**Administrative**

**Today’s session**

Homework 5 key

Recursion

Merge sort

Quick sort

Stack using linked list

**Session Topics**

**Homework 5 key**

● The Homework 5 key is available on Blackboard.

**Recursion**

● **Recursion** is the process of a method calling itself.

● A recursive method has one or more:

✓ **Base cases** that signal the end of the recursion.

✓ **Recursive cases** in which the method calls itself with a subset of the task.

● A stack is used to handle a recursive operation.

● Recursion with an improper/missing base case will result in infinite recursion. This may generate an StackOverflowError or other exception.

● Some programming tasks may be done using either iteration or recursion.

● Generally, iteration is easier to read and debug.

● A recursive method may be tail-recursive. This means that the last statement executed by the method is a call to itself.

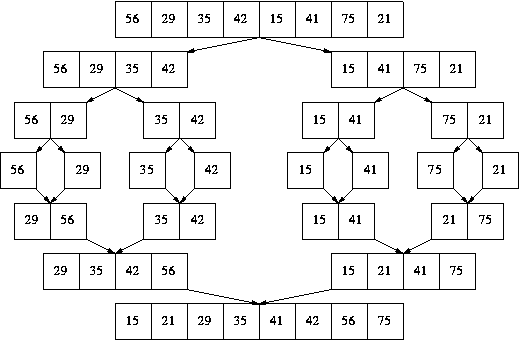
● A tail-recursive method can save stack space.

● See **Recursion – factorial** sample application on Blackboard.

● See **Recursion – folder tree** sample application on Blackboard.

**Merge sort**

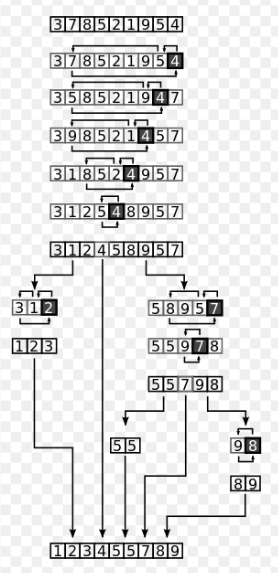
● The **merge sort** recursively splits the array into single-element arrays and then recombines them to form the sorted array.



**Quick sort**

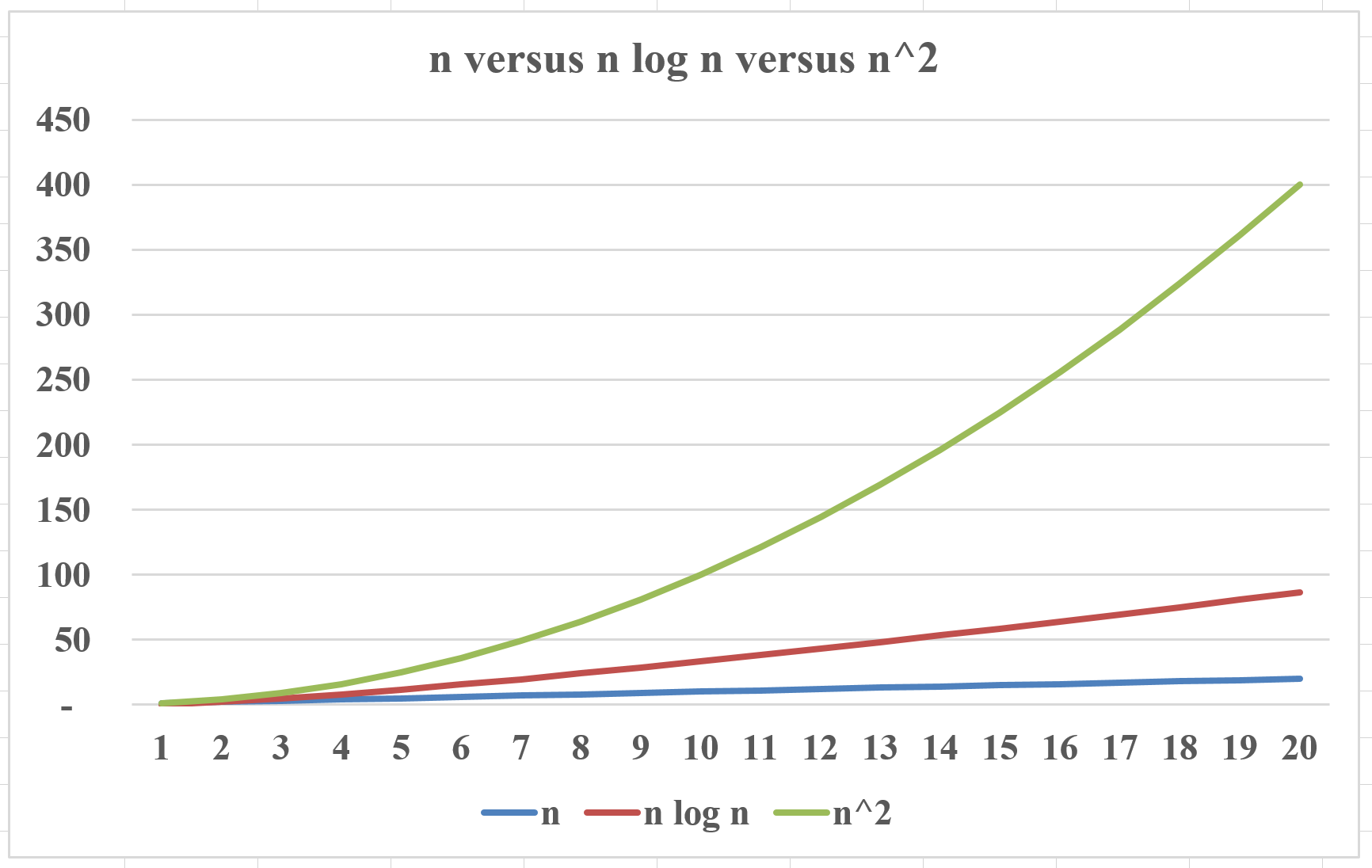
● The **quick sort** partitions the array around a pivot value. It then recursively quick sorts each side of the array around the pivot value.

● The base case of the recursion is when an array has size zero or one.



● Here is a comparison of the three non-recursive sorting methods and the two recursive ones for an array of n elements:

|  |  |  |  |
| --- | --- | --- | --- |
| Sort method | Best case | Average case | Worst case |
| Insertion | n | n2 | n2 |
| Selection | n2 | n2 | n2 |
| Bubble | n | n2 | n2 |
| Merge | n log n | n log n | n log n |
| Quick | n log n | n log n | n2  (pivot is leftmost/rightmost element on already sorted list) |



● See **Array Sorting – recursive** sample application on Blackboard.

**Stack (reprise)**

● A **stack** is a data structure that stores data as a series of linked nodes.

● A stack may contain zero or more linked nodes.

● A node is a packet of information containing:

✓ At least one datum.

✓ A link to the next node in the stack, or **null** if the node is the bottom of the stack.

● The link to the first node of the stack is called the **top**.

● The top is the only connection between the stack and the outside world.

● A stack is considered last-in first-out (LIFO).

● There are two possible operations on a stack:

✓ Push – add a node to the top of the stack.

✓ Pop – if the stack is not empty, remove a node from the top of the stack.

● See **Stack using linked list** sample application on Blackboard.