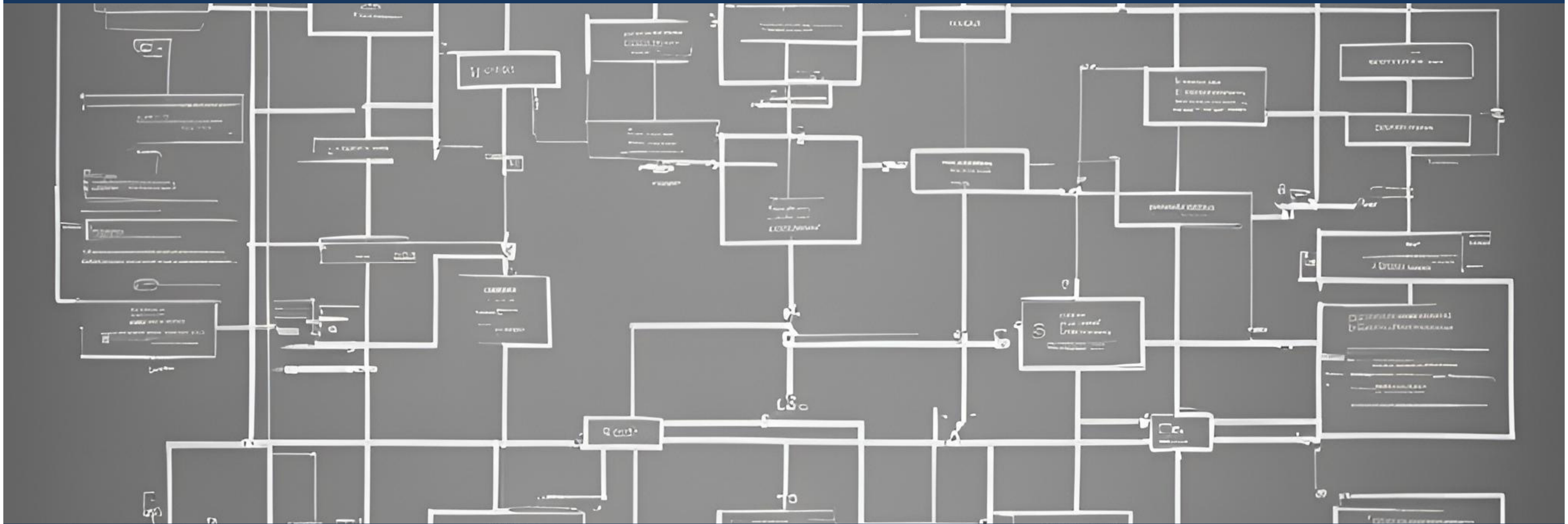
ITM 517 Algorithm

Ja-Hee Kim

Tree



Huffman coding

Huffman encoding

- The Huffman encoding algorithm is a greedy algorithm
- Given the percentage each character appears in a corpus, determine a variable-bit pattern for each char.
- You always pick the two smallest percentages to combine.

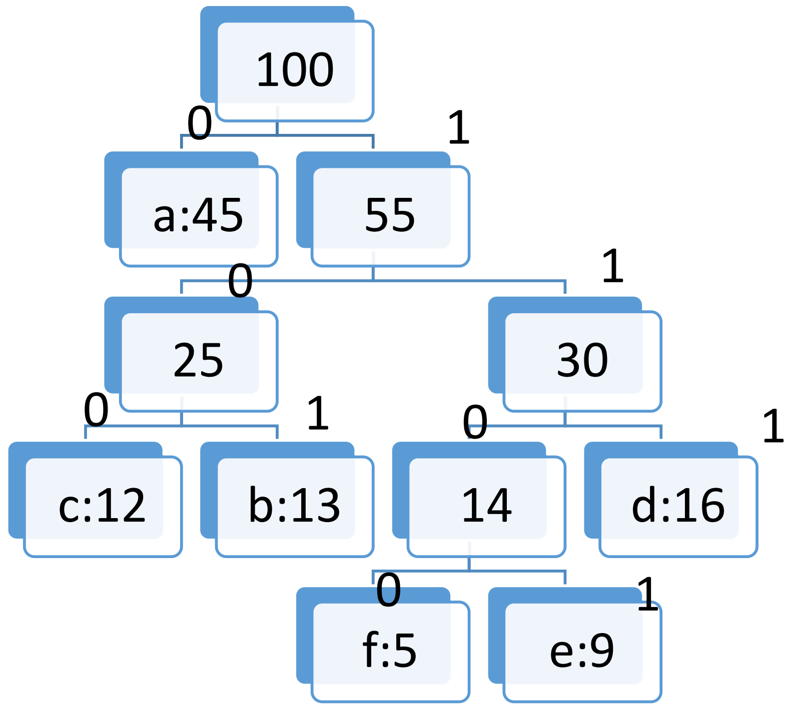
- Example

	a	b	c	d	e	f
frequency	45	13	12	16	9	5
Fixed-length code	000	001	010	011	100	101
Variable-length code	0	101	100	111	1101	1100

- Fixed: (45+13+12+16+9+5)*3=300 bits
- Various: $\sum freq \times code\_length$
  $= 45 \times 1 + (13 + 12 + 16) \times 3 + (9 + 5) \times 4 = 224 \text{ bits}$

Prefix code

- Prefix code: only codes in which no code word is also a prefix of some other codeword.
- More frequent characters are placed at lower levels of the tree, resulting in shorter codes.
- Left child: 0, right child:1



- Encoding cafe = 100010101011('c' → 100,'a' → 0,'f' → 1010,'e' → 1011)
- Cost of Tree T $B(T) = \sum_{c \in C} c.freq \times d_T(c)$

Algorithm

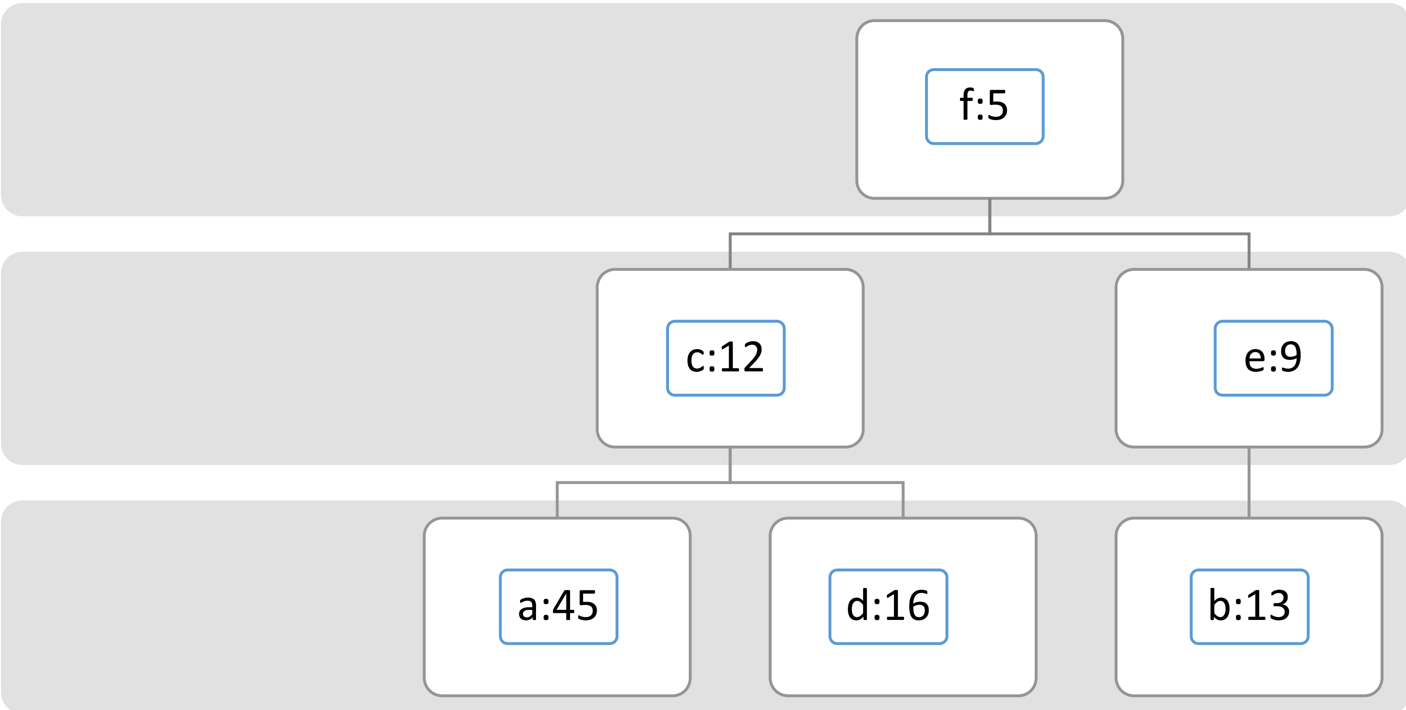
1. Create a leaf node for each unique character and build a min heap of all leaf nodes.
2. Extract two nodes with the minimum frequency from the min heap.
3. Create a new internal node with a frequency equal to the sum of the two nodes frequencies.
4. Repeat steps#2 and #3 until the heap contains only one node.

Example

- Given array of character and frequencies

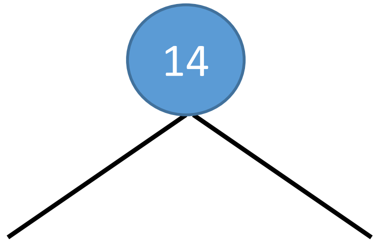
a:45 b:13 c:12 d:16 e:9 f:5

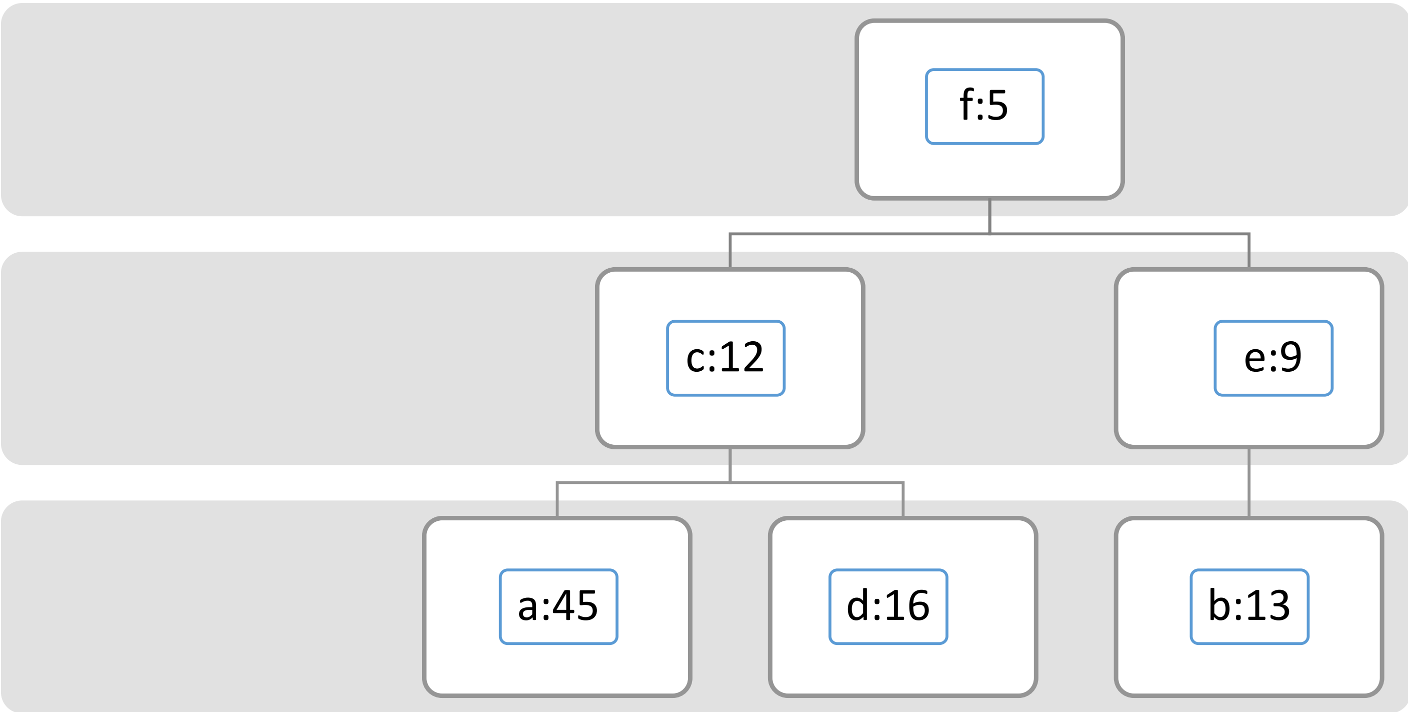
- Create a min Heap



Example

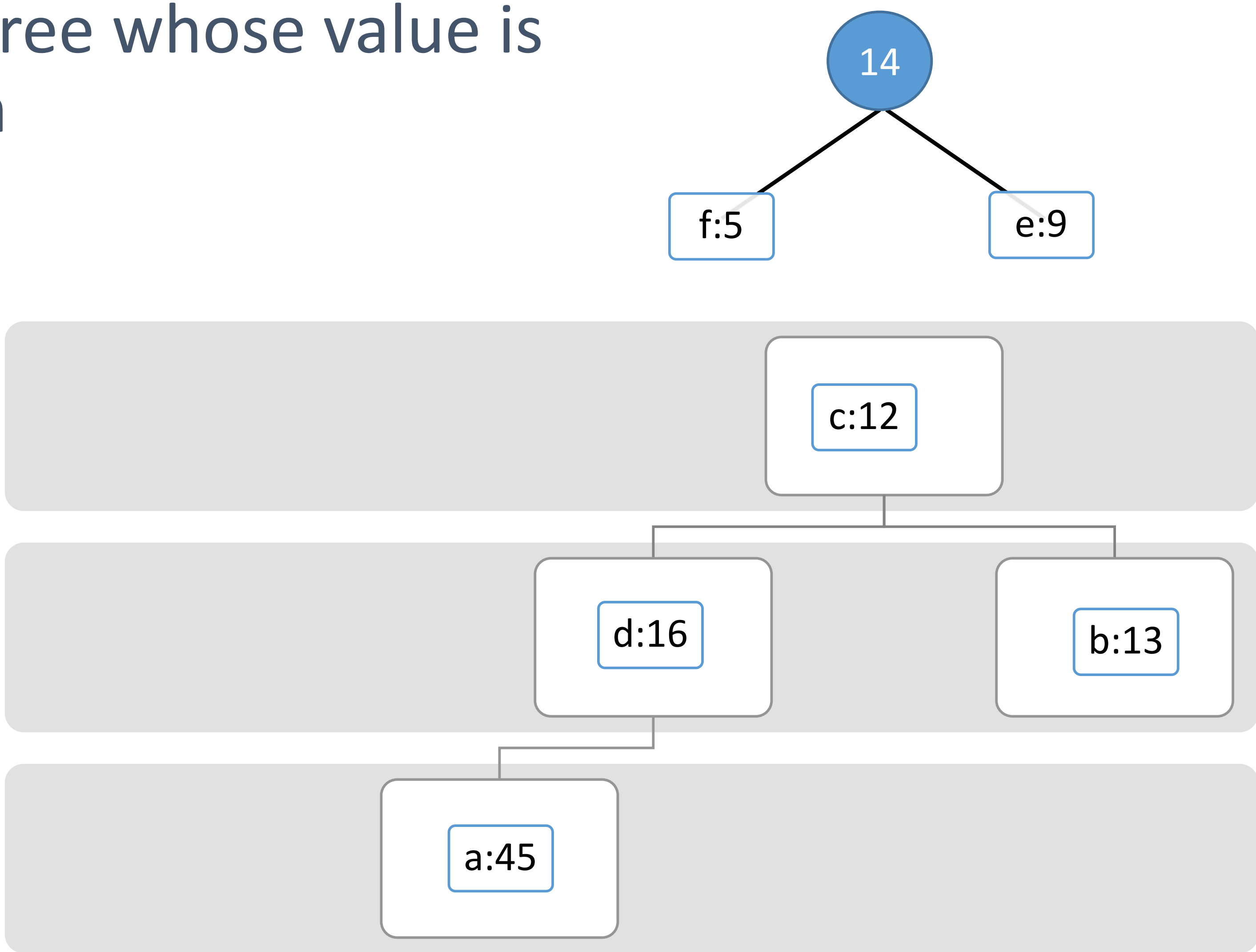
- Extract two nodes and create new subtree whose value is their sum





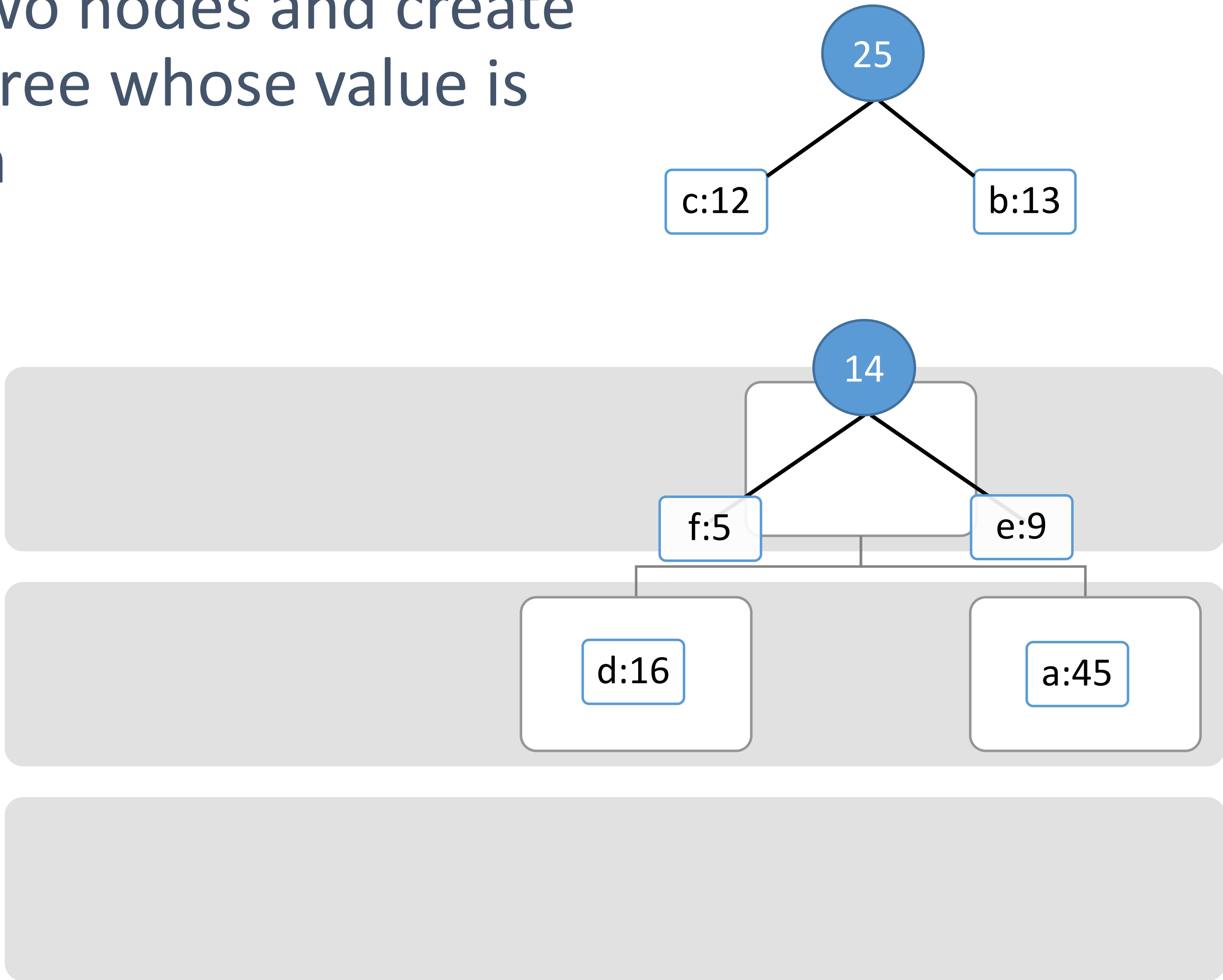
Example

- Extract two nodes and create new subtree whose value is their sum



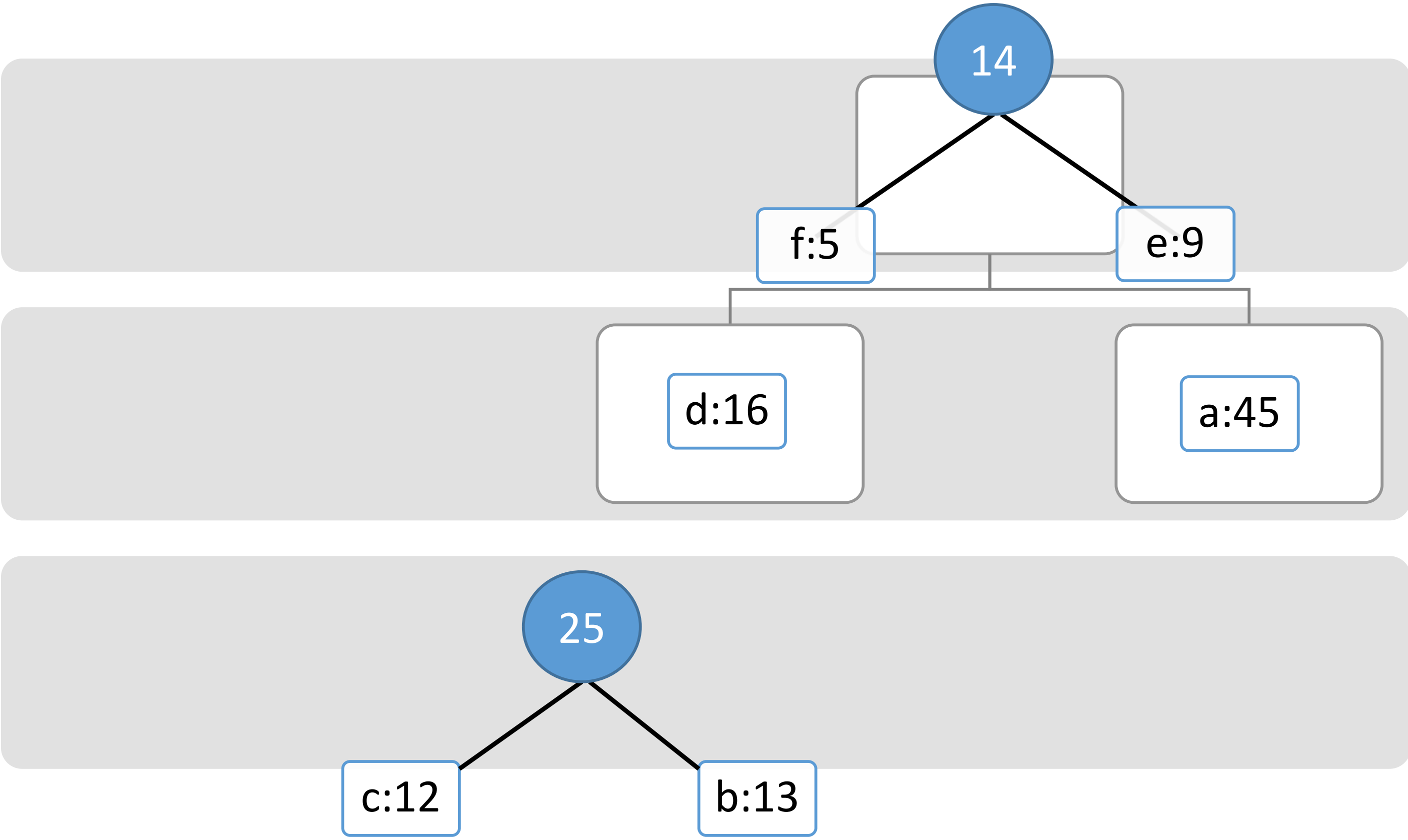
Example

- Extract two nodes and create new subtree whose value is their sum



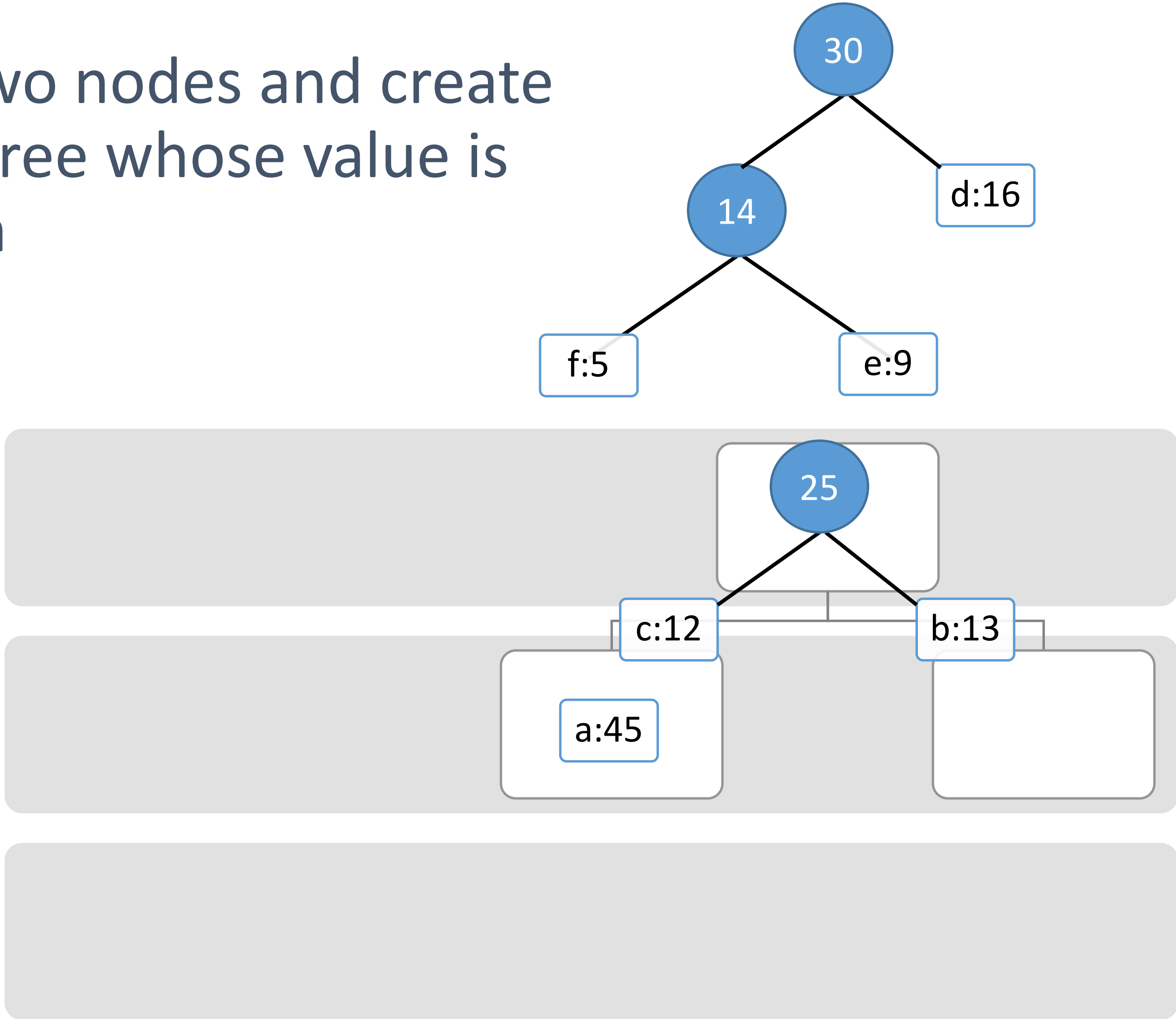
Example

- Extract two nodes and create new subtree whose value is their sum



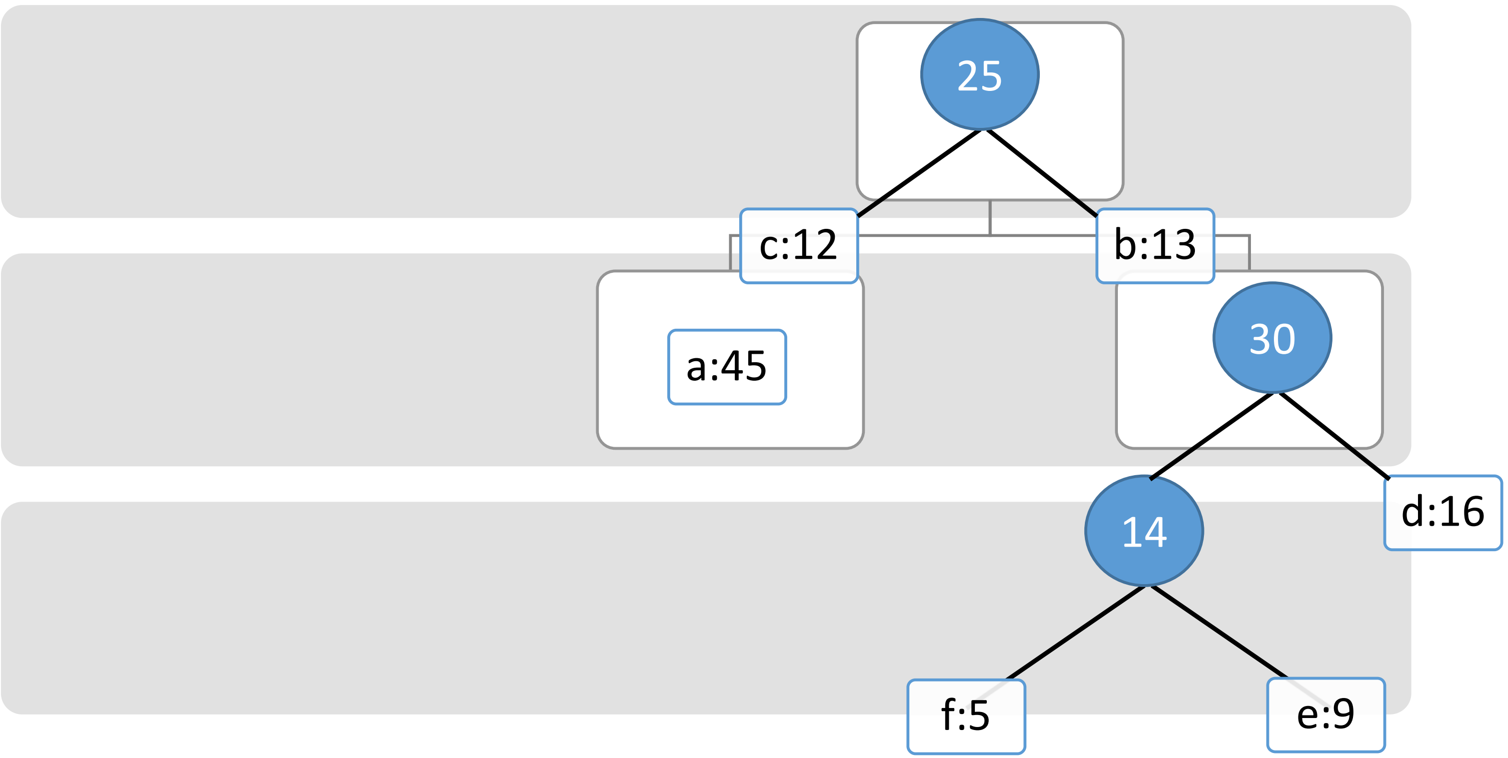
Example

- Extract two nodes and create new subtree whose value is their sum



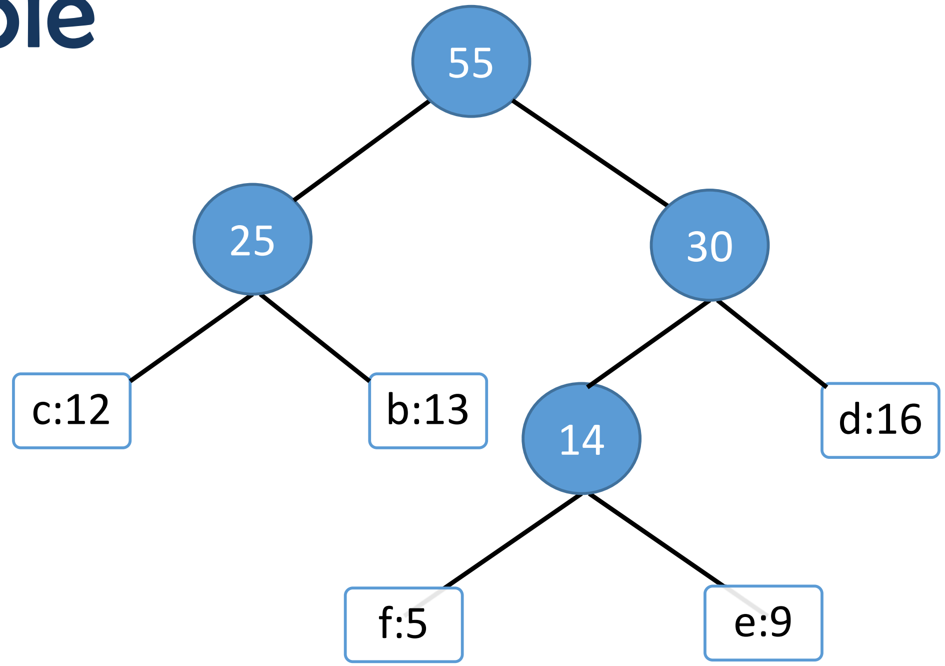
Example

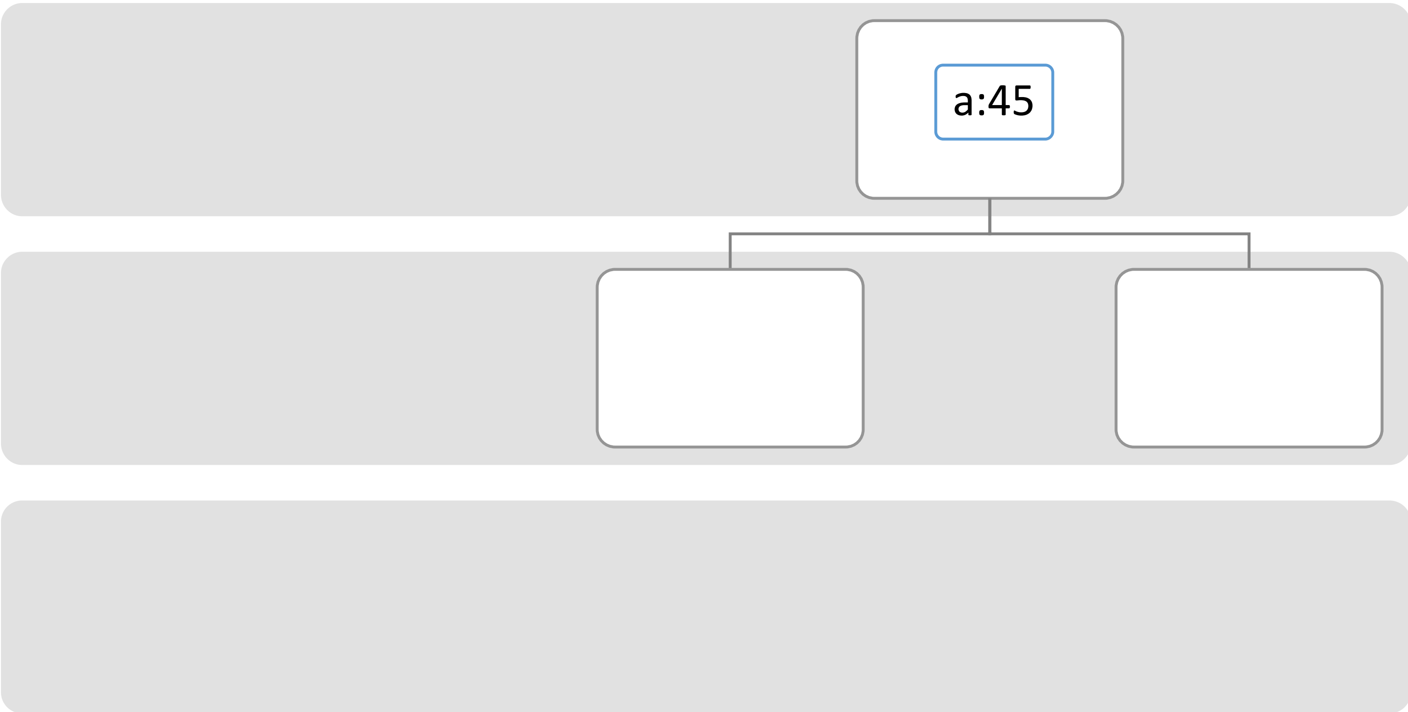
- Extract two nodes and create new subtree whose value is their sum



Example

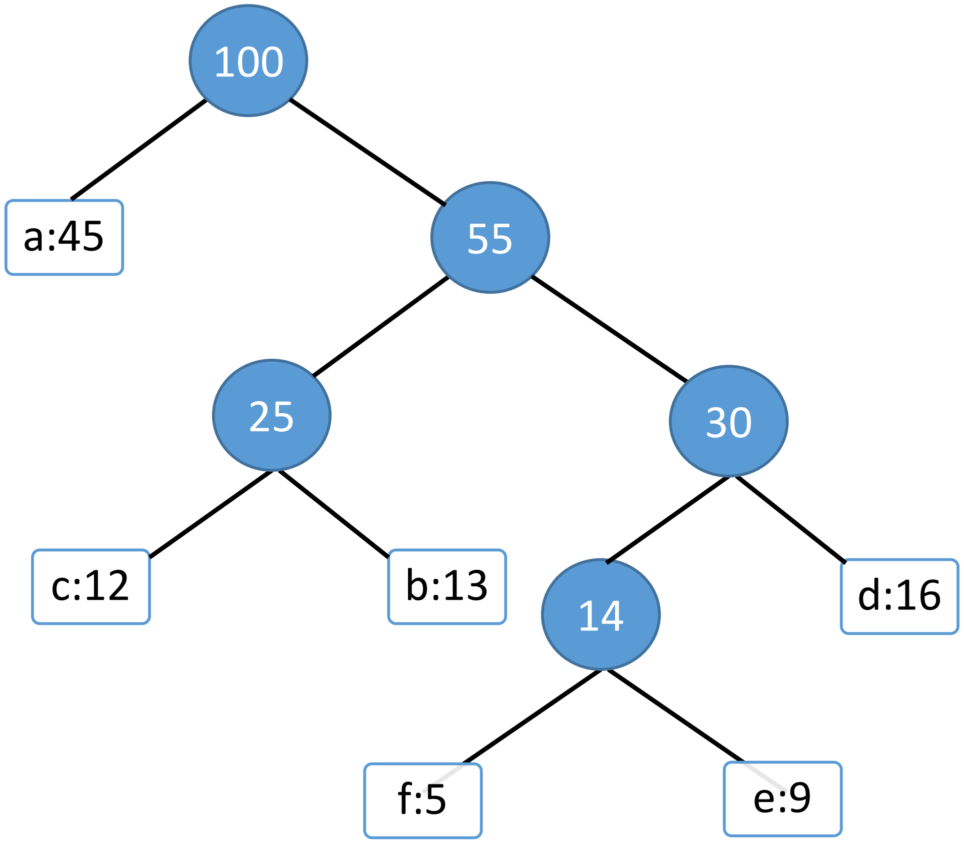
- Extract two nodes and create new subtree whose value is their sum





Example

- Repeat the previous steps until the heap contains only one node.



Analysis

- The algorithm typically makes (approximately) n choices for a problem of size n
  - (The first or last choice may be forced)
- Hence the expected running time is: $O(n * O(choice(n)))$, where $choice(n)$ is making a choice a mong n objects
  - Counting: Must find largest useable coin from among k sizes of coin (k is a constant), an $O(k)=O(1)$ operation;
    - Therefore, coin counting is (n)
  - Huffman: Must sort n values before making n choices
    - Therefore, Huffman is $O(n \log n) + O(n) = O(n \log n)$

Thanks