NPTEL MOOC,JAN-FEB 2015
Week 2, Module 7

DESIGN AND ANALYSIS OF ALGORITHMS

Quicksort

MADHAVAN MUKUND, CHENNAI MATHEMATICAL INSTITUTE
http://www.cmi.ac.in/~madhavan

Merge Sort: Shortcomings

- Merging A and B creates a new array C
  - No obvious way to efficiently merge in place
- Extra storage can be costly
- Inherently recursive
  - Recursive call and return are expensive

Alternative approach

- Extra space is required to merge
- Merging happens because elements in left half must move right and vice versa
- Can we divide so that everything to the left is smaller than everything to the right?
  - No need to merge!

Divide and conquer without merging

- Suppose the median value in A is m
- Move all values ≤ m to left half of A
  - Right half has values > m
  - This shifting can be done in place, in time O(n)
- Recursively sort left and right halves
- A is now sorted! No need to merge
  - t(n) = 2t(n/2) + n = O(n log n)

Divide and conquer without merging

- How do we find the median?
  - Sort and pick up middle element
  - But our aim is to sort!
- Instead, pick up some value in A — pivot
  - Split A with respect to this pivot element

Quicksort

- Choose a pivot element
  - Typically the first value in the array
- Partition A into lower and upper parts with respect to pivot
- Move pivot between lower and upper partition
- Recursively sort the two partitions

Quicksort

- High level view

Quicksort

- High level view

43	32	22	78	63	57	91	13

Quicksort

- High level view

43	32	22	78	63	57	91	13

Quicksort

- High level view

43	32	22	78	63	57	91	13

Quicksort

- High level view

13	32	22	43	63	57	91	78

Quicksort

- High level view

13	22	32	43	57	63	78	91

Quicksort: Partitioning

Quicksort: Partitioning

43	32	22	78	63	57	91	13

Quicksort: Partitioning

Quicksort: Partitioning

43	32	22	78	63	57	91	13

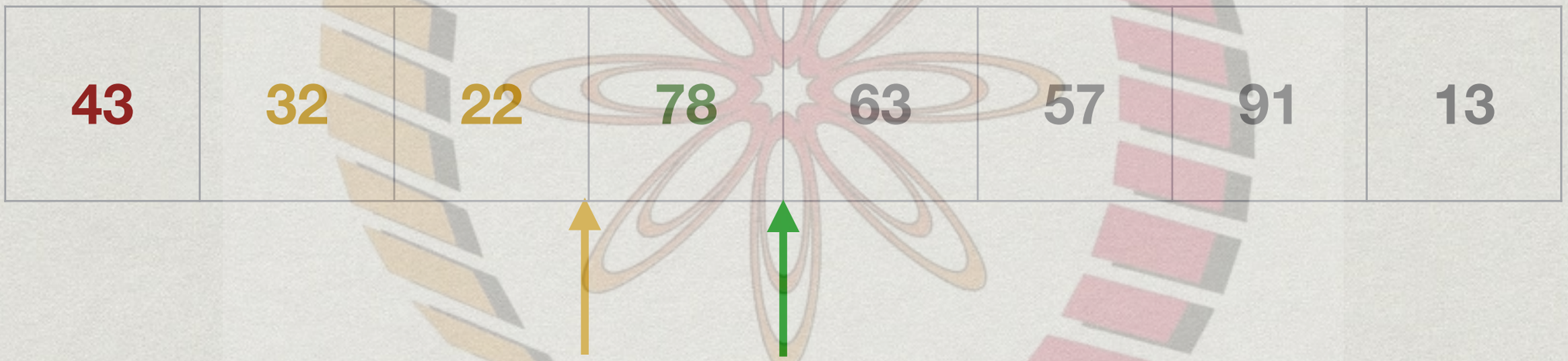
Quicksort: Partitioning

43	32	22	78	63	57	91	13

Quicksort: Partitioning

43	32	22	78	63	57	91	13

Quicksort: Partitioning



Quicksort: Partitioning

Quicksort: Partitioning

Quicksort: Partitioning

Quicksort: Partitioning

Quicksort: Partitioning

Quicksort: Partitioning

Quicksort: Implementation

```
Quicksort(A,l,r) // Sort A[l..r-1]

    if (r - l <= 1)) return; // Base case

    // Partition with respect to pivot, a[l]
    yellow = l+1;
    for (green = l+1; green < r; green++)
        if (A[green] <= A[l])
            swap(A,yellow,green);
            yellow++;

    swap(A,l,yellow-1); // Move pivot into place

    Quicksort(A,l,yellow);  // Recursive calls
    Quicksort(A,yellow+1,r);
```

Quicksort:
Another Partitioning Strategy

Quicksort: Another Partitioning Strategy

43	32	22	78	63	57	91	13

Quicksort:
Another Partitioning Strategy

Quicksort: Another Partitioning Strategy

Quicksort: Another Partitioning Strategy

43	32	22	78	63	57	91	13

Quicksort: Another Partitioning Strategy

43	32	22	13	63	57	91	78

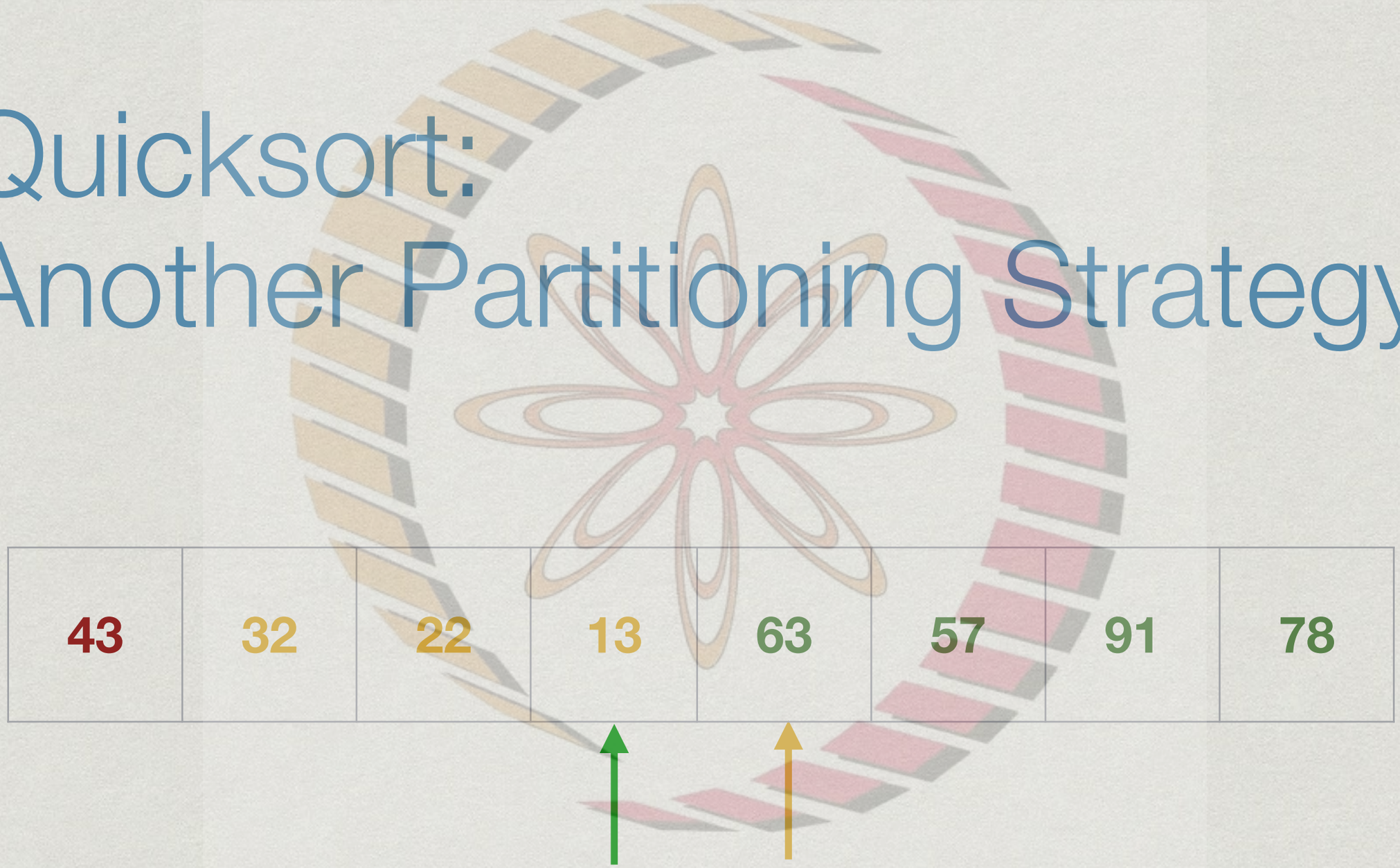
Quicksort:
Another Partitioning Strategy

43	32	22	13	63	57	91	78

Quicksort: Another Partitioning Strategy

43	32	22	13	63	57	91	78

Quicksort:
Another Partitioning Strategy



Quicksort: Another Partitioning Strategy