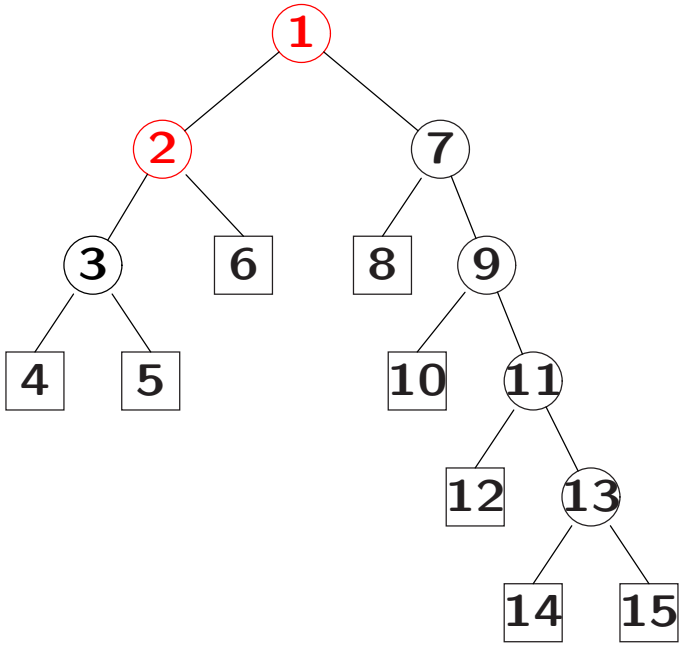


- bit string showing nodes are leaf (square) or internal (circle) enables reconstruction of tree structure

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

Code Tree Representation

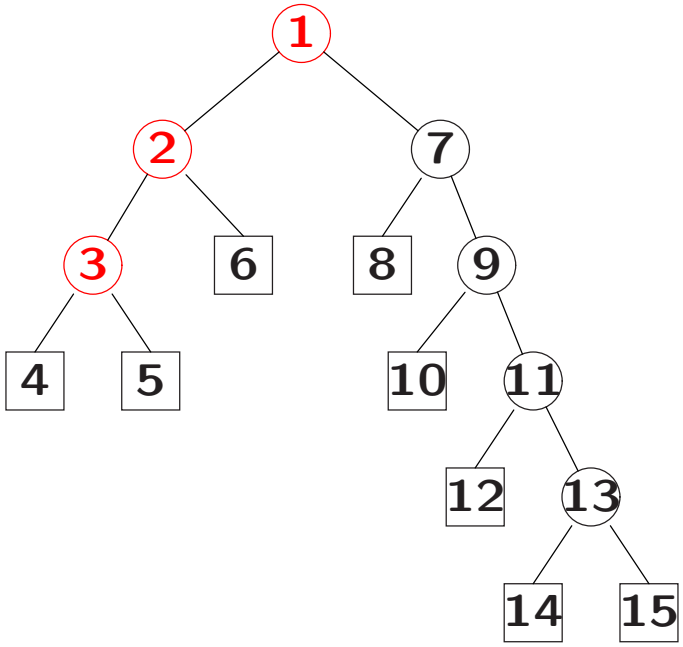
- diagram shows preorder numbering of tree nodes



- bit string showing nodes are leaf (square) or internal (circle) enables reconstruction of tree structure

Code Tree Representation

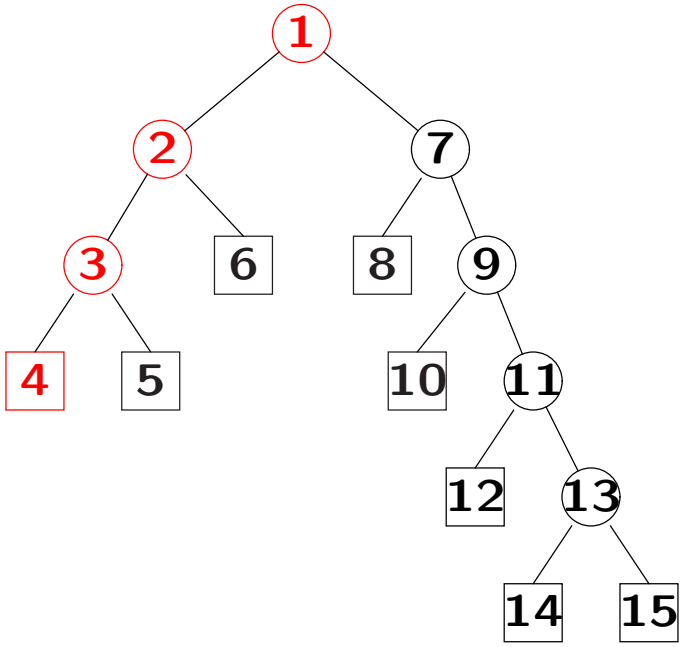
- diagram shows preorder numbering of tree nodes



- bit string showing nodes are leaf (square) or internal (circle) enables reconstruction of tree structure

Code Tree Representation

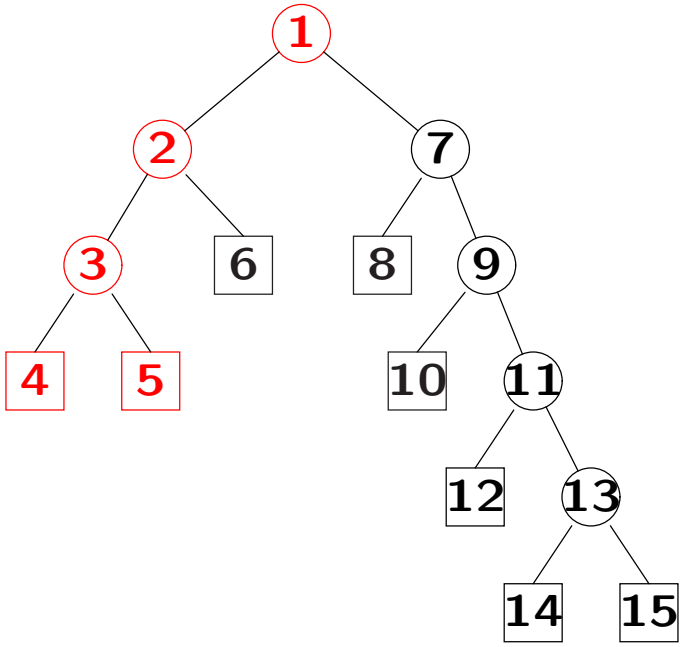
- diagram shows preorder numbering of tree nodes



- bit string showing nodes are leaf (square) or internal (circle) enables reconstruction of tree structure

Code Tree Representation

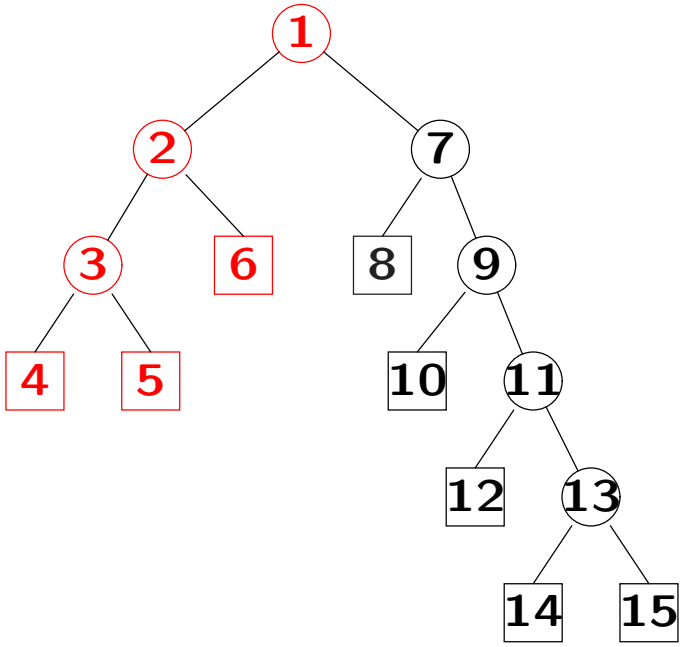
- diagram shows preorder numbering of tree nodes



- bit string showing nodes are leaf (square) or internal (circle) enables reconstruction of tree structure

Code Tree Representation

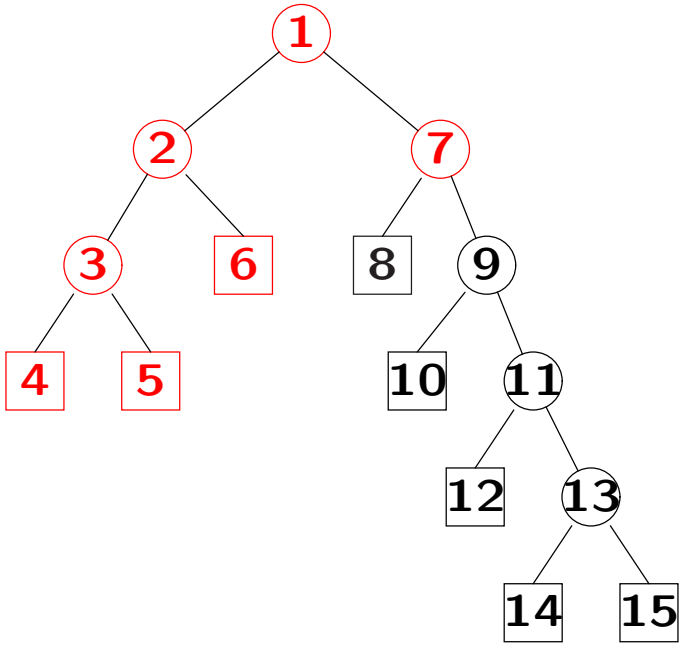
- diagram shows preorder numbering of tree nodes



- bit string showing nodes are leaf (square) or internal (circle) enables reconstruction of tree structure

Code Tree Representation

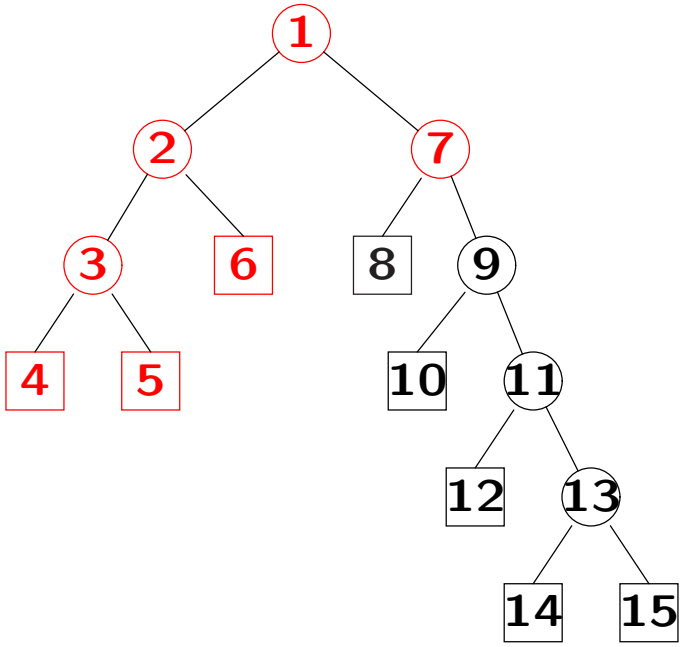
- diagram shows preorder numbering of tree nodes



- bit string showing nodes are leaf (square) or internal (circle) enables reconstruction of tree structure

Code Tree Representation

- diagram shows preorder numbering of tree nodes



- bit string showing nodes are leaf (square) or internal (circle) enables reconstruction of tree structure

- express structure (which determines that there are $n + 1$ leaves) then express n characters in order (assumes that last leaf is EOF)