The background features a large, semi-transparent watermark of the NPTEL logo. It consists of a circular emblem with a stylized flower or sunburst in the center, surrounded by a ring of colored segments. Below the emblem, the word "NPTEL" is written in large, bold, sans-serif capital letters.

NPTEL MOOC, JAN-FEB 2015
Week 2, Module 3

DESIGN AND ANALYSIS OF ALGORITHMS

Selection Sort

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

Sorting

- * Searching for a value
 - * Unsorted array — linear scan, $O(n)$
 - * Sorted array — binary search, $O(\log n)$
- * Other advantages of sorting
 - * Finding **median** value: midpoint of sorted list
 - * Checking for duplicates
 - * Building a frequency table of values

How to sort?

- * You are a Teaching Assistant for a course
- * The instructor gives you a stack of exam answer papers with marks, ordered randomly
- * Your task is to arrange them in descending order

NPTTEL

Strategy 1

- * Scan the entire stack and find the paper with minimum marks
- * Move this paper to a new stack
- * Repeat with remaining papers
 - * Each time, add next minimum mark paper on top of new stack
- * Eventually, new stack is sorted in descending order

Strategy 1 ...

74

32

89

55

21

64

NPTTEL

Strategy 1 ...

74

32

89

55

~~21~~

64

21

NPTTEL

Strategy 1 ...

74

32

89

55

21

64

21

32

NPTTEL

Strategy 1 ...

74

32

89

55

21

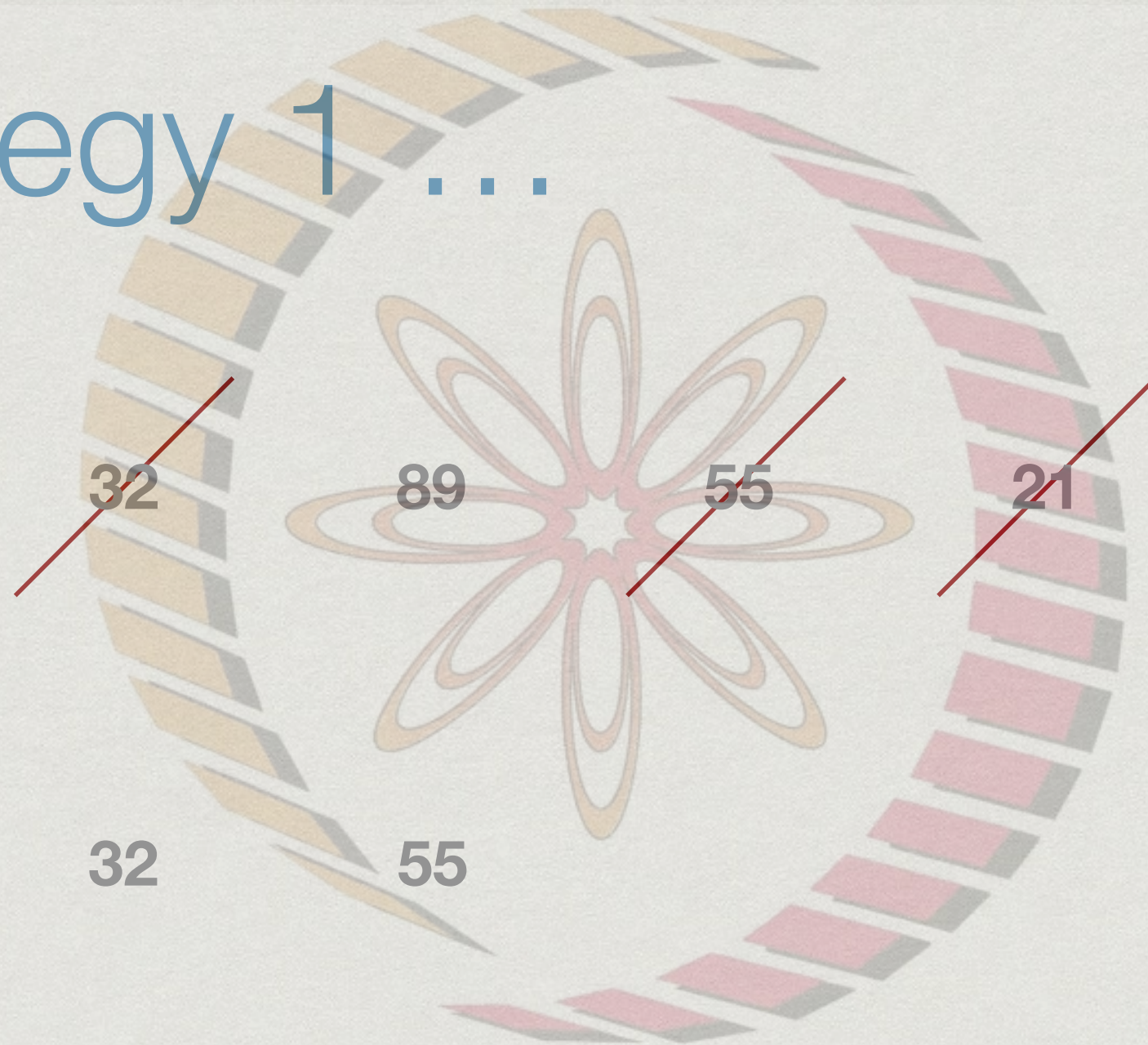
64

21

32

55

NPTTEL



Strategy 1 ...

74

32

89

55

21

64

21

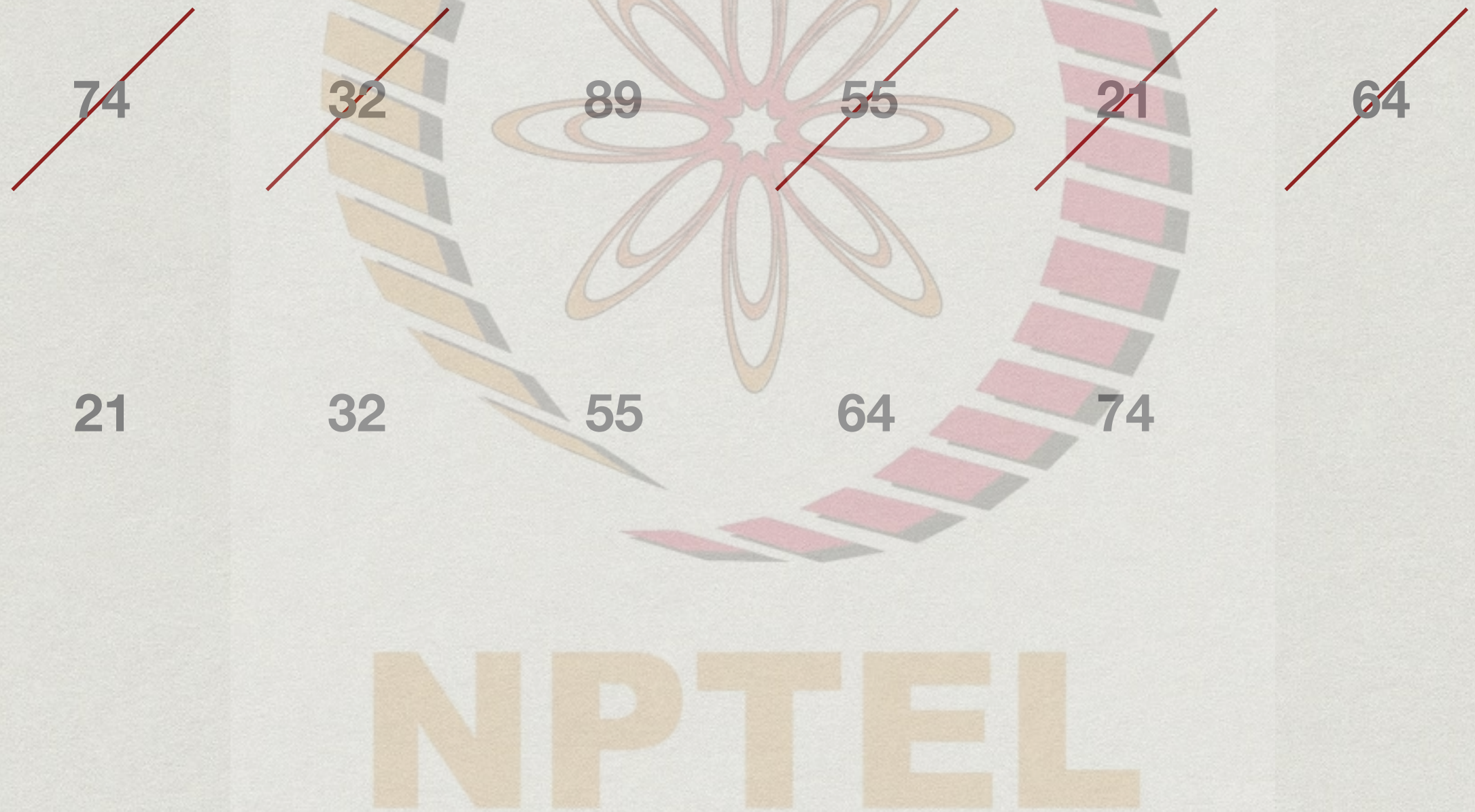
32

55

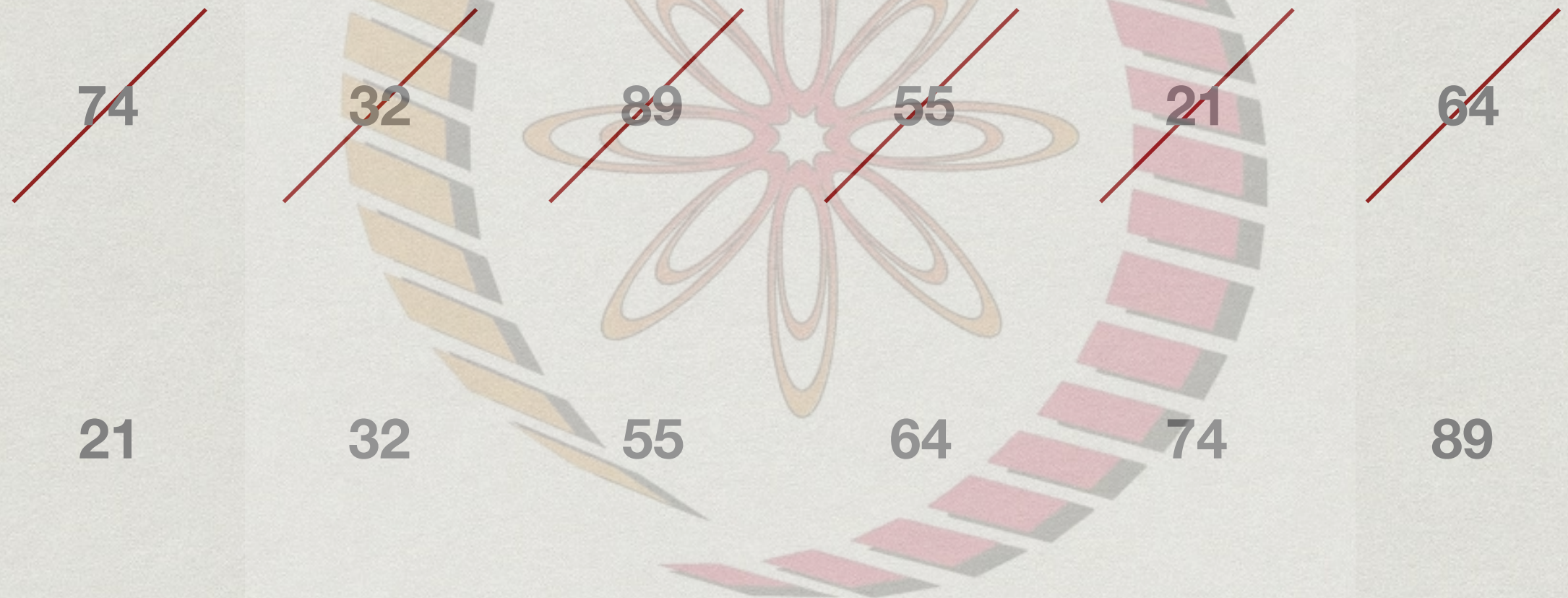
64

NPTTEL

Strategy 1 ...



Strategy 1 ...



NPTTEL

Strategy 1 ...

Selection Sort

- * **Select** the next element in sorted order
- * Move it into its correct place in the final sorted list

NPTTEL

Selection Sort

- * Avoid using a second list
- * Swap minimum element with value in first position
- * Swap second minimum element to second position
- * ...

NPTTEL

Selection Sort

74

32

89

55

21

64

NPTTEL

Selection Sort

74

32

89

55

21

64

NPTTEL

Selection Sort

21

32

89

55

74

64

NPTTEL

Selection Sort

21

32

89

55

74

64

NPTTEL

Selection Sort

21

32

89

55

74

64

NPTTEL

Selection Sort

21

32

89

55

74

64

NPTTEL

Selection Sort

21

32

55

89

74

64

NPTTEL

Selection Sort

21

32

55

89

74

64

NPTTEL

Selection Sort

21

32

55

64

74

89

NPTTEL

Selection Sort

21

32

55

64

74

89

NPTTEL

Selection Sort

21

32

55

64

74

89

NPTTEL

Selection Sort

21

32

55

64

74

89

NPTTEL

Selection Sort

```
SelectionSort(A,n) // Sort A of size n
```

```
for (startpos = 0; startpos < n; startpos++)
```

```
    // Scan segments A[0]..A[n-1], A[1]..A[n-1], ...
```

```
    // Locate position of minimum element in current segment
```

```
    minpos = startpos;
```

```
    for (i = minpos+1; i < n; i++)
```

```
        if (A[i] < A[minpos])
```

```
            minpos = i;
```

```
    // Move minimum element to start of current segment
```

```
    swap(A,startpos,minpos)
```


Analysis of Selection Sort

- * Finding minimum in unsorted segment of length k requires one scan, k steps
- * In each iteration, segment to be scanned reduces by 1
- * $t(n) = n + (n-1) + (n-2) + \dots + 1 = n(n+1)/2 = O(n^2)$

NPTTEL

Recursive formulation

- * To sort $A[i \dots n-1]$
 - * Find minimum value in segment and move to $A[i]$
 - * Apply Selection Sort to $A[i+1 \dots n-1]$
- * Base case
 - * Do nothing if $i = n-1$

Selection Sort, recursive

```
SelectionSort(A,start,n) // Sort A from start to n-1
```

```
if (start >= n-1)  
    return;
```

```
// Locate minimum element and move to start of segment
```

```
minpos = start;  
for (i = start+1; i < n; i++)  
    if (A[i] < A[minpos])  
        minpos = i;
```

```
swap(A,start,minpos)
```

```
// Recursively sort the rest
```

```
SelectionSort(A,start+1,n)
```


Alternative calculation

- * $t(n)$, time to run selection sort on length n
 - * n steps to find minimum and move to position 0
 - * $t(n-1)$ time to run selection sort on $A[1]$ to $A[n-1]$
- * **Recurrence**
 - * $t(n) = n + t(n-1)$
 $t(1) = 1$
 - * $t(n) = n + t(n-1) = n + ((n-1) + t(n-2)) = \dots =$
 $n + (n-1) + (n-2) + \dots + 1 = n(n+1)/2 = O(n^2)$