# CSC 172– Data Structures and Algorithms

Lecture #17

Spring 2018

Please put away all electronic devices

# Announcement

- Project 3 is out
  - Due: 04/13/2018 (11:59 pm)

- Huffman tree
- Priority Queue
- HashMap
- Encoding
- Decoding

# HUFFMAN CODING

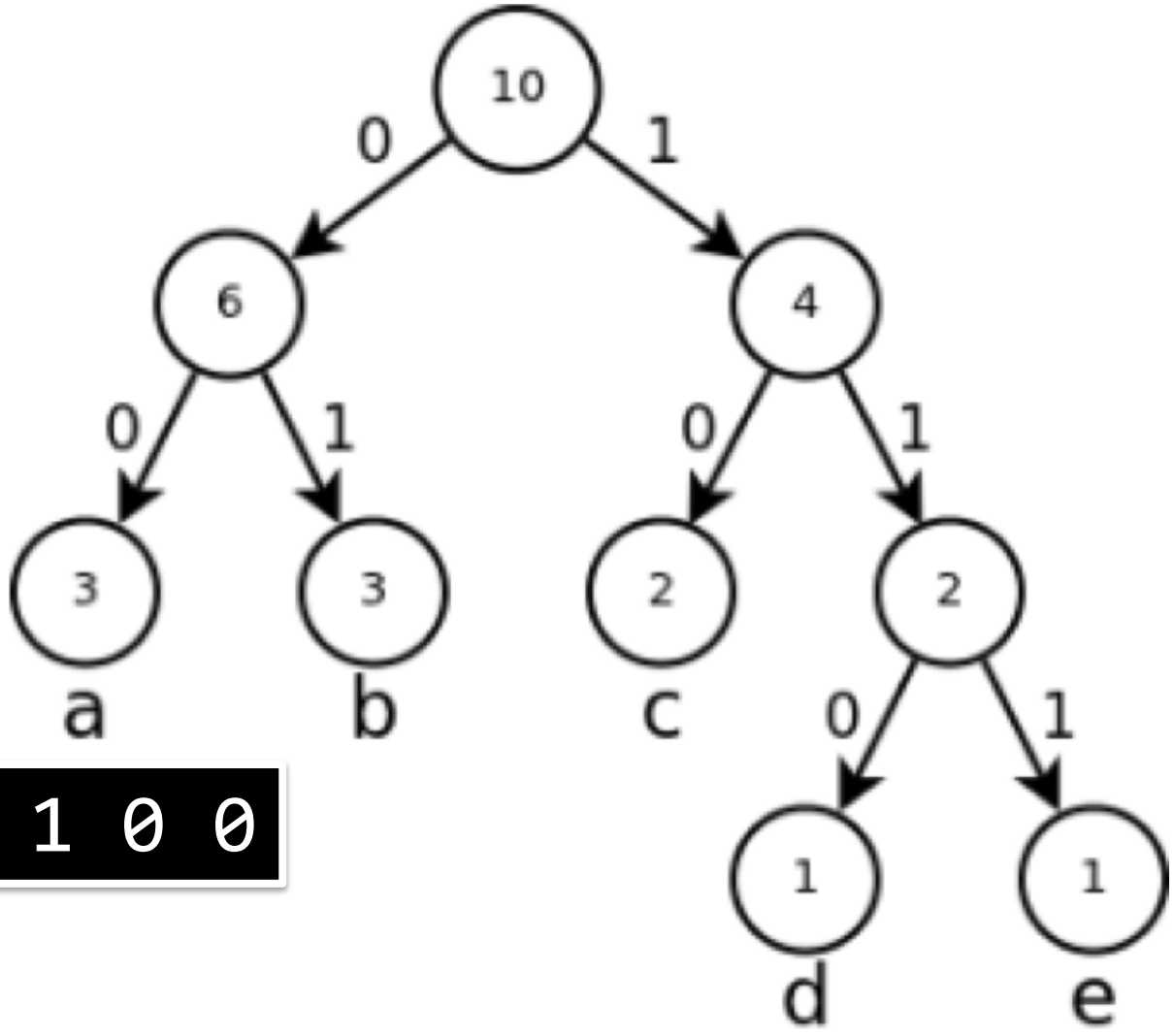
# Recall: Variable-Length Encoding: Idea

- Encode letter E with fewer bits, say $b_E$ bits
- Letter J with many more bits, say $b_J$ bits
- We gain space if
- $$b_E \cdot f_E + b_J \cdot f_J < 8f_E + 8f_J$$

where ***f*** is the frequency vector

- Problem: how to decode?

# Recall: Solution --- Prefix-Free Codes




```
1 0 1 1 1 0 1 0 0
```

```
c e b a
```

# Objectives

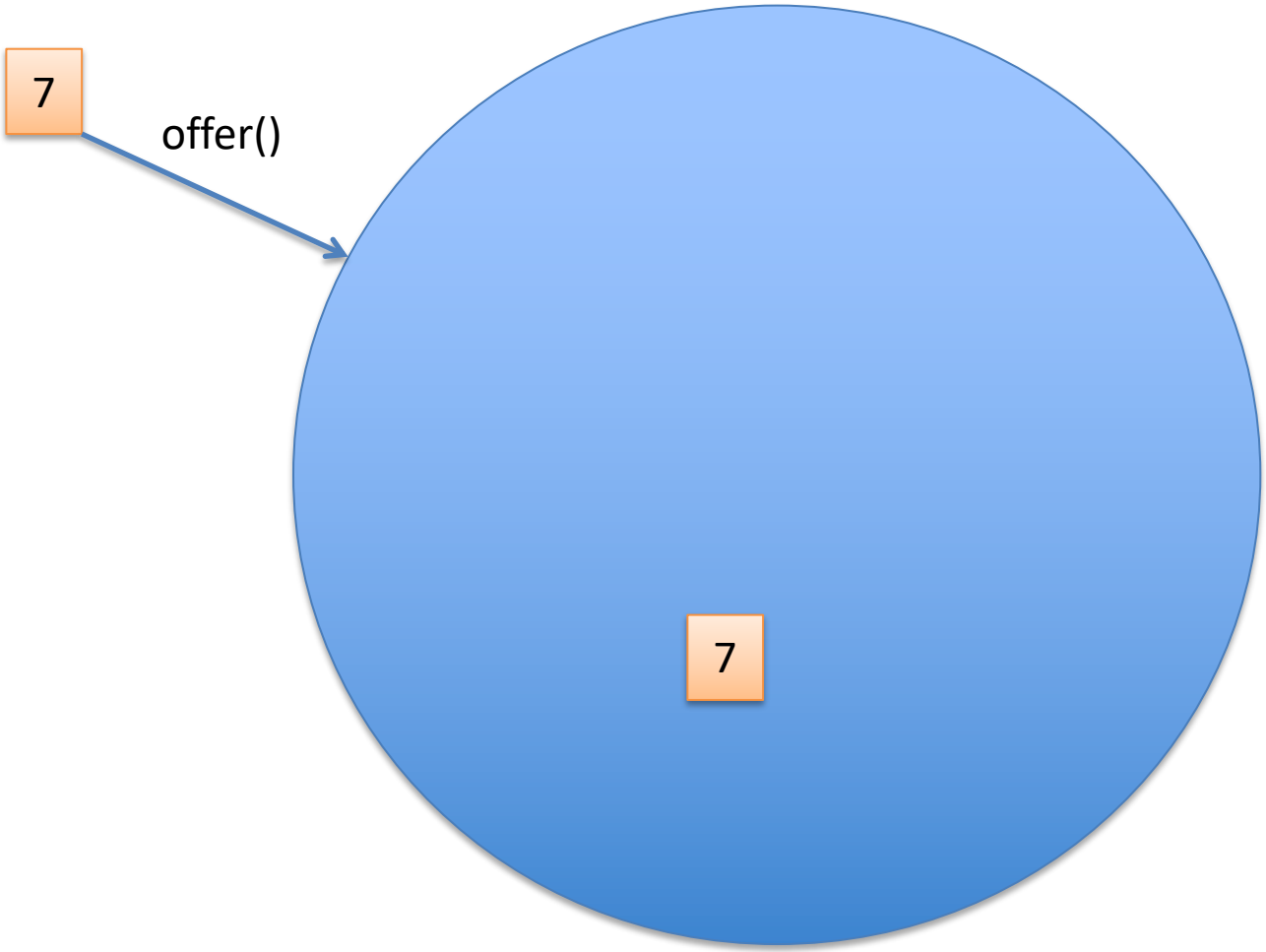
- Huffman tree
- Priority Queue
- HashMap
- Encoding
- Decoding

# PRIORITY QUEUE

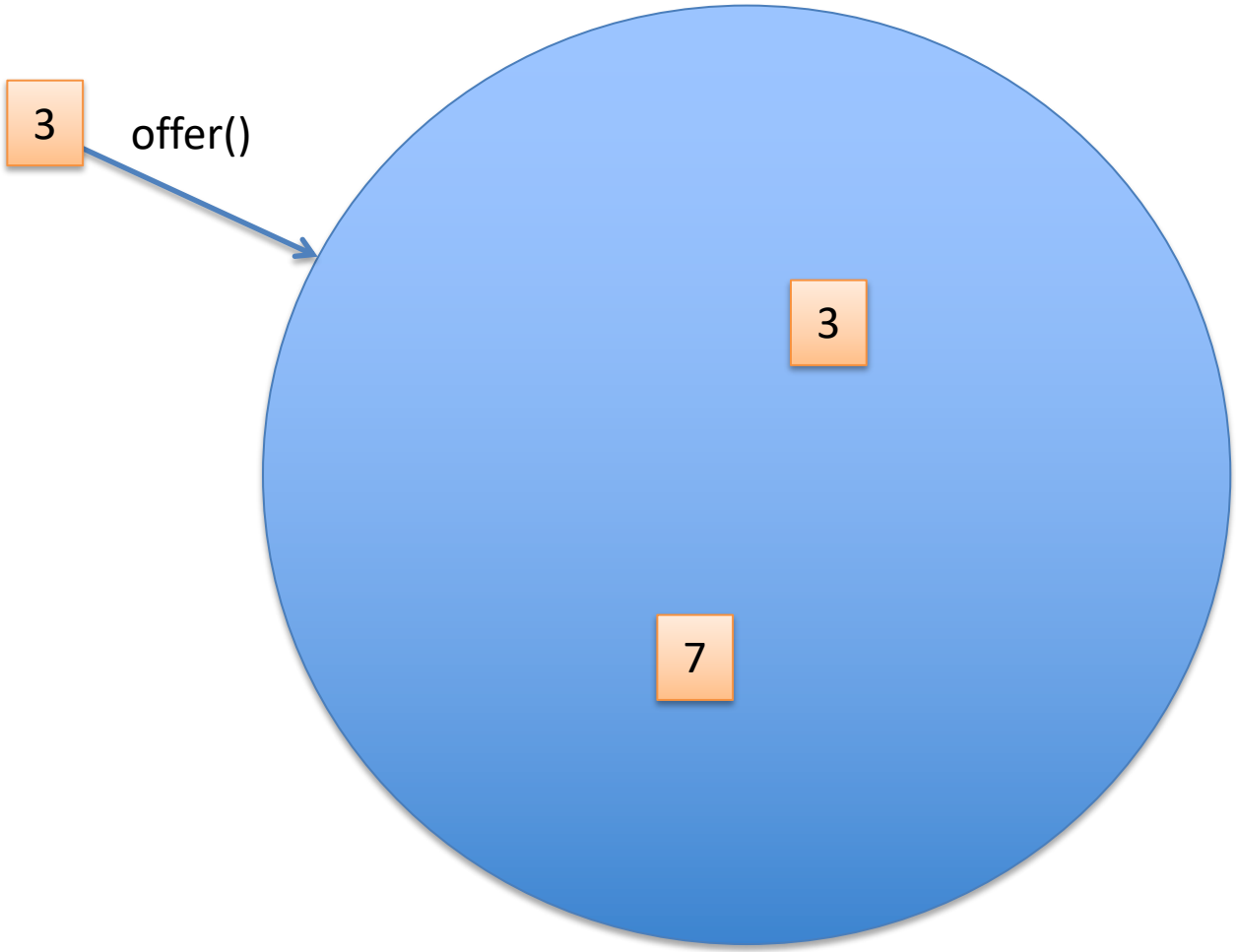
# Priority Queues

- Stack: FILO
  - `push(), pop(), peek()`
- Queue: FIFO
  - `offer(), poll(), peek()`
- Priority Queue:
  - Is a Queue
  - Each element has a “priority”
  - `offer()` stores new element (with “priority”)
  - `poll()` removes element with highest/lowest priority
  - `peek()` shows that element but doesn’t remove

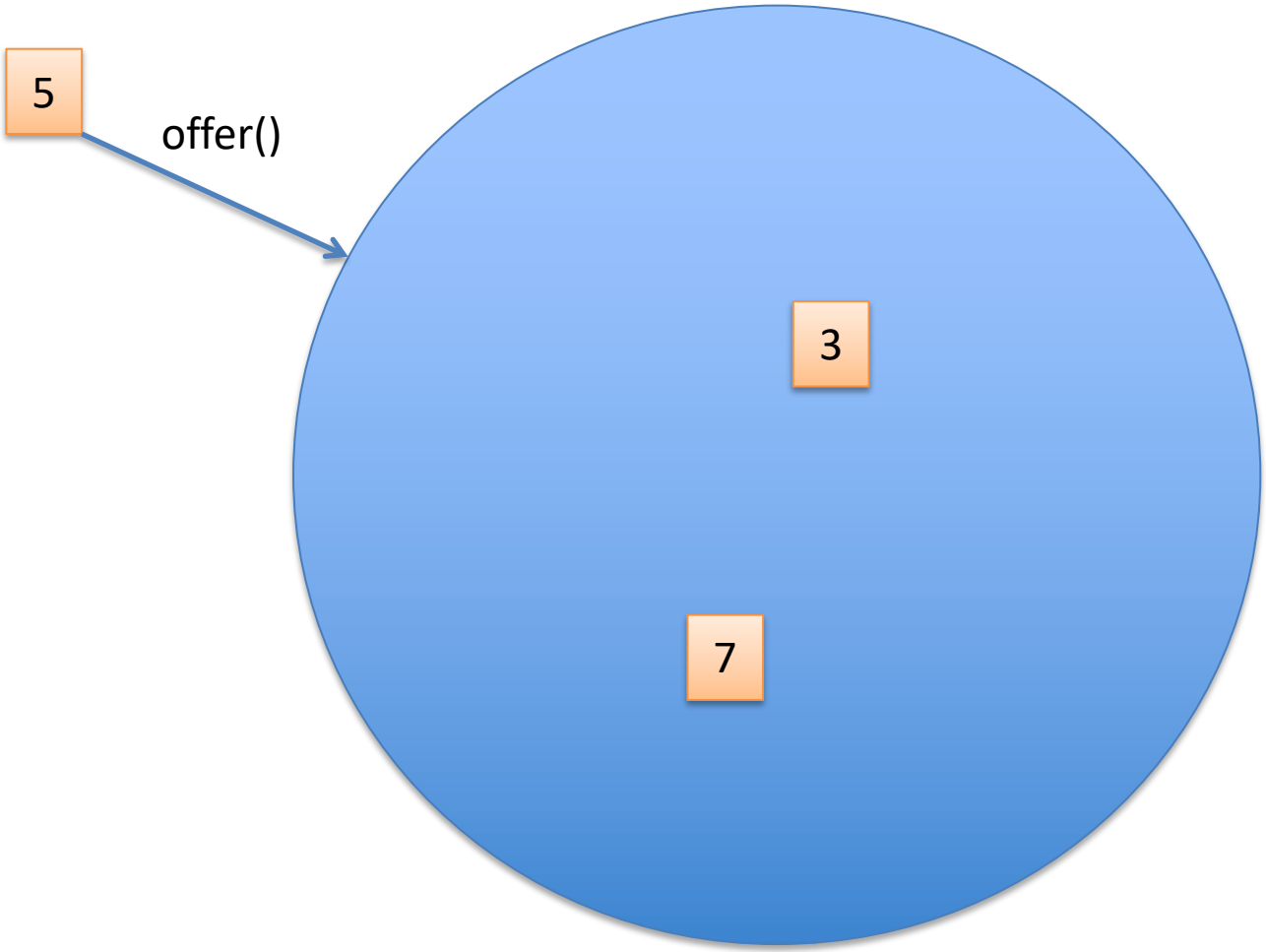
# A Priority Queue in Picture



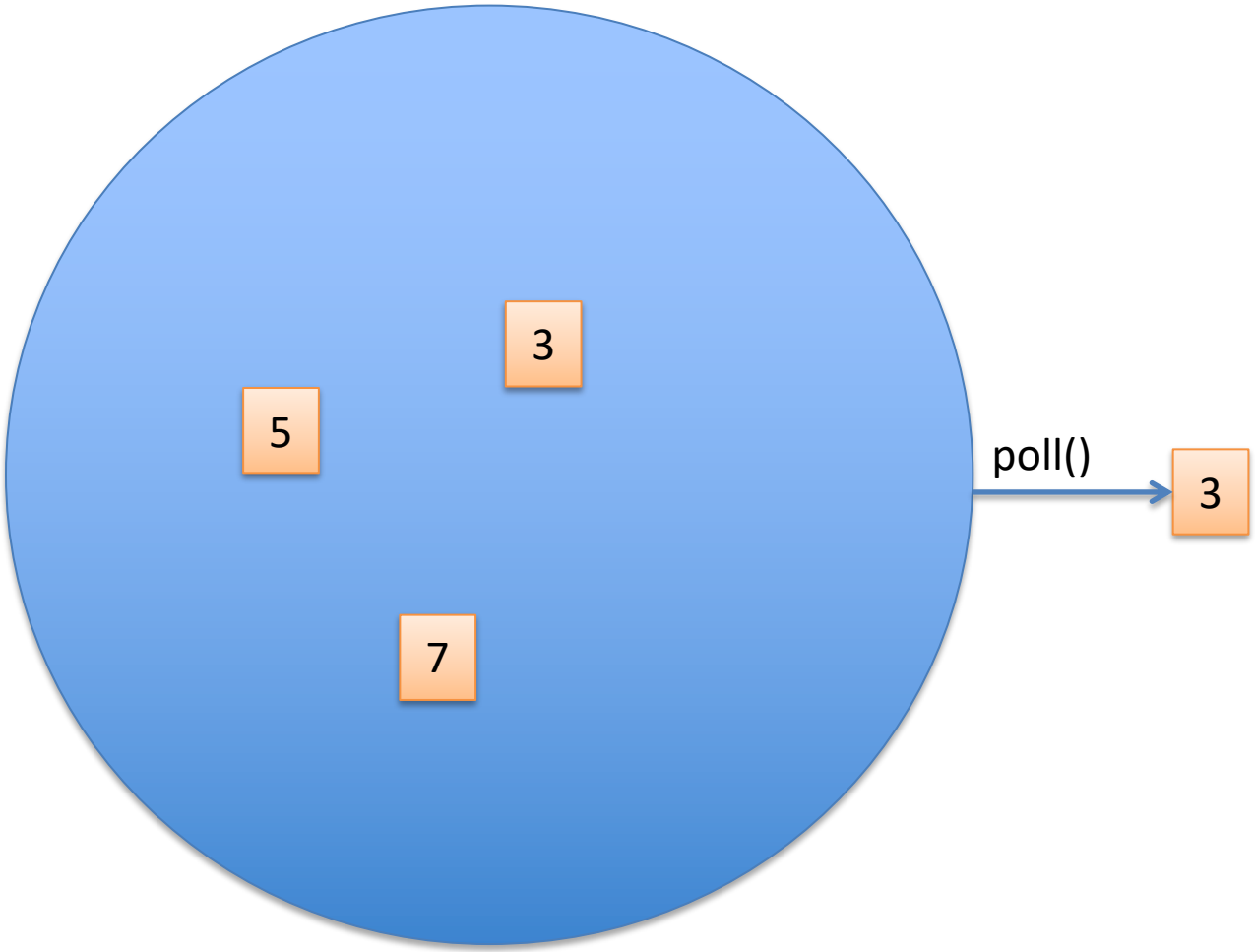
# A Priority Queue in Picture



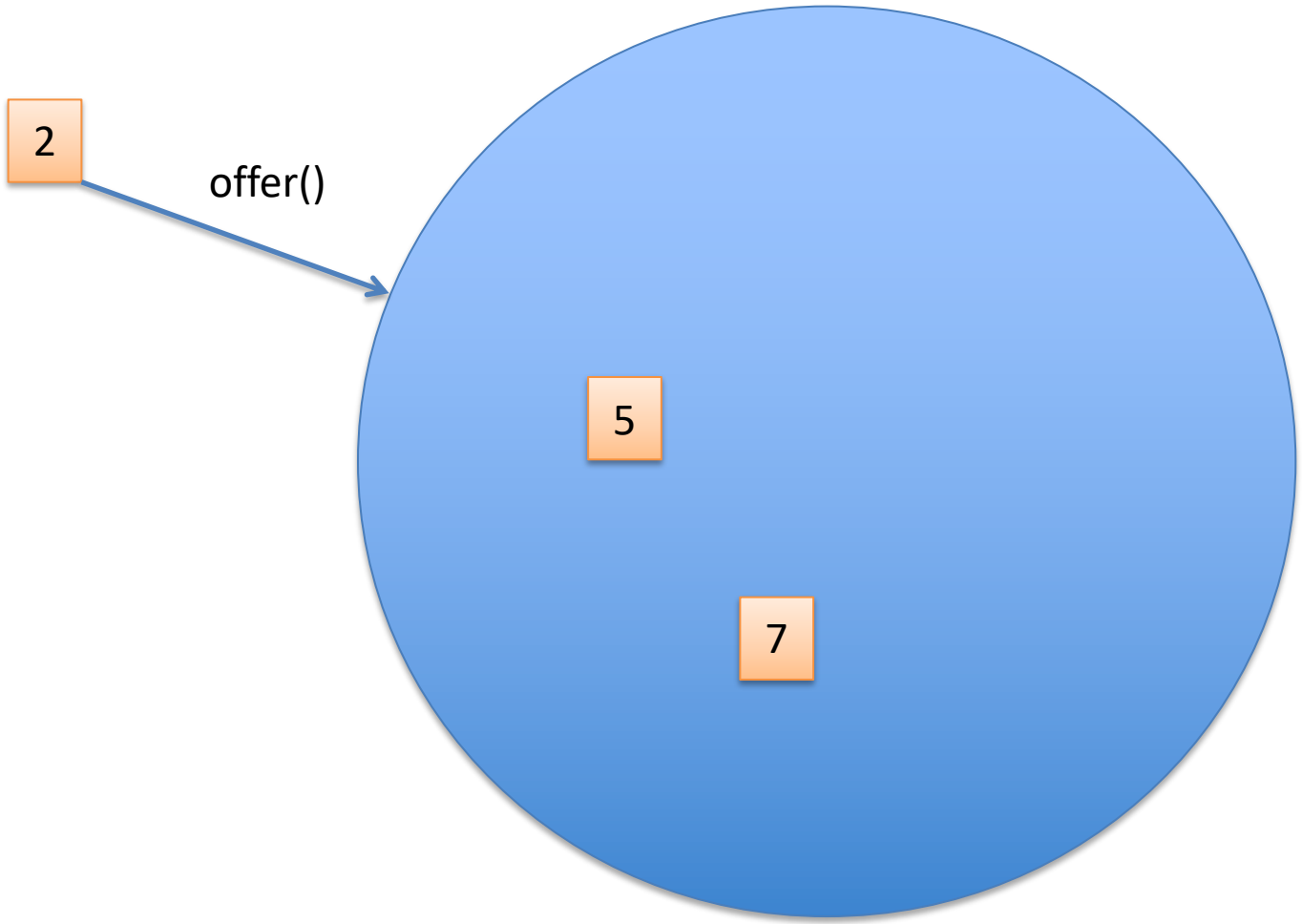
# A Priority Queue in Picture



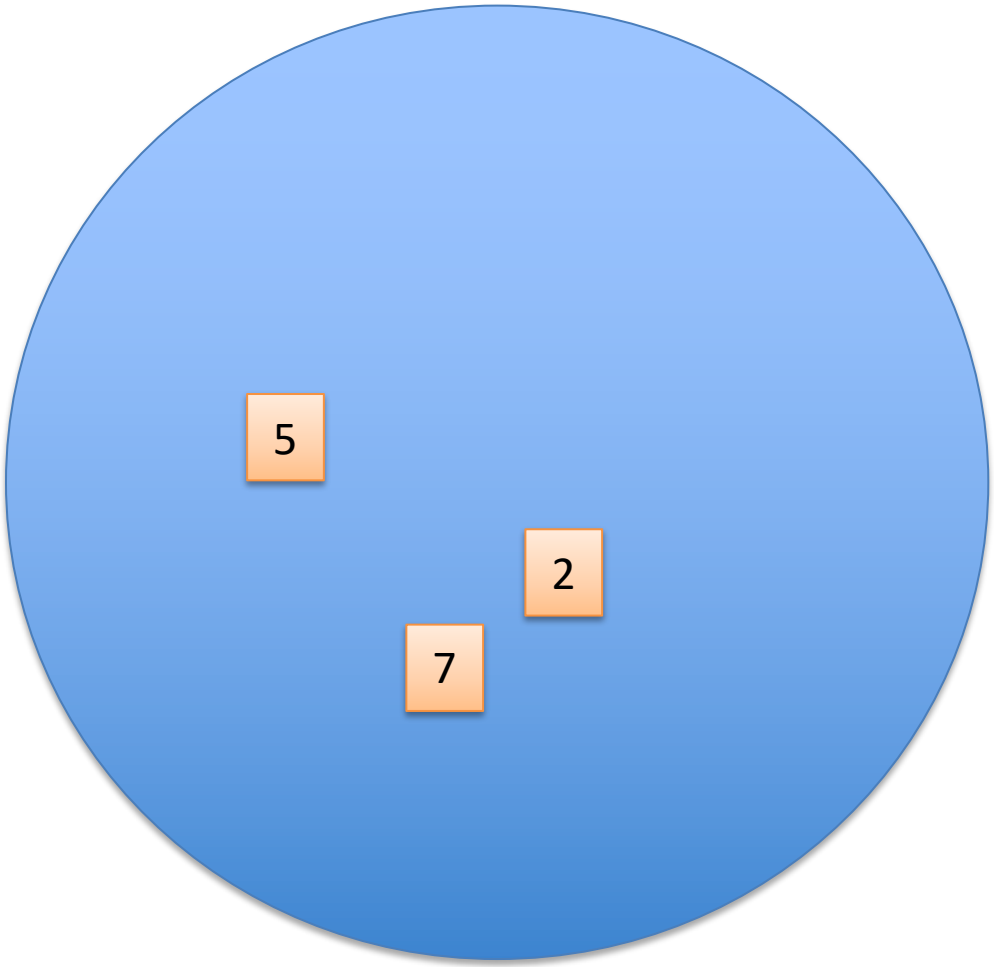
# A Priority Queue in Picture



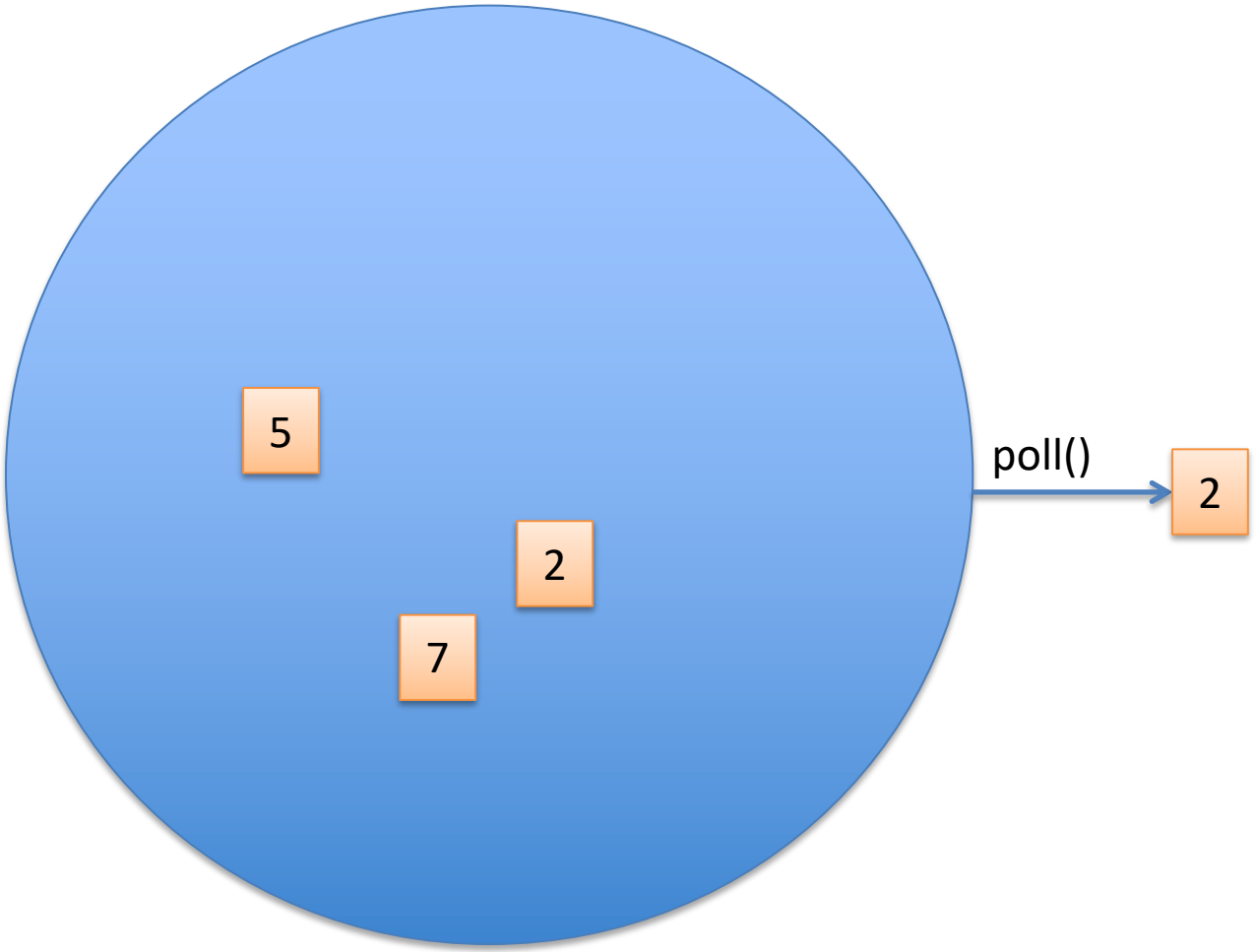
# A Priority Queue in Picture



# A Priority Queue in Picture



# A Priority Queue in Picture



# Code

```
public static void main(String args[])
   {
       // Creating empty priority queue
       PriorityQueue<Integer> pq =
                          new PriorityQueue<Integer>();

       // Adding items to the pq
       pq.offer(7);
       pq.offer(3);
       pq.offer(5);

       // Printing the highest priority element
       System.out.println("Head value :"+ pq.peek());

       // Printing all elements
       System.out.println("The pq elements:");
       Iterator itr = pq.iterator();
       while (itr.hasNext())
           System.out.println(itr.next());

       // Removing the top priority element (or head) and
       // printing the modified pQueue
       pq.poll();
       System.out.println("After removing an element");
       itr = pq.iterator();
       while (itr.hasNext())
           System.out.println(itr.next());

       pq.offer(2);
       System.out.println("After adding 2");
       itr = pq.iterator();
       while (itr.hasNext())
           System.out.println(itr.next());
   }
```

```
Head value using peek function:3
The pq elements:
3
7
5
After removing an element
5
7
After adding 2
2
7
5
```

How does PriorityQueue compare elements?

Comparable Interface

But what about objects whose class does not implement Comparable Interface?

What about if I want to change the priority of order:
Ascending to Descending

Comparator Interface

# Custom Comparator

```
public class Desc implements Comparator<Integer>
{
    // Used for sorting in ascending order of
    // roll name
    public int compare(Integer a, Integer b)
    {
    int i = 0;
        if (a < b)  return 1;
        if ( a > b) return -1;
        return 0;
    }
}
```

# Output

```
PriorityQueue<Integer> pq =
•                   new PriorityQueue<Integer>(10, new Desc());
```

```
Head value using peek function:7
The pq elements:
7
3
5
After removing an element
5
3
After adding 2
5
3
2
```

# Comparable vs Comparator

- Comparable:
  - Compares itself
  - must implements the java.lang.Comparable interface

- Comparator:
  - Compares others
  - must implements java.util.Comparator interface

- Ref: http://www.digizol.com/2008/07/java-sorting-comparator-vs-comparable.html

# HASHMAP

# Example

```
static void printFreq(int arr[])
    {
        // Creates an empty HashMap
        HashMap<Integer, Integer> hmap =
                    new HashMap<Integer, Integer>();

        // Traverse through the given array
        for (int i = 0; i < arr.length; i++)
        {
            Integer c = hmap.get(arr[i]);

            // If this is first occurrence of element
            if (hmap.get(arr[i]) == null)
               hmap.put(arr[i], 1);

            // If elements already exists in hash map
            else
              hmap.put(arr[i], ++c);
        }

        // Print result
        for (Map.Entry m:hmap.entrySet())
          System.out.println("Frequency of " + m.getKey() +
                             " is " + m.getValue());
    }
```

```
int arr[] = {2, 4, 5, 4, 3, 3, 4};
```

```
Frequency of 2 is 1
Frequency of 3 is 2
Frequency of 4 is 3
Frequency of 5 is 1
```

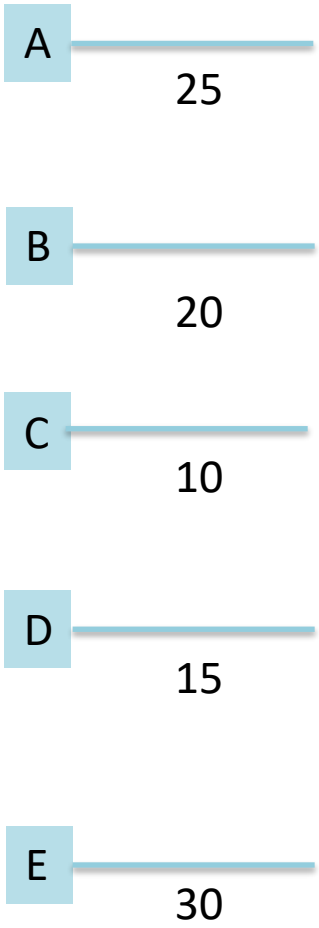
http://www.geeksforgeeks.org/hashmap-treemap-java/

# HUFFMAN CODING

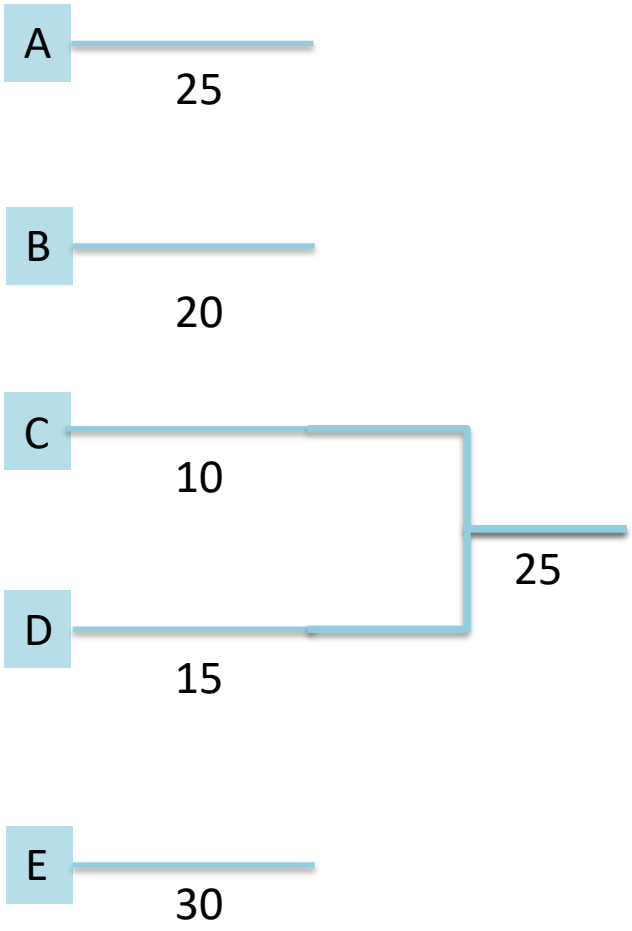
# Start with Frequency Table

| Letters | Frequency | Code |
|---|---|---|
| A | 25 | ? |
| B | 20 | ? |
| C | 10 | ? |
| D | 15 | ? |
| E | 30 | ? |

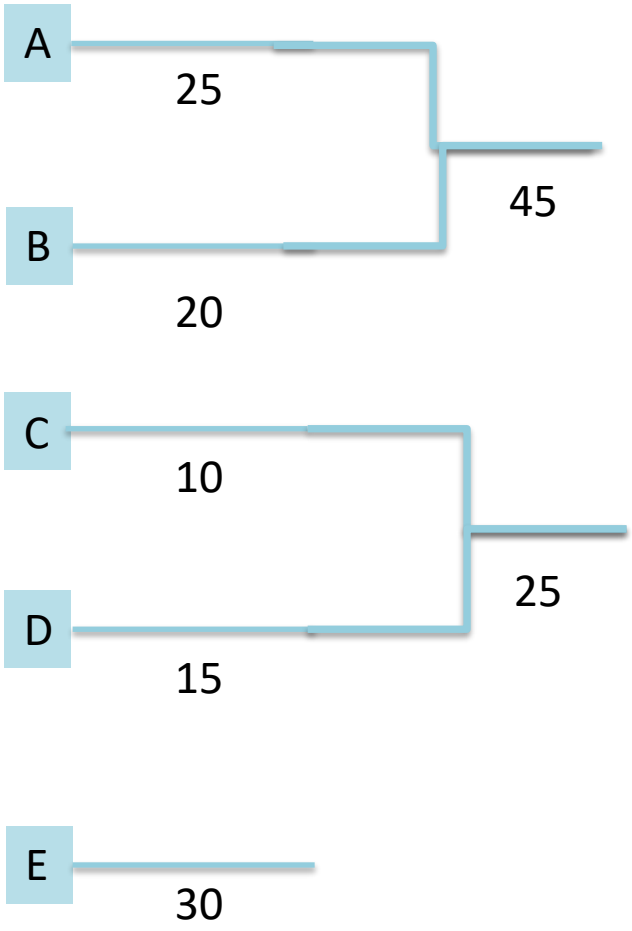
# Generate Codes



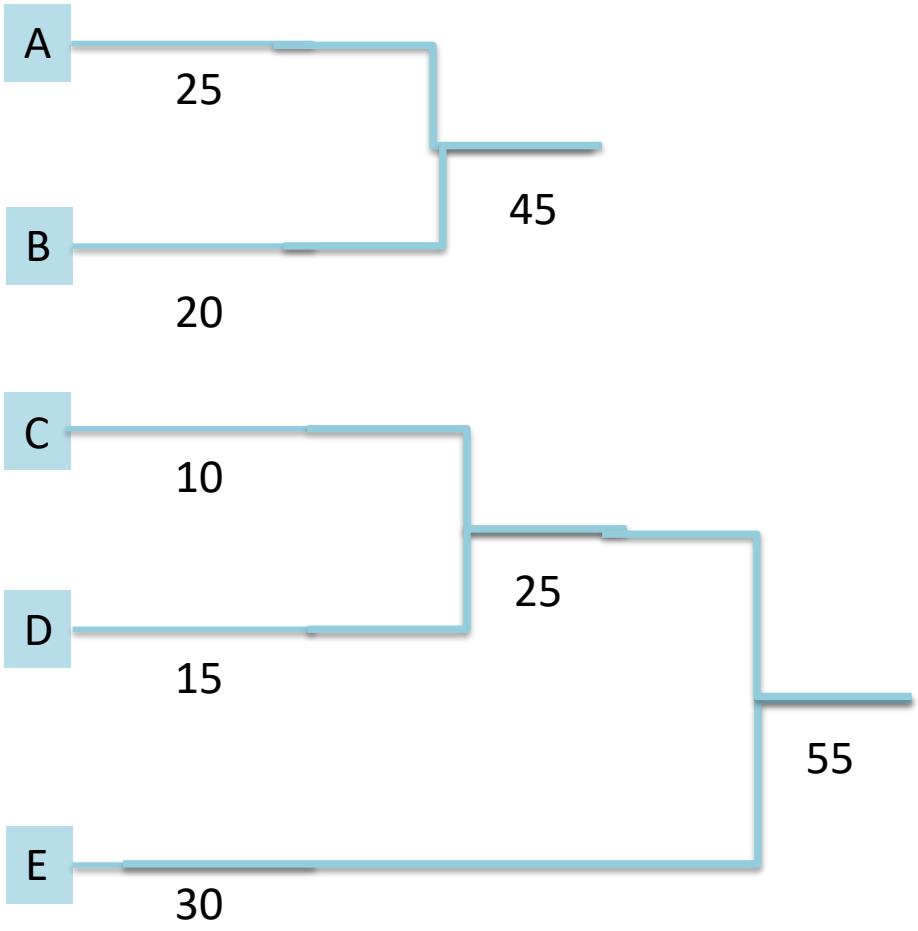
# Generate Codes



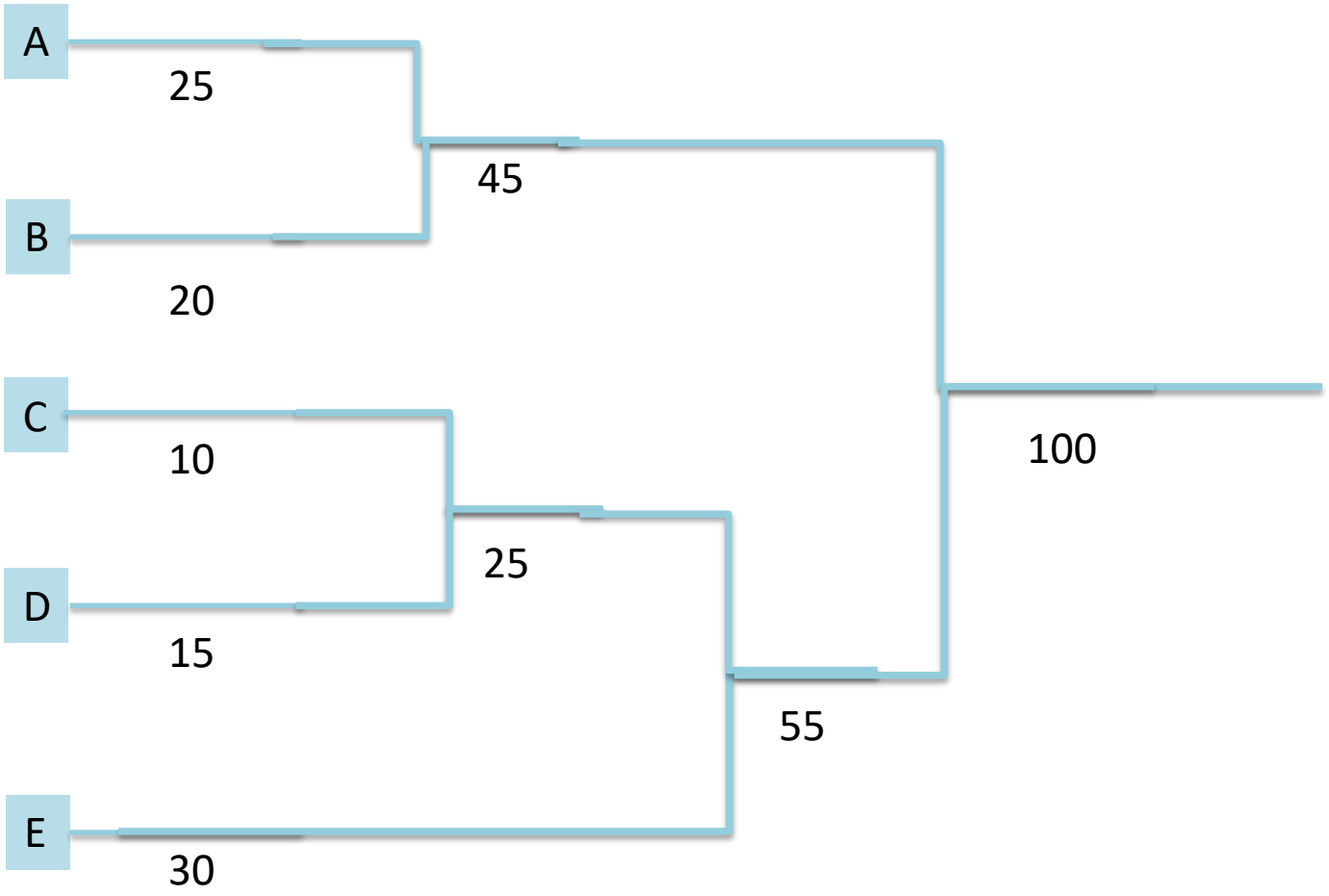
# Generate Codes



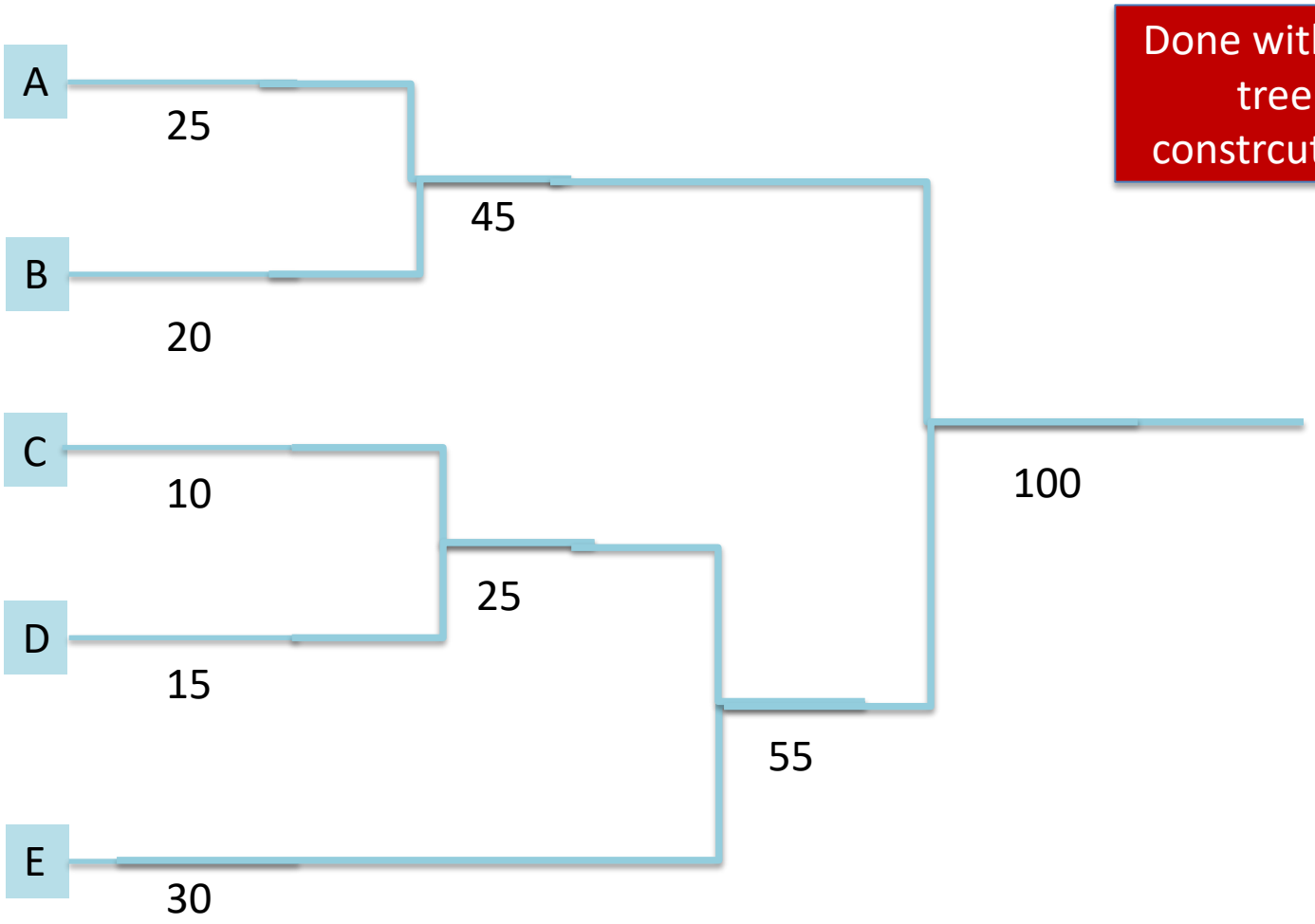
# Generate Codes



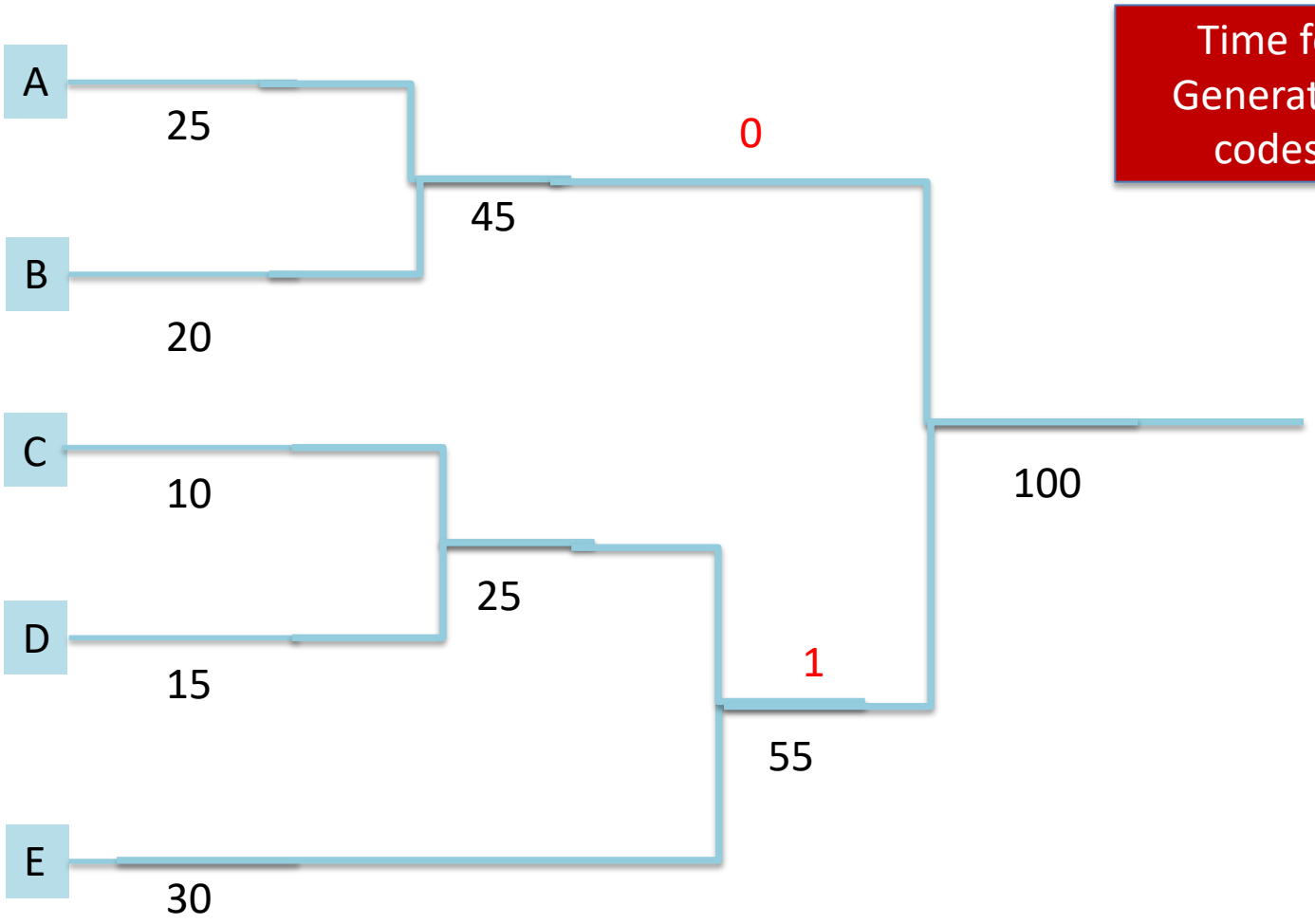
# Generate Codes



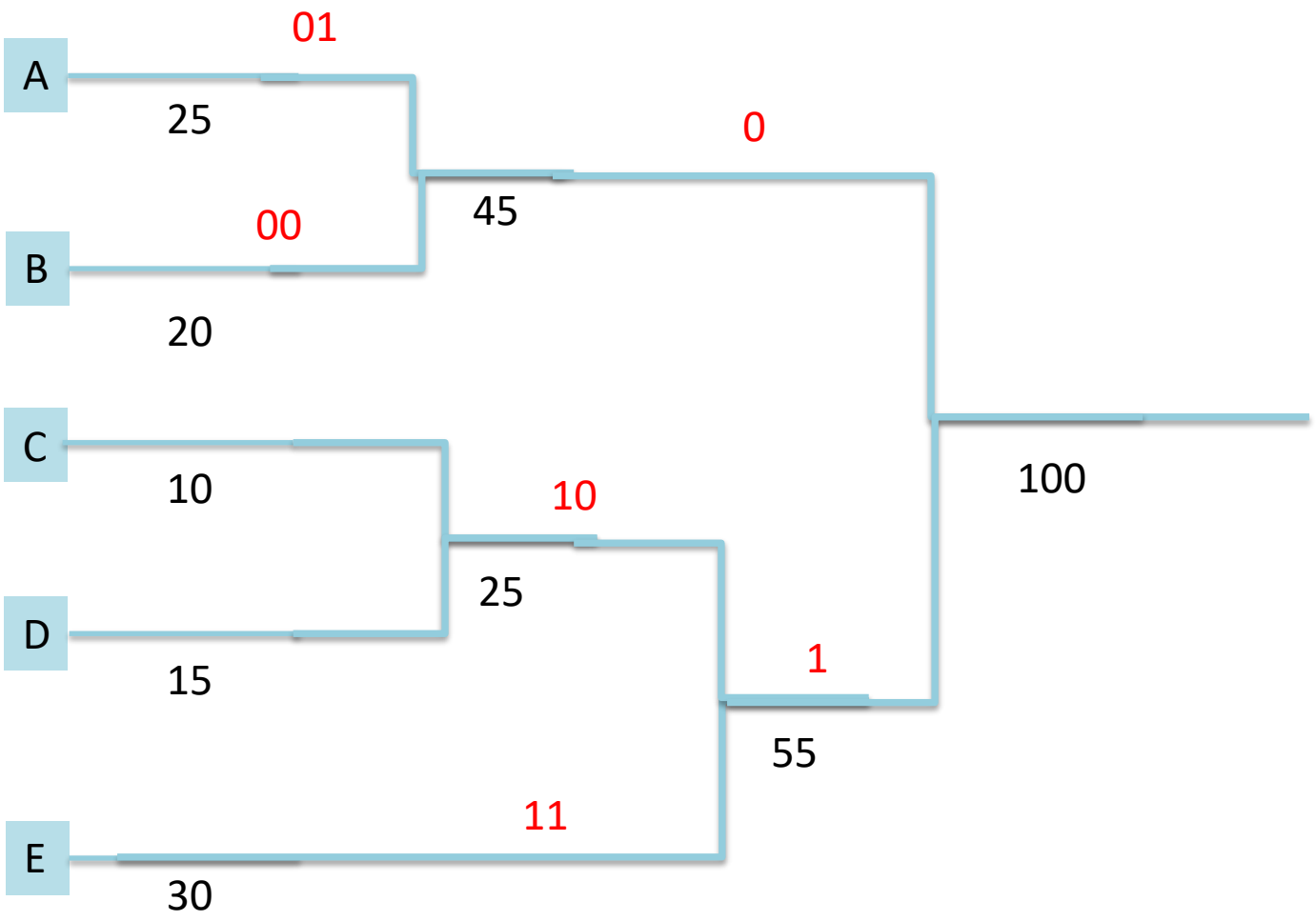
# Generate Codes



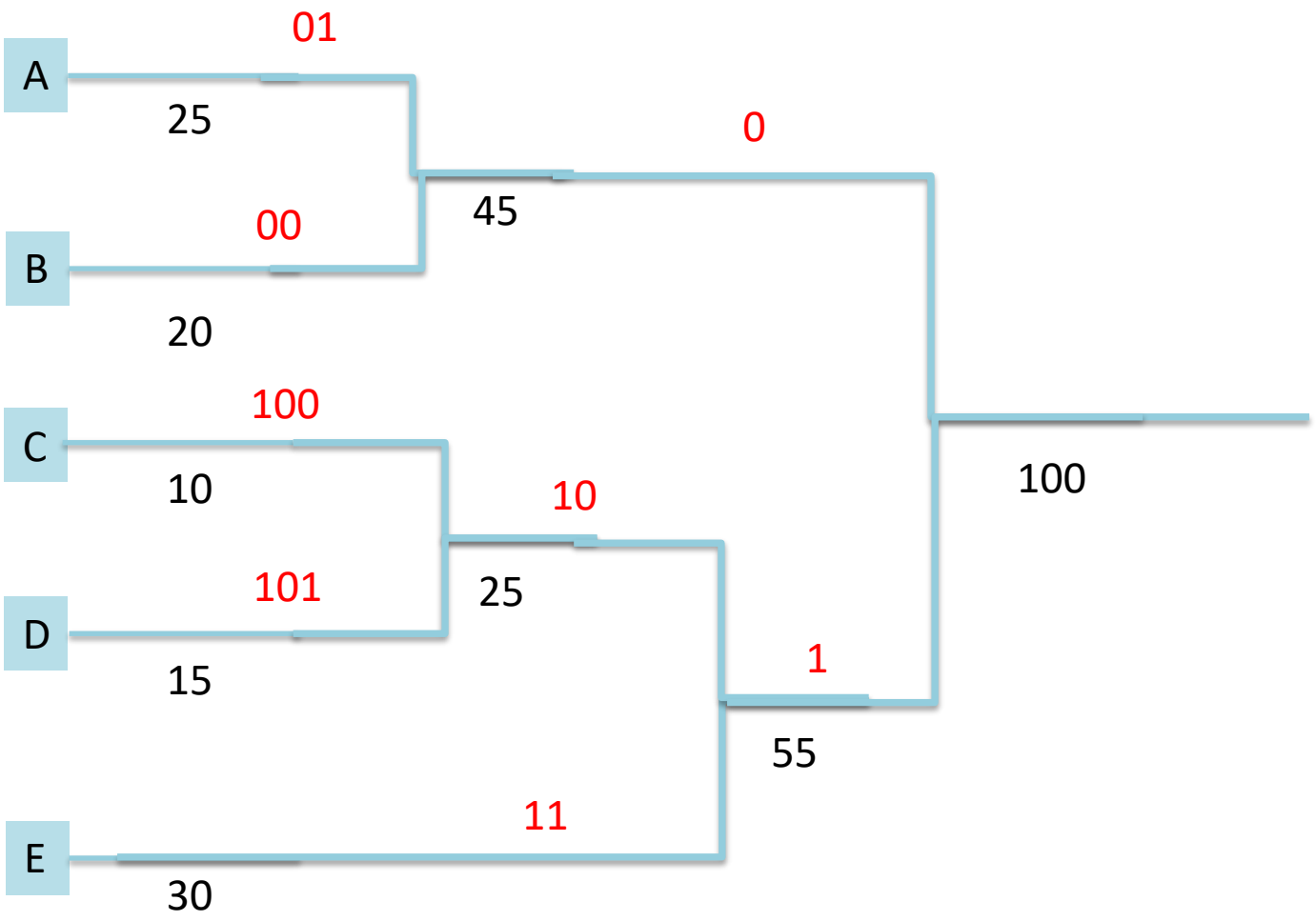
# Generate Codes



# Generate Codes



# Generate Codes



# Huffman Coding

| Letters | Frequency | Code |
|---|---|---|
| A | 25 | 01 |
| B | 20 | 00 |
| C | 10 | 100 |
| D | 15 | 101 |
| E | 30 | 11 |

00011011000100

BADCAB

# Let's do another

| Letters | Frequency | Code |
|---|---|---|
| A | | |
| B | | |
| C | | |
| D | | |
| E | | |

# How to implement in Java

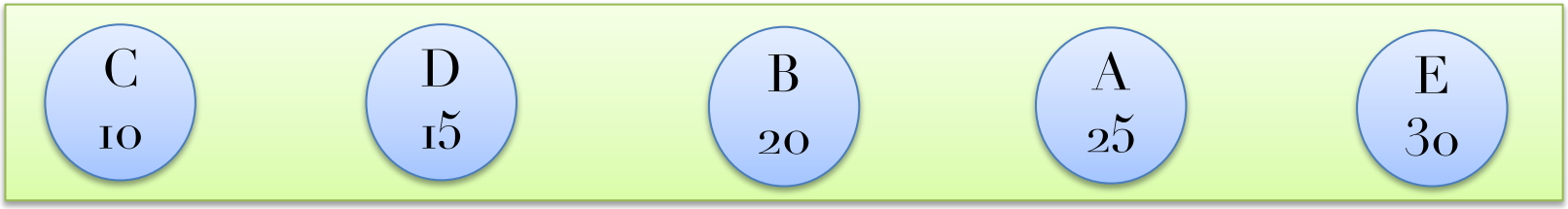
- Priority Queue and Tree Node

# Recall: How to construct Expression Trees?

4 3 2 + * 6 3 – 5 * 3 / -

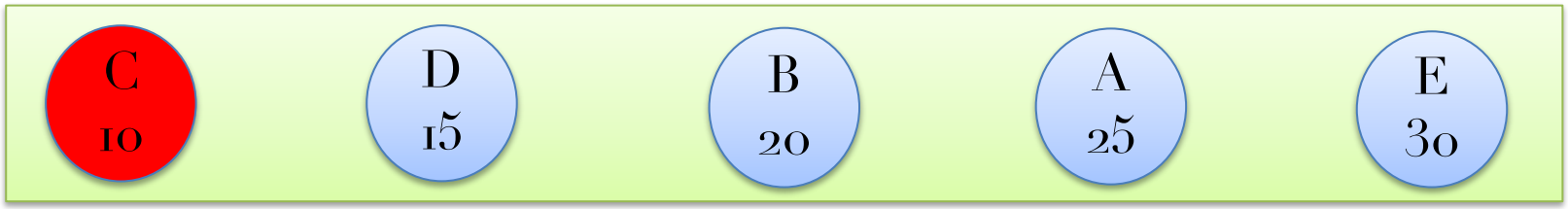

Similar Idea. Instead of using a Stack we will use a Priority Queue

# Huffman Code in Java

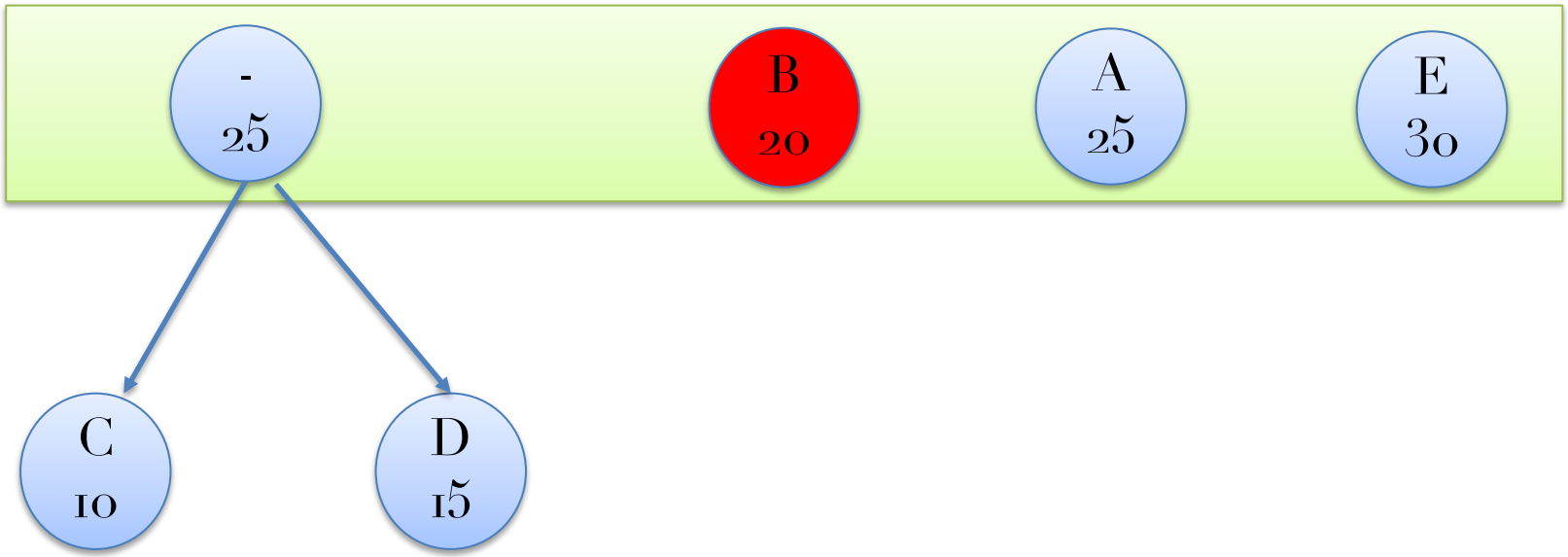



Similar Idea. Instead of using a Stack we will use a Priority Queue

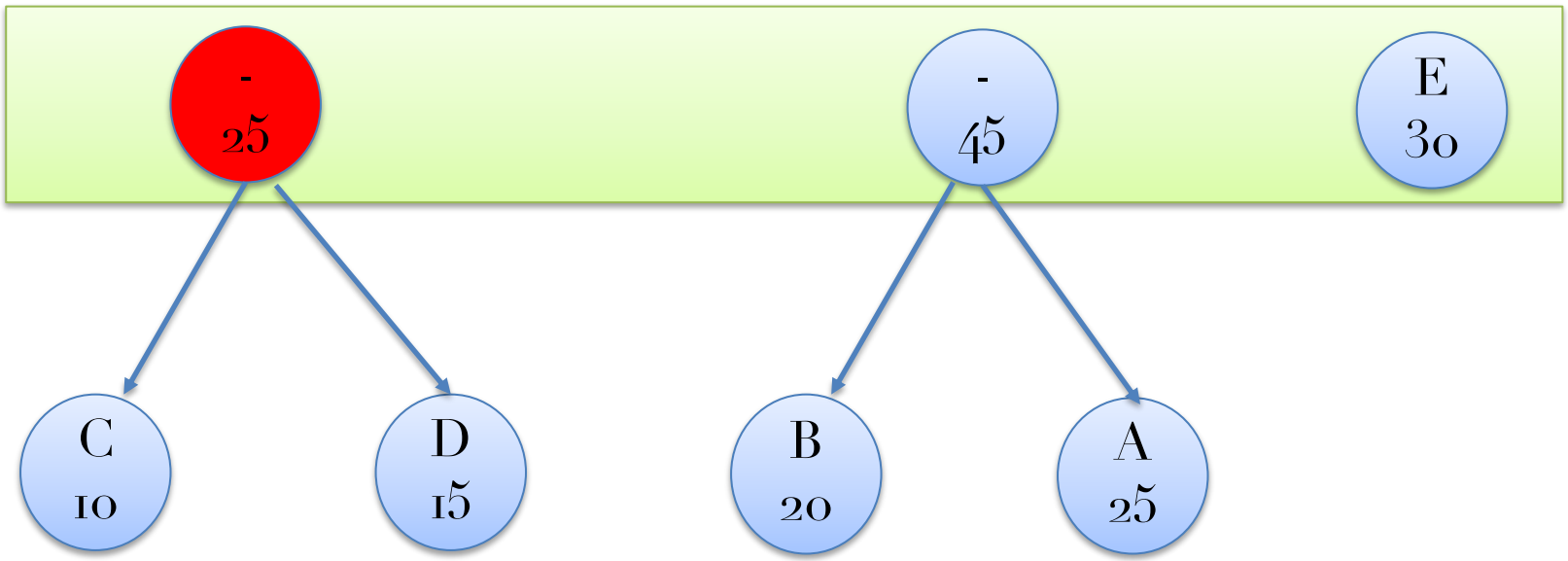
# Huffman Code in Java



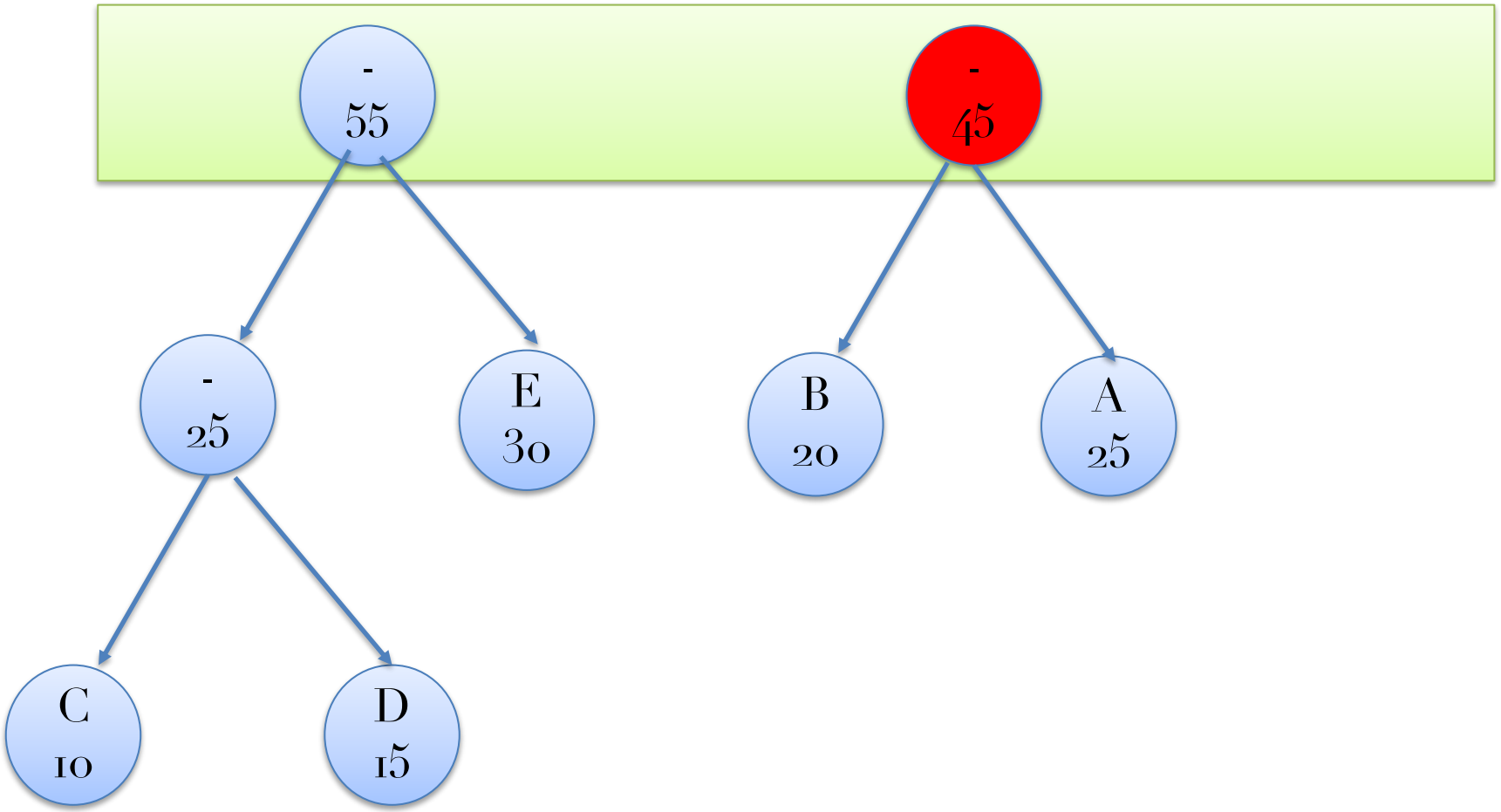
# Huffman Code in Java



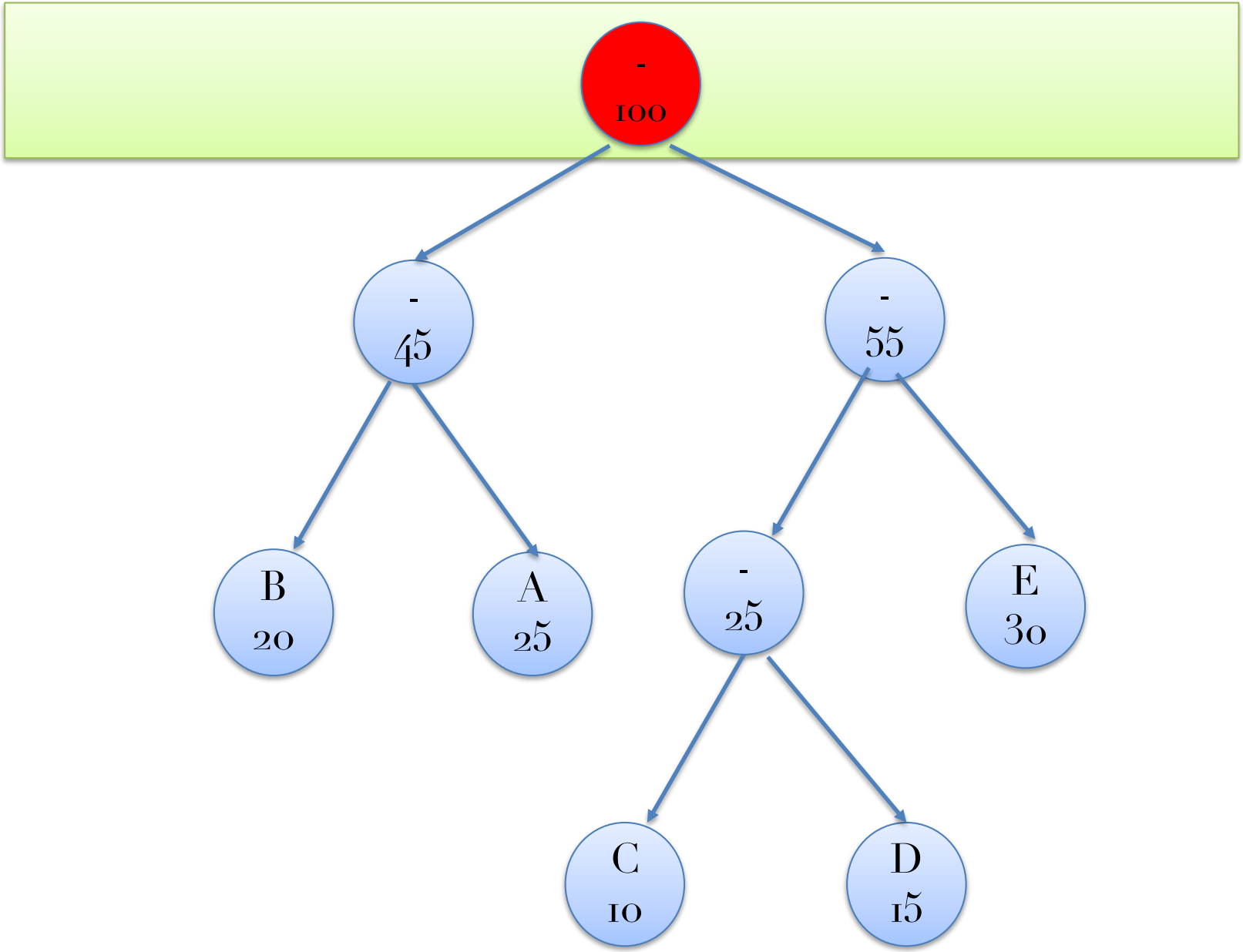
# Huffman Code in Java



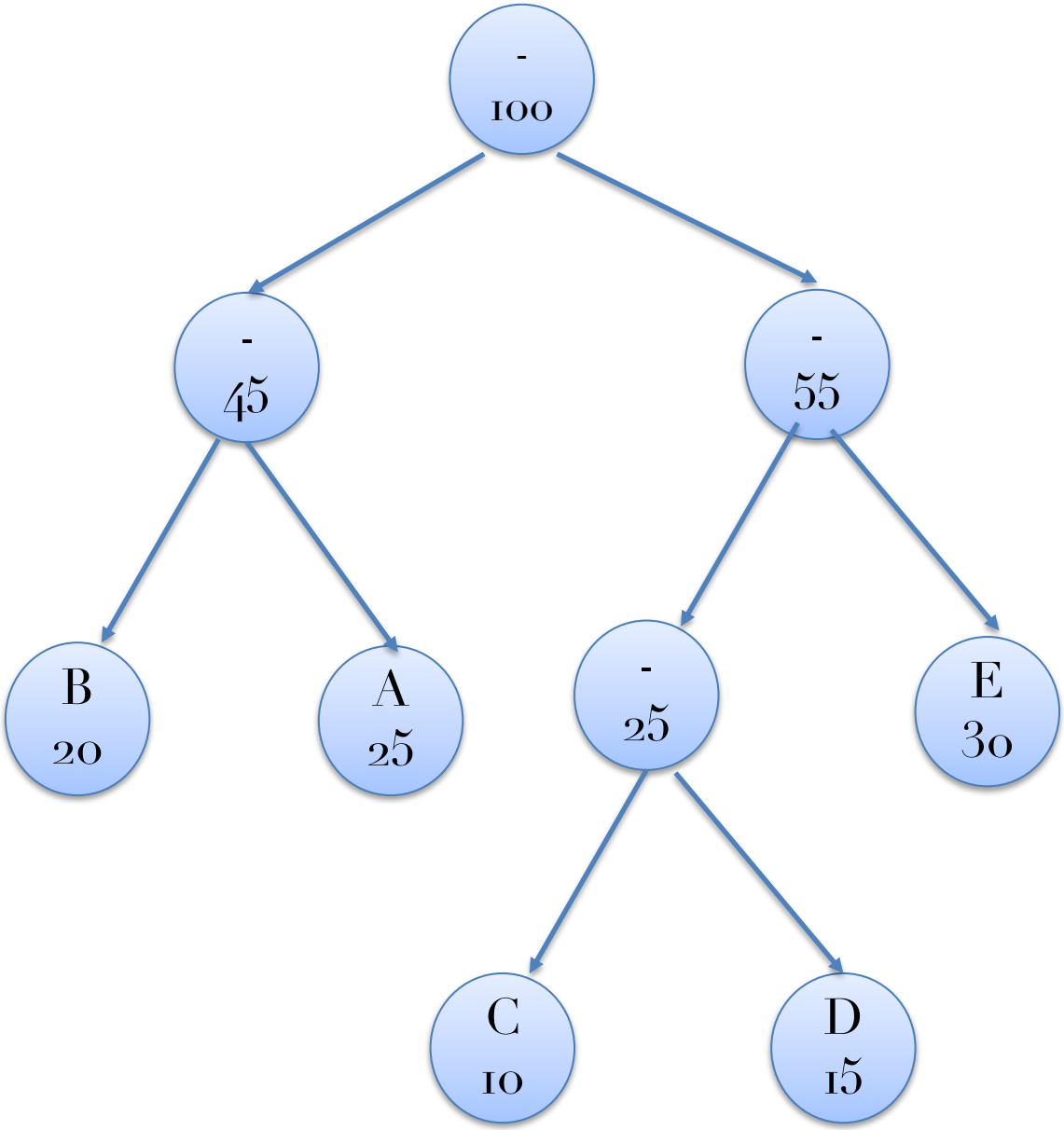
# Huffman Code in Java



# Huffman Code in Java

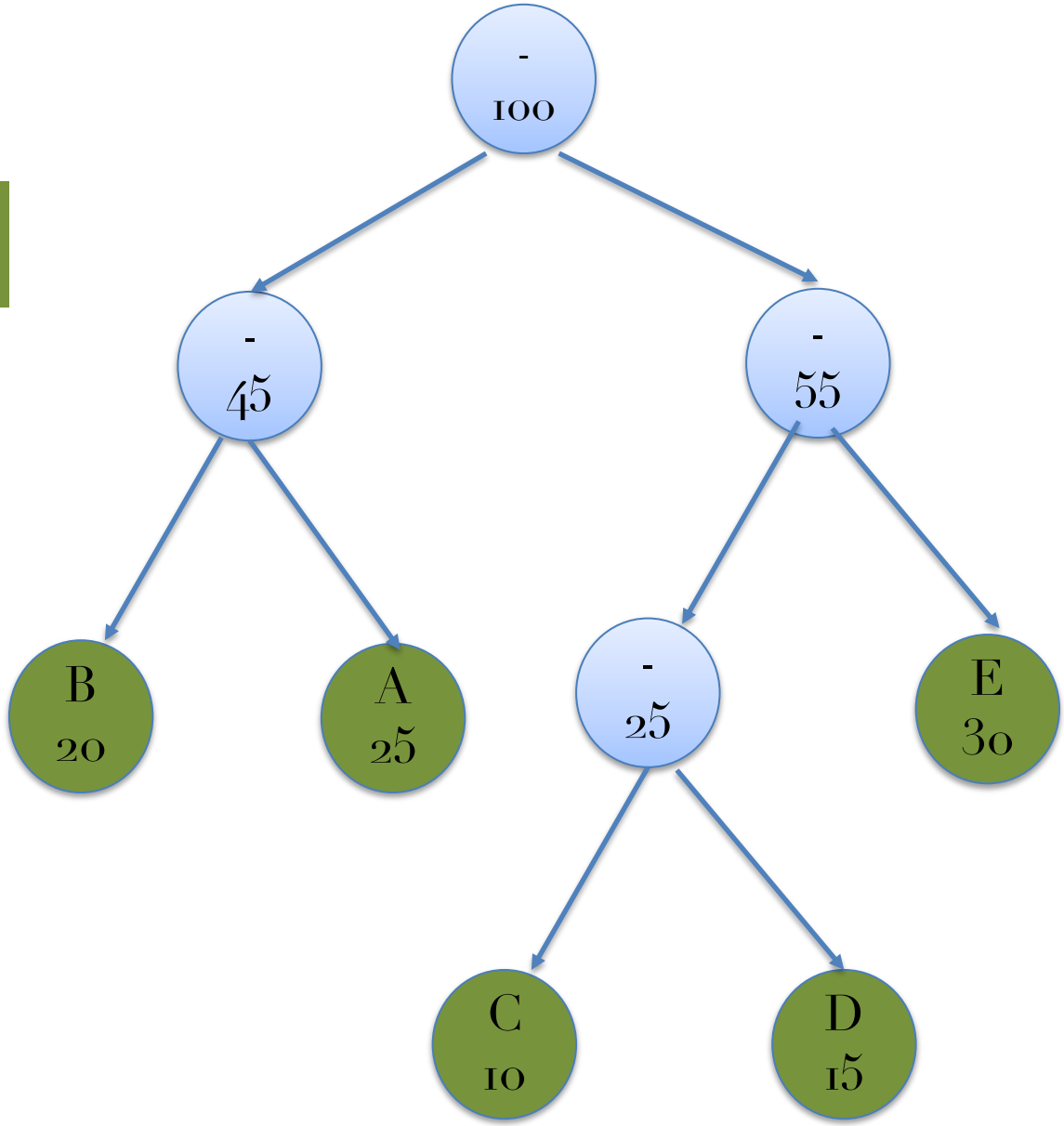


# Huffman Code in Java



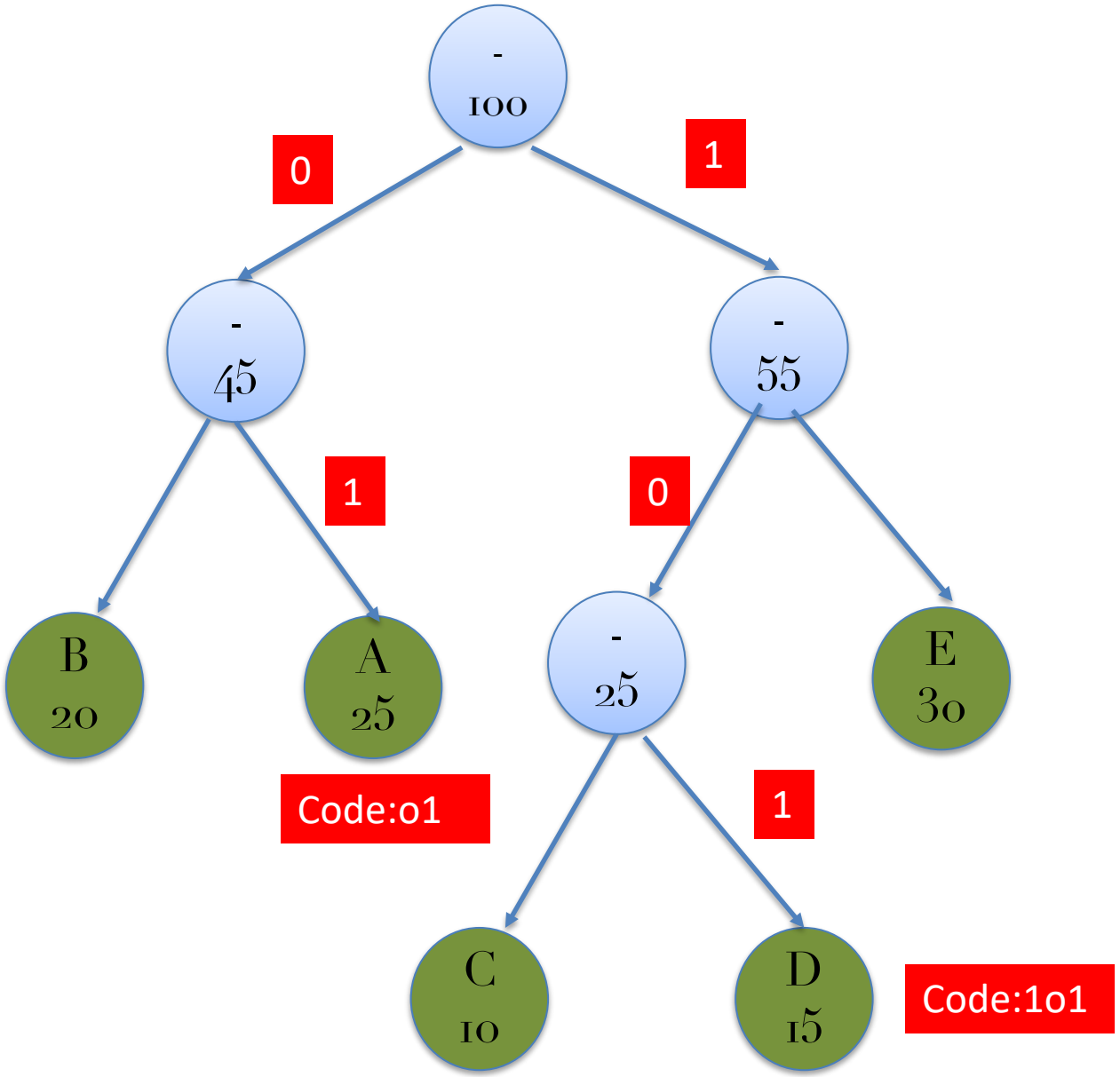
# Huffman Code in Java

We care about only the leaves



# Huffman Code in Java

Left 0
Right 1



# Huffman Coding

| Letters | Frequency | Code |
|---|---|---|
| A | 25 | 01 |
| B | 20 | 00 |
| C | 10 | 100 |
| D | 15 | 101 |
| E | 30 | 11 |

00011011000100

BADCAB

# Huffman Coding and Project 3

- Please go through:
- http://lti.cs.vt.edu/LTI_ruby/Books/CS172/html/Huffman.html